

UX from 30,000ft

A Guide to User Experience for Software Engineers and Developers

Simon Harper

UX from 30,000ft

A Guide to User Experience for Software Engineers and Developers

Simon Harper

This book is for sale at <http://leanpub.com/UX>

This version was published on 2023-10-10



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License](#)

Tweet This Book!

Please help Simon Harper by spreading the word about this book on [Twitter](#)!

The suggested hashtag for this book is [#UX30000](#).

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

[#UX30000](#)

Contents

Preamble	i
Text Structure	iii
Secondary ‘Text’	iv
AI Assistance	v

Part I: Introducing the User Experience **1**

1. Be Curious, Be Critical!	2
1.1 Being Critical	2
1.2 Being Curious	5
1.3 Summary	8
2. What is UX?	11
2.1 HCI Foundations	11
2.2 UX Emergence	12
2.3 The Importance of UX	15
2.4 Modern UX	16
2.5 Summary	23
3. UXD and Visual Design	25
3.1 User Experience (UX) and User Experience Design (UXD)	25
3.2 Design Systems and Languages	26
3.3 Interaction Design	29
3.4 Zero UI and Conversational Interfaces	30
3.5 Summary	31

Part II: Designing the User Experience **33**

4. People are Complicated!	34
4.1 UX is Everywhere	34
4.2 People & Computers	36
4.3 Input and Control	45
4.4 Summary	48
5. Practical Ethics	50
5.1 The Need for Ethical Approaches	50
5.2 Subjects or Participants?	52
5.3 Keeping Us Honest	53
5.4 Practical Ethical Procedures	54

CONTENTS

5.5	Potted Principles of Practical Ethical Procedures	55
5.6	Summary	60
6.	Gathering User Requirements	63
6.1	What is Requirements Analysis?	63
6.2	Digital Phenotyping	65
6.3	User Centred Design	65
6.4	What Information Do I Need?	70
6.5	How Do I Get ‘The Information That I Need’?	73
6.6	Summary	81
7.	Modelling Requirements	83
7.1	Informal Methods	83
7.2	Semi-Formal Methods	88
7.3	Formal Methods	92
7.4	Summary	100
Part III: Building the User Experience	102	
8.	Developing for UX	103
8.1	UX Development	103
8.2	Development Methodologies and Lifecycles	105
8.3	Methodologies More Suited to the UX Process	111
8.4	Separation of Concerns	114
8.5	Interface Frameworks and the GUI	120
8.6	Windows Toolkits	122
8.7	Summary	124
9.	Prototyping and Rapid Application Development	126
9.1	Prototyping	127
9.2	The Fidelity Spectrum	128
9.3	Prototypes in Software Engineering	130
9.4	Users, Commissioners, Engineers!	133
9.5	Rapid Application Development	135
9.6	Summary	138
10.	Principles of Effective Experience (Accessibility)	140
10.1	Effective, effectual, accessible	140
10.2	Barriers to Effectual Use	144
10.3	Technical Accessibility Issues	149
10.4	Potted Principles of Effectual User Experience	153
10.5	Summary	159
11.	Principles of Efficient Experience (Usability)	162
11.1	The Xerox ‘Star’	164
11.2	Universal Design and Design for All!	167
11.3	Usability Models	169
11.4	Collated Usability Principles, Guidelines, and Rules	171
11.5	Potted Principles of Efficient User Experience	174
11.6	Summary	181

CONTENTS

12. Principles of Affective Experience (Emotion)	185
12.1 Visceral, Behavioural, and Reflective	187
12.2 Narrative Art and Visual Aesthetics	192
12.3 Visual Attention	193
12.4 Collated Affective Concepts and Touch-points	194
12.5 Potted Principles of Affective User Experience	198
12.6 Summary	203
13. Principles of Engagement (Digital Umami)	206
13.1 Group Dynamics	207
13.2 Funology	209
13.3 Gamification	212
13.4 Collated Concepts of Engagement	216
13.5 Potted Principles of Dynamic User Experience	218
13.6 Summary	222
Part IV: Validating the User Experience	224
14. User Evaluation	225
14.1 Expert Evaluation via the Audit	226
14.2 Qualitative (Fieldwork) Methods	227
14.3 Quantitative & Hybrid Methods	229
14.4 Tools of the Trade	238
14.5 Summary	243
15. Human-in-the-Loop Systems and Digital Phenotyping	245
15.1 Human-in-the-Loop (HITL) Systems	245
15.2 Digital Phenotyping	248
15.3 Summary	254
16. Evaluation Analysis	256
16.1 Scientific Bedrock	257
16.2 Evaluation Design and Analysis	261
16.3 A Note on Statistics & How To Use & Interpret	270
16.4 Caveat	277
16.5 Summary	278
Part V: In Real Life	280
17. In Real Life	281
17.1 Realistic, Practical, Pragmatic, and Sloppy!	281
17.2 Expect Imperfection	282
17.3 Commissioning Constraints	284
17.4 Information Requirements	287
17.5 Limitations	288
17.6 Available Skills	290
17.7 Optimism	293
17.8 Summary	297

CONTENTS

18. Final Thoughts	299
18.1 Design	299
18.2 Development	300
18.3 Validation	301
18.4 As Practically Applied	302
18.5 Final Thoughts	303
Appendices	305
Appendix: Zen and the Art of Motorcycle Maintenance in UX	306
The Collision of Two Opposing Ideologies	306
Perception of the User Experience	307
The Discussion and Framing of Science in the User Experience	308
The Conceptualisation of Theoretical and Empirical User Experience	308
Rhetoric, Argumentation, and the User Experience	309
Values, and the Intangible Nature of the User Experience	310
The ZAMM Narrative Enhances the User Experience	310
Appendix: Defining UX	311
Appendix: Defining Accessibility	314
Appendix: Ethics Annex	316
Thanks...	321
References	322
Glossary of Terms	329

Preamble

Welcome to User Experience! For many of you, this is your first exposure to the wider discipline of Human Computer Interaction (HCI) and Human Factors. Some of you may have already experienced some basic HCI work as part of the software engineering lifecycle. However, this text takes a far more in-depth look at the tools, techniques, and knowledge you need to understand the user experience within software engineering.

It is worth noting that this text may be more in depth than those to which you may already have been exposed. But remember, it is by no means intended to teach you everything you need to know about human facing software engineering. Indeed, entire undergraduate degrees are built around the subject of human factors and ergonomics, and you should not confuse this high-level overview of the domain with the knowledge you would acquire in a full three-year degree programme. That said, the aim of this text is to give you, the reader, the tools, techniques, and the mindset necessary to competently approach your first user testing and user experience job. The text is designed from a practical perspective and will enable you to take a junior role in a user experience department or usability company. And will provide you with the overall knowledge to communicate with others and make sensible suggestions regarding UX work. Further, it will provide you with a basis for future self-study within the UX domain or the wider human factors world. In this case, I've decided to call the text, more fully, 'User Experience from 30,000 feet'¹.



Figure: Tom's Diner. "Tom's Diner" is an a cappella pop song written in 1981 by American singer-songwriter Suzanne Vega. Vega wrote the song based on a comment by her friend Brian Rose, a photographer, who mentioned that in his work, he sometimes felt as if "he saw his whole life through a pane of glass, and ... like he was the witness to a lot of things but was never really involved in them" —Image Credit: Wikipedia.

¹UX does have one advantage in that it is a new, practical, cross-disciplinary subject. No one has yet trained on a bespoke UX only degree programme, and so everyone has their own background and 'slant' to the area. There are many UX Industrial Departments/Companies, there are few UX courses. The combination of your technical Computer Science training coupled with this UX training will give you an advantage; a software focused UX professional.



Our need to understand the user experience manifests itself in many and varied ways; even in the most unexpected places. Go listen to the song '[Tom's Diner](#)' by Suzanne Vega – make sure it's the a Cappella version – now do a little Web searching and answer these questions:

1. What is the significance Tom's Diner in your everyday life?
2. Why is Tom's Diner significant for the User Experience?
3. What properties of Tom's Diner makes it so significant?
4. Why does the significance of Tom's Diner represent 'Good' science?

Now, don't panic I don't expect you to know the answers to all of these questions right now. But humour me – do some research and come up with some answers that seem plausible to you.

As we'll see later, the UX domain is still very young and in the process of formation. But, it does bring together many already established areas within the human-computer interaction field. As a placeholder, you may wish to think of UX as the practical application of research knowledge repurposed from other domains into the user-facing software engineering process. In this case, suggesting a particular UX text would only serve to skew your education to a particular view of the UX domain (although you may wish to check out some UX books as well: [[Unger and Chandler, 2009](#); [Bowles and Box, 2011](#); [Lund, 2011](#)]). I feel it is better for you to understand my view while realising there may be different ones out there. In the end, you will need to decide, after reading this text, the sort of UX you wish to do and how you think about the area. Therefore, this text has been written from scratch and represents an overview of UX as seen by the human factors and web science researchers I work with.

You don't need to panic! The text is intended to provide you with an overview of the UX field as well as an understanding of the key concepts within this field. Plus a method of assessing your understanding of the topics covered, and finally, a reference to reading material which will enable your deeper study, both now and in your future work.

Finally, I've tried to keep the jargon (and explain any I introduce²) to a minimum. And I hope you'll see that the writing style is 'formally academic' but clear and easy to read.

Origins

To give you an idea of how this text arose, it may be useful to give you an idea of its origins.

Similar to many Computer Science Departments, the School of Computer Science at Manchester has its roots in the Computer Group of the Electrical Engineering Department and the Computer Laboratory of the Department of Mathematics. After its formation in 1964, it continued to focus on computational hardware and low-level or embedded software. This focus resulted in the world's first stored-program computer; the Manchester Small-Scale Experimental Machine (SSEM), nicknamed 'Baby'. The School, run in the early days by such luminaries as Frederic C. Williams, Tom Kilburn,

²Email me with any jargon I haven't explained, or that you think I should explain better.

Geoff Tootill, and Alan Turing, continued to focus on the numerate, applied, and engineering aspects of Computer Science.

In this environment there was little focus on the human aspects of computer science. Human aspects that would inevitably come to play an important role as the computer moved from the laboratory to the desktop and onward to be embedded in many more products and devices. However, some of the researchers at the School became increasingly aware that we were not addressing areas which would – we thought – become increasingly important.

In light of this perception a user experience text was developed to serve as a cohesive starting point for us to be able to teach HCI and UX to technical Computer Science, Software Engineering, Hardware Engineering, and Artificial Intelligence students. It is therefore based on the human factors research activity already established within the School over the past 15 years, in support of the embryonic teaching activities arising after 2010.

Before we go any further it may also be useful to make one more point. You will notice that I use the term User Experience Specialist, usability specialist, UX'er, etc. throughout the book when referring to you the reader. This is because I've found it useful to reinforce your belief that this is what you can become as you learn more about the field. I think using this terminology also makes you more likely to think of yourselves in a different way to the 'developers' that you probably are, and the engineering domain you feel comfortable within.

Text Structure

UX is an empirical discipline, whereby established principles and guidelines are applied to software development and the correctness of that software is established via experimentation in the form of user feedback. In this case, the UX specialist is mainly interested in questioning the development to ascertain its correctness.

This correctness normally exists along many dimensions. But briefly, can be thought of as the primary utility of the development; its effectiveness; its efficiency; and its emotional impact (both tangibly and intangibly, represented by the concept of engagement). In this case, the text structure is intended to give you the principles, tools, and techniques needed for UX work along with a critical and questioning mindset to enable you to apply these techniques effectively.

Therefore, you will learn the tools and techniques needed for effective interaction with each dimension. These include the scientific and empirical techniques to join and apply these principles and tools to a new bespoke development. And the mindset to be critical of, and to question the outcomes of, this development to establish the user experience.

Secondary 'Text'

"Zen and the Art of Motorcycle Maintenance (ZAMM) [[Pirsig, 1974](#)]" may seem like a strange text to use for a Computer Science based UX course... and it probably is. In reality, I am not interested in you remembering anything in ZAMM – however the critical and questioning skills developed from its analysis will be of use to you. ZAMM is not really about Zen or indeed motorcycle maintenance, it's about science, quality, and rhetoric; the USA's Library of Congress describes it as:



"Acclaimed as one of the most exciting books in the history of American letters, this modern epic became an instant bestseller upon publication in 1974, transforming a generation and continuing to inspire millions. This 25th Anniversary Quill Edition features a new introduction by the author important typographical changes and a Reader's Guide that includes discussion topics, an interview with the author, and letters and documents detailing how this extraordinary book came to be. A narration of a summer motorcycle trip undertaken by a father and his son, the book becomes a personal and philosophical odyssey into fundamental questions of how to live. The narrator's relationship with his son leads to a powerful self-reckoning the craft of motorcycle maintenance leads to an austere beautiful process for reconciling science, religion, and humanism. Resonant with the confusions of existence, Zen and the Art of Motorcycle Maintenance is a touching and transcendent book of life."

While Wikipedia says:



"The book describes, in the first person, a 17-day journey on his motorcycle from Minnesota to California by the author (though he is not identified in the book) and his son Chris, joined for the first nine days by close friends John and Sylvia Sutherland. The trip is punctuated by numerous philosophical discussions, referred to as Chautauquas by the author, on topics including epistemology, ethical emotivism and the philosophy of science. Many of these discussions are tied together by the story of the narrator's own past self, who is referred to in the third person as Phaedrus (after Plato's dialogue). Phaedrus, a teacher of creative and technical writing at a small college, became engrossed in the question of what defines good writing, and what in general defines good, or 'quality'. His philosophical investigations eventually drove him insane, and he was subjected to electroshock treatment which permanently changed his personality."

ZAMM is useful to us because it serves as a framework for criticism while enabling us to apply UX principles, rational thought, and scientific argument to a well understood and well-respected text. As for my thoughts on ZAMM; well I might not agree with everything the book has to say, but I do think it is an important and interesting topic and discussion which needs to be had. At this point, you need not worry too much about the book itself – other than the fact that you should read it. All will become obvious as the text progresses. But you may like to read a longer discussion of "Zen and the Art of Motorcycle Maintenance" and its usefulness for UX Teaching in [the Appendix](#).

AI Assistance

As I began updating these notes at the beginning of 2023, ChatGPT and a slew of other AI ‘Assistants’ were released, to general uproar from thought workers, academics, and people in the digital space alike. What will this mean for learning, what will this mean for thought? Well, it is difficult to tell without trying to integrate them into your workflow and using them properly; not just as tests (no matter how extended). It is difficult to know how to think about AI assistants and so I thought I ask an AI.



How should I think about ChatGPT?

ChatGPT is an AI-powered language model developed by OpenAI, which uses a deep learning algorithm to generate responses to user input. As an AI language model, ChatGPT is designed to understand natural language and provide helpful and informative responses to user queries.

You can think of ChatGPT as a tool to help you find answers to your questions or provide you with information on a particular topic. While ChatGPT is not a human expert and may not always provide the exact answer you are looking for, it is capable of providing useful insights and pointing you in the right direction.

It is important to remember that ChatGPT’s responses are generated by an algorithm and may not always be accurate or completely up-to-date. Additionally, because ChatGPT is an AI language model, it may not be able to provide nuanced or context-specific responses in the way that a human expert might.

Overall, ChatGPT can be a useful tool for gaining insights and information, but it is important to use its responses as a starting point and to continue your own research and investigation to ensure that you have a complete and accurate understanding of the topic at hand.

And highlighting the most relevant parts of this response ‘it is capable of providing useful insights and pointing you in the right direction’ and ‘it may not be able to provide nuanced or context-specific responses in the way that a human expert might’ is the most useful and telling after my experiences here.

After using assistance for the update to this text, which in reality only represents around 15 pages of the work, 5 pages could be used for the most part as generated, 5 pages needed extensive modifications, and 5 pages need very extensive correction because concepts and elaborations were just plain incorrect; I found it to be useful for starting text to be modified, or for straightforward work which was just about connecting concepts. In general, it produces adequate but surface-level detail of concepts, does not provide context or nuanced discussion based on an understanding of the real world and so needs human oversight. However, it was very useful as a naive collaborator that provided initial writing waiting for the domain expert (me) to make subsequent ‘passes’ and edits. So if you are looking for text which is not expert but is a starting point then it is a very useful tool akin to spellcheckers or more recently intelligent grammar checkers. The most useful skill a human needs to learn when collaborating with an AI assistant is how to formulate the questions presented to it to generate the best answer.



However you should realise that everything in the text is written by me alone or in collaboration with any other's including an AI has my stamp of approval in that it is correct and that I want you to have the information.

Now, let's get on with understanding and shaping the User Experience!

Part I: Introducing the User Experience

1. Be Curious, Be Critical!

most people are fools, most authority is malignant, God does not exist, and everything is wrong.

– Ted Nelson

‘The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information’ [[Miller, 1956](#)] is a very famous human factors research paper, written in the mid-Fifties and concerning Psychoacoustics. The paper is interesting because it has spawned an often incorrectly understood, usability principle. That our working memory can only handle seven (± 1) items at any one time and therefore we should only make menus or lists a maximum of seven items long.

However, “Miller’s paper is a summary/analysis of research findings couched in the form of bits per channel. He amusingly relates the fact that the number seven seems to be plaguing his every move because seven seems to appear repeatedly regarding the amount of information that can be remembered or differentiated by humans. However, there are two caveats to this assertion: firstly, that the user is required to make absolute judgements of unidimensional stimuli and secondly that that stimulus is not clustered. This last point is quite important because using clustering means that we can remember or distinguish more than the unitary seven. For instance, we can remember seven characters in sequence, seven words in order, or seven phrases. In reality then, we have trouble differentiating uni-dimensional stimuli such as audible tones played without reference to each other, but we can differentiate more than seven tones when played in a sequence, or separately when multiple dimensions such as loudness and pitch are varied. Further, we can remember more than seven things within a list especially if those things are related or can be judged relatively, or occur as part of a sequence.

1.1 Being Critical

So this is a well found psychological finding (and very well written paper) that working memory can handle seven (± 2) arbitrarily sized chunks of absolute uni-dimensional stimuli. And becomes the often quoted but mostly incorrect usability/HCI principle that you should only include seven items in a menu, or seven items in a list... and on and on.”

Indeed, this often incorrectly understood usability principle is a misconception that has become a firmly held belief; this is the ‘wrongness’ that Ted alludes to, and there are plenty of others. Let’s look at a few.

1.1.1 Buzz and Hype

Jacob Nielsen famously suggests that usability evaluations can be conducted with only five people, and this will catch over 80% of the usability errors present [[Nielsen](#),

2000]. In reality, Nielsen added a lot of caveats and additional conditions to be met, which have been lost from the ‘buzz’ surrounding the ‘five users’ evaluation work. Eager usability practitioners hooked upon this five user message as a way of justifying small studies, even when those small studies tested multiple and disjoint usability tasks. Even user studies that do not try to generalise their results need to make sure that the kinds of tasks performed are both limited and holistic. With these kinds of usability tests, Faulkner [**Faulkner, 2003**], demonstrates that the amount of usability errors uncovered could be as little as 55%. Further, Schmettow [**Schmettow, 2012**] tells us that user numbers cannot be prescribed before knowing just what the study is to accomplish. And Robertson [**Robertson, 2012**] tells us that the interplay between Likert-Type Scales, Statistical Methods, and Effect Sizes are more complicated than we imagine when performing usability evaluations.

1.1.2 Shorthand

The Moon Orbits the Earth. More common misconceptions work their way into our understanding, usually because of a certain shorthand that makes the concepts more easily expressed, at the expense of introducing some minor errors. For instance, the Moon does not orbit the Earth (see figure [Moon’s Orbit](#)). Well, it kind of does, but in actuality the Moon orbits the combined centres of gravity of both the Moon and the Earth. This just so happens to be very near the centre of gravity exerted by the Earth because the Moon’s centre of gravity is so small.

In general, this misconception does not affect most people’s everyday life. However, it is technically important – how celestial bodies interact with each other is a major area within the study of Astronomy.

1.1.3 Sounds Right

The Earth is Closer to the Sun in the Summer than in the Winter.

Let’s look at another misconception, the Earth is closer to the Sun in the summer than in the winter, and it is this closeness which accounts for the seasons. Again, this is kind-of right, part of the Earth is nearer to the Sun in the summer than in the winter, however, the Earth as a whole is not. The seasons are accounted for by the Earth being tilted on its access by 23.4°. As the Earth orbits the Sun, different parts of the world receive different amounts of direct Sunlight and this accounts for the warming or cooling. But this warming or cooling is based on the angle of the Earth’s axis in relation to the Sun. And, therefore, the nearness of either the southern or the northern hemisphere to the Sun, not the Earth as a whole.

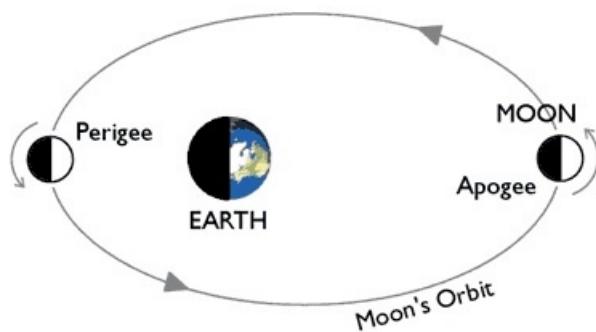


Figure: Moon’s Orbit. The Moon’s Orbit –Image Credit: sciencebrainwaves.com.

In your work as a user experience specialist, you often find there are many, unsaid, or shorthand's expressed by users. These are technically incorrect but are satisfactory for their everyday work, but may introduce large problems with your technical solutions.

1.1.4 Common Sense

Blind People Can't See. Certain facts seem to be 'common sense', you should always question common sense pronouncements. For instance blind people cannot see, this seems obvious, this seems like common sense, and in actuality it is incorrect. The reality of the situation is that some blind people, a very small percentage of blind people (termed profoundly blind) cannot see. In reality, the vast majority of blind people can see colours, shapes, blurred objects, and movement. In short most blind people can see something; only 4% can see nothing at all and are 'profoundly' blind.

1.1.5 Obviously!

Vision is Parallel, Hearing is Serial. It is often thought that vision is parallel, in that we can see multiple objects all at the same time whereas hearing is serial because we can only hear one sound. This is incorrect, in reality the basilar membrane of the cochlear experiences different frequencies from its base to apex (see [Figure: Basilar Membrane](#)). These nerve cells are connected to different areas within the brain that are responsive to different frequencies and those frequencies are also processed in parallel.

1.1.6 A Little Knowledge is a Dangerous Thing

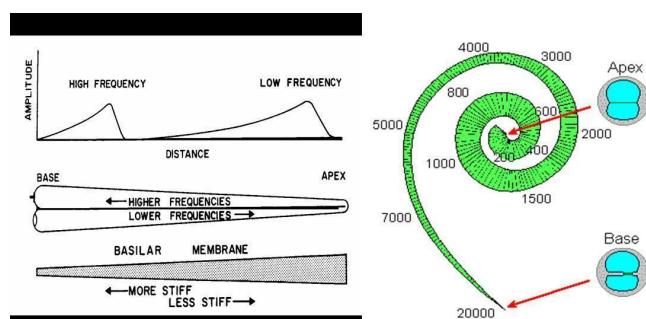


Figure: Basilar Membrane. The Basilar Membrane – Image Credit: cueflash.com.

one and within that time a concept called neuroplasticity is in effect. And states that the brain can change structurally and functionally as a result of input (or not) from the environment. This means that the brain can change and re-purpose unused areas [[Burton, 2003](#)]. For instance, the visual cortex of a child who has become profoundly blind at, say, ten is overtime repurposed to process hearing, and possibly memory and touch. The brain develops in this way repurposing unused areas and adapting to its environment until development slows down at the age of twenty-one.

All Brains Have the Same Organisation. We all know that the brain has the same organisation. Indeed, we can see this in many diagrams of the brain in which there are specific areas devoted to specific jobs, such as vision, language, hearing and the like. But just how true is this? In reality, this is not as true as you may imagine. The brain mostly develops up to the age of twenty-

1.1.7 One Final Thought...

Look at this famous video from Daniel Simons and Christopher Chabris¹. It requires you to count passes of a Basketball. Try to count them now, and then [skip ahead](#) for the result.

1.2 Being Curious

The story of UX and, more widely, HCI is not one you'd expect and you should always maintain a healthy curiosity when discussing it; it certainly isn't as two-dimensional as just front end coding.

1.2.1 The Collision of Two Opposing Ideologies

In the past we characterised practical human computer interaction in terms of usability and interaction engineering (in some cases accessibility was included but mainly as an afterthought). In this case, we decided if an interface was usable and the interaction design was good, based on tangible, measurable metrics such as task completion time. These kinds of metrics enabled us to understand the interactive experience in terms of time, theorising that the least time spent using the interface, the better; and this may have been, in some ways correct, as most computers were used in work situations.

As time has passed our concept of the computer and the interface evolved such that computers were no more tied down to the desk but could be mobile or ubiquitous, and the interface was not solely confined to software but may also include aspects of hardware, moving computers from the workplace and into the consumer product domain. Our ways of measuring and valuing the goodness of these interfaces however remained the same, task completion time, errors and error rates, correction times, Fitts Law pointing predictions, etc. Indeed this was the scientific or classic view of technology. At this point intangibles were seen as being soft science, unmeasurable and too open to incorrect interpretation. Other aspects, which might also affect how interfaces were experienced, but which could not be directly measured, and which relied more on subjective views of the user was seen as being, at best inconsequential, and at worst just plain old bad-science.

This clash of ideologies runs large through the whole of UX, its main theme being how to unify both the scientific and the romantic; the classic and the aesthetic; the tangible and the intangible; the measurable and the experiential. Indeed, UX is our attempt to unite classic HCI with modern ideas of experience and perception in which accessibility, usability, and interaction engineering (tangible, scientific, measurable) are combined with aesthetic, emotional, fun, affective, collaborative, and gameplay (intangible, humane, difficult to measure).

By understanding the clash of world we can also understand the more successful combination of these two – seemingly opposing – ideologies into a unified and cohesive whole as practically applied in UX. By understanding these nuances we can better understand what issue we will need to overcome in future UX design, build, and evaluation and

¹see <http://www.youtube.com/watch?v=vJG698U2Mvo>.

also begin to value – and better understand – the subjective qualitative views of the interactive experiences of our users.

1.2.2 Perception of the User Experience

One of the other major themes of UX is that of perception, and the differences which lie between people and their experience of the world. The complicated experiences and perceptions of experience should be taken as warnings to anyone working in user experience. Our perceptions are complicated and incredibly difficult to categorise; what may seem to be obvious to one person, maybe obscure to another. User experience, as opposed to classical HCI, takes these different subjective perceptions into account in its desire to create practical pleasing experiences for each user.

Perception is something not normally measured or quantified in classic HCI and so only in the modern additions of emotion, fun and dynamic interaction can these intangibles be acknowledged. UX shows us that experience can be massively divergent, and that outliers are as important as the general cases (if not more so).

1.2.3 Framing of Science in the User Experience

In our comparison of subjective and objective paradigms – the classical and the romantic – we must also take into account science and the scientific method especially in terms of empiricism, objectivity and the belief systems that arise around both objectivity and subjectivity.

Science has its limitations. For instance, many people have a common understanding of quality, in that they can tell quality when they see it, but quality is difficult to measure in any empirical or objective way; or describe with any degree of clarity. It seems in some ways an emergent property, or an umbrella term under which other more easily measured objective indicators can play a part. However, the richness of the description of quality is difficult to place only in such objective terms. So we can see that science cannot be the only measure of user experience, because science is mostly about generalisation, and because we do not have a full model of the universe; we therefore do not know all the variables which may arise to influence the user experience of a single individual. By nature we must conclude, in some regard, that objective, empirical science cannot give us all the answers at this time (until our model is complete), only the answer to testable questions.

This discussion of science is directly related to our discussions of the application of user experience, how we understand modern user experience, and how older styles of human computer interaction serve as an excellent base, but cannot provide the richness which is associated with the intangible, and often unquantifiable subjective, and emotional aspects which we would expect any user experience to comprise of.

We must, however, be cautious. By suggesting that subjective measures may not be testable means that we may be able to convince ourselves and others that a system is acceptable, and even assists or aids the user experience; while in reality there is no evidence, be it theoretical or experimental, which supports this argument. It may be that we are using rhetoric and argumentation to support subjective measures as a way of sidestepping the scientific process which may very well disprove our hypotheses as opposed to support it.

1.2.4 Theoretical and Empirical User Experience

Notice, in the last section we discussed one fundamental of science, the fact that we can disprove or support a hypothesis, in empirical work we cannot prove one. We cannot prove a hypothesis because in the real world we are not able to test every single condition that may be applied to the hypothesis. In this case we can only say that our hypothesis is strong because we have tried to destroy it and have failed.

However, those trained in rhetoric, in theoretical not empirical work, has a conception of science which is different to ours. In theoretical science it is quite possible to prove or disprove a hypothesis. This is because the model of the world is known in full ('closed world'), all tests can be applied, all answers can be evaluated. This is especially the case with regard to mathematics or theoretical physics whereby the mathematical principles are the way the world is modelled, and this theoretical world works on known principles. However this is not the case in user experience, and it is not the case in empirical science whereby we are observing phenomena in the real world ('open world') and testing our theories using experiments, which may be tightly controlled, but are often as naturalistic as possible. In this case, it is not possible to prove a hypothesis, because our model of the world is not complete, because we do not know the extent of the world, or all possible variables, in complex combination, which are able to affect the outcome.

In real-world empirical work we only need one negative result to disprove our hypothesis, but we need to have tested all possibilities to prove our hypothesis correct; we just don't know when everything has been tested.



This is often a reason why technologists, software engineers and the like feel uncomfortable in a UX setting where there are no 100% correct answers.

The worlds in which Code, Maths, Electronics and the like exist are closed worlds, and can be tested such that responses are not ambiguous. In the 'human' part of UX this is not the case, the world is open, not 100% testable, and prone to change.

I'm expecting that you can code the front-end, build patterns, create algorithms and the like. **However, I'm trying to show you how to be comfortable in ambiguity, while reducing that ambiguity such that it can be understood to acceptable levels of confidence.**

1.2.5 Rhetoric, Argumentation, and the User Experience

So, how can we satisfy ourselves that subjective, or intangible factors are taken into account in the design process and afterwards. While we may not be able to measure the subjective outcomes or directly generalise them we are able to rationalise these aspects with logical argumentation; and rhetoric – the art of using language effectively so as to persuade or influence others – can obviously play a key role in this. However, you will notice that the problem with rhetoric is that while you may be able to persuade or influence others, especially with the aid of logical argumentation, your results and premise may still be incorrect. If you haven't thought of, or don't predict, the [counter]arguments that will be made, and have your own convincing counter arguments you'll loose -

you may be right, but if you are, your arguments and counter arguments should be complete; that's the point of rhetoric and rhetorical debate.

Remember though, that with the user experience it is not our job to win an argument just for the sake of argumentation or rhetoric itself. We are not there to prove our eloquence, but we are there to support our inductive and deductive reasoning, and our own expertise when it comes to understanding the user experience within the subjective or intangible.

Further, the difference between a good UX'er and a bad one, like the difference between a good mathematician and a bad one, is precisely the ability to select the good facts from the bad ones on the basis of quality. You have to care! This is an ability about which formal traditional scientific method has nothing to say.

1.2.6 Values, and the Intangible Nature of the User Experience

'Values' are more important than you might imagine when working with humans, and it is useful to remember this in the context of understanding the user experience. In reality, many of the intangibilities which arise in UX work stem from these often hidden values.

More interestingly however, values which are often related to the world-views of individuals and more importantly their perception of these values, influences what they consider to be important.

We see the world not as it is, but as we are – or, as we are conditioned to see it.

– Anaïs Nin ... or Stephen R. Covey

Indeed, point here is that people bring their own values based on previous experience and their emotional state of equilibrium to any experience. These aspects are intangible and can be difficult to spot especially with regard to understanding the user experience. However, we can also use these values and world-views (if we have some idea of them) to positively influence users emotional response to an interface or interaction. Remember, we have already discussed that the expectation or perception of an experience, be it good or bad, will influence to a large degree the perception of that actual experience once enacted.

1.3 Summary

If there are two traits you should possess as a UX specialist, they are curiosity and the ability to be constructively critical. As we have seen, there are many reasons why errors and misconceptions can be embedded within both an organisation and the systems by which it runs. We can also see that if people want to believe their software has a good user experience they often will do. They are often blind to the real nature of the interface software, or systems, which they have built.

In UX, there is often no 100% correct answer (that we as UX specialists can derive) when it comes to creating, building, and then testing a system. The unique nature of

the human individual means that there are many subjective aspects to an evaluation. Understanding whether the interface is correct is often based on anecdotal evidence and general agreement from users, but is also based on a statistical analysis of quantitative measures. This area of uncertainty can allow rhetoric to be applied, whereby an argument may be proposed which seems stronger than your empirical evidence, but which is not empirically supportable. Also, there is also the danger that a sloppy or incorrect methodology will taint the empirical work such that the answers derived from a scientific method are incorrect. There are many cases of bad science², and incorrect outcomes from supposedly well-conducted surveys. For instance, evidence suggests that 95 percent of our decisions are made without rational thought. So consciously asking people how they will behave unconsciously is at best simplistic and at worst, can really mess up your study.

One of the most well-known examples is the launch of New Coke. “Coca-Cola invested in ‘cutting-edge customer research’ to ensure that New Coke would be a big success. Taste tests with thousands of consumers clearly showed that people preferred it. The reality was somewhat different, however. Hardly anyone bought it.”³

In another set of studies⁴, people were asked what messages would be most successful at persuading homeowners to make certain changes, such as turning down the heating, recycling more, and being more environmentally friendly.

Most said that “receiving information about their impact on the environment would make them change. However, this made very little difference at all. In another study, people who said that providing them with information about how much money they could save if they reduced consumption led to them to use even more! Interestingly, the message that most successfully changed their behaviour (information about how neighbours were making changes) was pretty much dismissed as unlikely to have any effect on them at all.”

Remember, quantitative instruments such as controlled field studies, or field observations, will often be cheaper in the long run than questionnaire-based approaches. In all cases, if you are not curious and you are not critical you will not produce accurate results.

1.3.1 Optional Further Reading

- [J. L. Adams]. Conceptual blockbusting: A guide to better ideas. Basic Books, 2019.
- [Joseph CR. Licklider]. “Man-computer symbiosis.” IRE transactions on human factors in electronics 1 (1960): 4-11.
- [R. M. Pirsig]. Zen and the art of motorcycle maintenance: an inquiry into values. Morrow, New York, 1974.
- [Suzanne Vega]. Tom’s Diner, A&M PolyGram, 1987.
- [Suzanne Vega] Vega, S. (2008, September 23). [Tom’s essay](#). The New York Times. Retrieved August 5, 2022.⁵

²see Ben Goldacre’s <http://www.badscience.net/>.

³see Steve J. Martin’s <http://scienceofyes.com/>

⁴Conducted by Wesley Schultz (Behavioural Scientist).

⁵<https://archive.nytimes.com/opinionator.blogs.nytimes.com/2008/09/23/toms-essay/>



Self Assessment Questions

Try these without reference to the text:

1. Why should you be cautious?
2. Why should you be curious?
3. Elaborate an example of common sense being incorrect.
4. Contrast the Open and Closed worlds view.
5. Why is Science sometimes not enough?

2. What is UX?

The human mind ... operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain. It has other characteristics, of course; trails that are not frequently followed are prone to fade, items are not fully permanent, memory is transitory. Yet the speed of action, the intricacy of trails, the detail of mental pictures, is awe-inspiring beyond all else in nature.

Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it. In minor ways he may even improve, for his records have relative permanency. The first idea, however, to be drawn from the analogy concerns selection. Selection by association, rather than by indexing, may yet be mechanised. One cannot hope thus to equal the speed and flexibility with which the mind follows an associative trail, but it should be possible to beat the mind decisively in regard to the permanence and clarity of the items resurrected from storage.

– Vannevar Bush

The idea of associative memory, first proposed by ‘[Vannevar Bush](#)’ in his 1945 Atlantic Monthly article ‘As We May Think’, is credited with being the inspiration, and precursor, for the modern World Wide Web. But for most of his article, Bush was not concerned solely with the technical aspects of his ‘MEMEX’ system. Instead, as with most computer visionaries, he was more concerned with how the computer system and its interfaces could help humanity. He wanted us to understand that instead of fitting into the way a computer interacts and presents its data, the human cognitive and interactive processes should be paramount. In short, the computer should adapt itself to accommodate human needs; not the reverse.

2.1 HCI Foundations

Since the early days of computer science, with the move from punch cards to QWERTY keyboards. From “[Doug Englebart’s](#)” mouse and rudimentary hypertext systems, via work on graphical user interfaces at Xerox PARC. To the desire to share information between any computer (the World Wide Web), the human has been at the heart of the system. Human-computer interaction then has had a long history in terms of computer science but is relatively young as a separate subject area. In some ways, its study is indivisible from that of the components which it helps to make usable. However, as we shall see, key scientific principles different from most other aspects of computer science, support and underlay the area, and by implication its practical application as UX.

We will discuss aspects of the user experience such as rapid application development and agility, people and barriers to interactivity, requirements gathering, case studies and focus groups, stories and personas. We’ll look at accessibility guidelines, and

usability principles, along with emotional design and human centred design. Finally, we'll touch on the scientific method, experimentation, and inferential statistics. This seems like quite a lot of ground to cover, but it is very small in relation to the wider Human Factors / HCI domain. For instance we won't cover:

- Adaptation;
- Customisation;
- Personalisation;
- Transcoding;
- Document Engineering;
- Cognitive Science;
- Neuroscience;
- Systems Behaviour;
- Interface Evolution;
- Emergent Behaviours;
- Application and User Agents;
- Widget Research & Design;
- Software Ethnography;
- Protocols, Languages, and Formats;
- Cognitive Ergonomics;
- Memory, Reasoning, and Motor Response;
- Learnability;
- Mental Workload;
- Decision-Making & Skilled Performance;
- Organisational Ergonomics;
- Socio-Technical Systems;
- Community Ergonomics;
- Cooperative Work;
- Inferential Statistics;
- Formal Experimental Methods; and
- Mobility and Ubiquity.

User experience (UX or UE) is often conflated with usability, but some would say takes-its-lead from the emerging discipline of experience design (XD). In reality, this means that usability is often thought of as being within the technical domain. Often being responsible for engineering aspects of the interface or interactive behaviour by building usability paradigms directly into the system. On the other hand, user experience is meant to convey a wider remit that does not just primarily focus on the interface but other psychological aspects of the use behaviour. We'll talk about this in more detail later, because as the UX field evolves, this view has become somewhat out of date.

2.2 UX Emergence

Human-Computer Interaction (HCI or CHI in North America) is not a simple subject to study for the Computer Scientist. HCI is an interdisciplinary subject that covers aspects of computer science, ergonomics, interface design, sociology and psychology. It is for this reason that HCI is often misunderstood by mainstream computer scientists. However, if HCI is to be understood and correctly applied then an enormous amount of effort, mathematical knowledge, and understanding is required to both create new

principles and apply those principles in the real world. As with other human sciences¹, there are no 100% correct answers, everything is open to error because the human – and the environments they operate within – are incredibly complicated. It is difficult to isolate a single factor, and there are many extraneous hidden factors at work in any interaction scenario. In this case, the luxury of a simple ‘yes’ or ‘no’ answer is not available².

The HCI field – of which UX is a part – is disparate, with each practitioner coming from a different specialism. In some cases psychology or ergonomics, in others sociology, for us, software engineering may be the primary specialisation, and in the context of UX, product designers also feature. This text has its roots firmly within the mainstream computer science domain, but it is aimed at a much broader audience. Therefore, whatever your background, you should find that this text covers the principle areas and key topics that you will need to understand, and manipulate the user experience. This means that, unlike other texts on UX, I will mainly be focusing on the tools and techniques required to understand and evaluate the interface and system. While I will spend one chapter looking at practice and engineering I feel it is more important to possess intellectual tools and skills. In this case, you will be able to understand any interface you work on as opposed to memorising a comprehensive treatise of the many

different interfaces you may encounter now or in the future. This is not to say that the study of past work, or best practice, is without value, but it should not be the focus of a compressed treatise such as this. As technically literate readers, I expect that you will understand computational terminology and concepts such as input³ and output⁴ conventions; Graphical User Interfaces (GUIs) conventions⁵; and other general terminology. I also assume that you will know next to nothing about UX of HCI.

HCI can normally be divided into three broad stages, that of the creation of the principles theories and methodologies, the application of those aspects into a development, and the testing of the outcomes of that development. While this may sound disjoint, techniques used for the investigation and discovery of problem areas in HCI are exactly

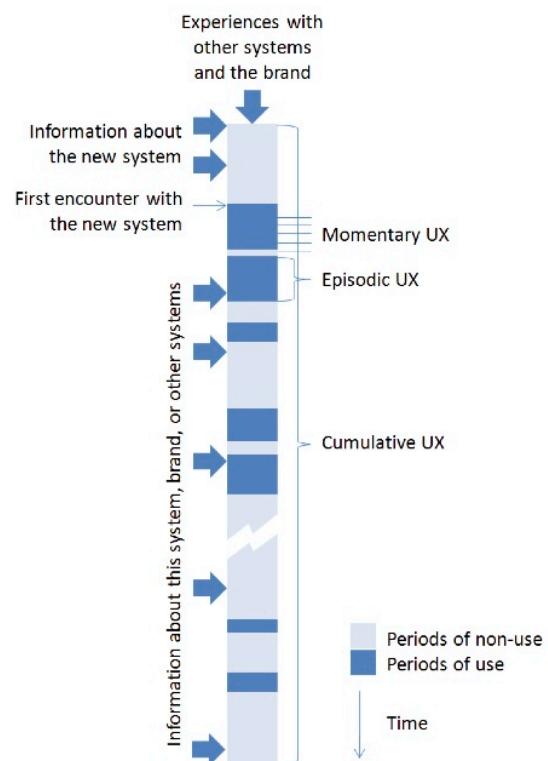


Figure: UX Periods. UX over time with periods of use and non-use –Image Credit: AllAboutUX.

¹Defining Human science within the context of HCI is a difficult proposition especially for an empirical, small ‘p’ positivist, such as myself. However, I see Human science as the investigation of human life and human activities that acknowledges the validity of data derived by impartial observation within an empirical framework. It includes the subjects such as history, sociology, anthropology, psychology and economics. And while not able to validate the subjective aspect of human life and activity, can contribute to an understanding of the human experience.

²The ‘up-side’ is that this level of complexity makes the study of the user experience incredibly interesting and incredibly challenging if done correctly.

³Such as the keyboard, mouse, trackpad, or trackball.

⁴Such as the monitor.

⁵Such as windows, panes, panels, and the like.

the same techniques that can be used to evaluate the positive or negative outcomes of the application of those techniques within a more production focused setting. Normally this can be seen as pre-testing a system before any changes are made, the application of those changes in software or hardware, followed by a final post-testing phase that equates to an evaluation stage in which the human aspects of the interface can be scientifically derived. This pre-testing is however, often missed during ‘requirements analysis’, in some cases because there is no system to pre-test, and in others because there is an over-inflated value given to the implicit understanding of the system already captured.

As a UX specialist, you will be concerned with the practical aspects surrounding the application of principles and guidelines into a development, and the testing of the outcomes of that development. This means that you will need to take into account the incremental nature of both the development and the experiences of the individual. Indeed we see that the user experience changes with time, and in some ways is linked with our memory and emotional state (see ‘Figure: UX Periods’ and ‘Figure: Time Spans of User Experience’ taken from [Roto et al., 2011]). In this way we can see that it is possible to counteract an initially bad systems experience, in a system that possibly must be complicated (such as an aircraft cockpit), by increasing the learnability of the controls and their layout. The initial momentary and episodic user experience may be complex – and in some cases seen as negative. However, the cumulative user experience may resolve as simple – equating to a positive user experience (especially in the case of the cockpit whereby the instrumentation, systems, and their layout are often replicated between aeroplanes, regardless of manufacturer or type).



In this case how does UX relate to the standard software engineering domain of requirements analysis? The definition of requirements in the IEEE standard 610 is given in three specific points:

“(1) a condition or capacity needed by a user to solve a problem or achieve an objective; (2) a condition or capability that must be met or possessed by a system or system component to satisfy a contract standard specification or other formerly imposed document; and (3) a documented representation of a condition or capability as in (1) or (2).”

We can, therefore, see that UX directly applies to condition ‘1’ indeed without access to, or the usability of, a system or component by a user this condition cannot be met. In the requirements analysis domain, work often progresses in a software engineering fashion. The methodology for requirements is coalesced around a set of modelling principles often initiated by a wave of interviews, discussions, systems analysis and modelling. In addition to focus-group participation leading to a formal specification or model of the systems and interaction requirement⁶ (requirements elicitation). These requirements have to be validated and modelled, however, as we shall see in later chapters, there are some problems with current approaches.

⁶Often as Universal Modelling Language (UML): Use Case Diagrams, the abstract description of user task composition and flow; Scenarios, Sequence Diagrams, and Narrative Text, the concrete description of user tasks; along with Class Diagrams, the description of user domain concepts; you probably already have experience of this from your second year software engineering course.

2.3 The Importance of UX

While the use of HCI as a tool for knowledge discovery is important, the significance of UX in the interface design and engineering domain should not be overlooked. User facing aspects of the interface are often created by software engineers or application programmers. While these are highly trained specialists, they are often less focused on aspects of user interaction than they are on the programme functionality and logic. In some cases, the end user is often seen as a silent participant in the application creation process and is usually only considered once the system aspects of the development have been created and tested. Indeed, in many cases there is an implicit idea that the user will need to conform to the requirements of the system, and the interface created by the developers. As opposed to the design of the system being a collaborative activity between user and engineer.

The focus of the UX specialist then is to make sure that the user is taken into account from the start. That participation occurs at all stages of the engineering process; that the resultant interface is fit for purpose in its usability and accessibility. And that, as much as possible, the interface is designed to fit the user and provoke a positive emotional response. By trying to understand user requirements, concerning both the system and the interface, the UX specialist contributes to the overall application design lifecycle. Often, in a very practical way which often belies the underlying scientific processes at work.

As we have seen, both requirements analysis and requirements elicitation are key factors in creating a usable system that performs the tasks required of it by the users and the commissioners of the system. Typically, however, requirements analysis and elicitation are performed by systems analysts as opposed to trained UX, human factors, or ergonomic specialists. Naturally, this means that there is often an adherence to set modelling techniques, usually adapted from the software architecture design process. This inflexibility can often be counter-productive because adaptable approaches of inquiry are often required if we are to better understand the user interaction. In reality, user experience is very similar to usability, although it is rooted within the product design community as opposed to the systems computing community of usability. Indeed, practical usability is often seen as coming from the likes of [Nielsen](#), [Shneiderman](#), and lately [Krug](#); while user experience came from the likes of [Norman](#), [Cooper](#), and [Gerrett](#). Although, thinking also suggests these views are becoming increasingly popular⁷:



"closely coupled in practice with a deeply anti-intellectual strain that wants to remove effort, learning, and expression from computing and that valorizes the new user to the exclusion of everything else. Today's software marketplace exaggerates this: what demos well, sells, and once you've made the sale, it's time to move on to the next user."

In this case, the usability specialist would often be expected to undertake a certain degree of software engineering and coding whereas the user experience specialist was often more interdisciplinary in focus. This meant that the user experience specialist might undertake the design of the physical device along with a study of its economic traits but might not be able to take that design to a hardware or software resolution. Indeed, user experience has been defined as:

⁷Mark Bernstein's - Creator of 'TinderBox' - [view](#).



"pertaining to the creation of the architecture and interaction models that impact a user's perception of a device or system. The scope of the field is directed at affecting all aspects of the user's interaction with the product: how it is perceived, learned, and used."

Therefore, user experience is sometimes seen as less concerned with quantifiable user performance but more the qualitative aspects of usability. In this way, UX was driven by a consideration of the ‘moments of engagement’, known as ‘touchpoints’, between people and the ideas, emotions, and the memories that these moments create. This was far more about making the user feel good about the system or the interface as opposed to purely the utility of the interactive performance. User experience then fell to some extent outside of the technical remit of the computer science-trained HCI specialist.

While once correct these views do not represent the current state of UX in the computer engineering domain. In this case, it is the objective of this text to provide you with an overview of modern UX. Along with the kinds of tools and techniques that will enable you to conduct your own well-formed scientific studies of human-facing interfaces and systems within the commercial environment.

2.4 Modern UX

Defining UX is akin to walking on quicksand. There is no firm ground, and you’re likely to get mired in many unproductive debates – indeed, to me it seems debates on definitions are currently ‘stuck in the muck’.

But why is defining UX important or even necessary? Well, it must be necessary because everybody seems to be doing it. Indeed ‘All About UX’ (AllAboutUX) have collected many definitions (see [the Appendix](#)) with multiple and different perspectives. Further, it’s important because it provides a common language and understanding, and a solid, succinct definition enables everyone to know where they’re going and – in some regard – predicts the road ahead.

So why is it such a problem? It seems to me that there is no clear definition of user experience because it is not yet a distinct domain. Everyone is an immigrant to UX⁸, and there are no native UX practitioners or indeed first-generation educated practitioners who share a common understanding of what the phenomenon of UX actually is. This is always the case with new, cross-disciplinary, or combinatorial domains. But this does not help us in our efforts to describe the domain such that we all understand what it is we do, where it is we are going, and what falls inside or outside that particular area.

2.4.1 The UX Landscape

Why am I so concerned with **Law's CHI 2009** paper and the work coalescing around AllAboutUX? The positive point about both of these sources is that they are created based on a community understanding of the area, which implicitly defines the landscape of the user experience domain. In other definitions, created by experts in the field, you are asked to subscribe to the author’s interpretation. Both Law and AllAboutUX base

⁸Indeed, if you’re reading this - you are probably an immigrant from the ‘technical’ Computer Science / Software Engineering domain.

their work on the populous view making little interpretation and allowing others to see the large differences within the comprehension, understanding, and definition of the UX field.

	#6: Subjectivity		#20: Emotional attachment		#22: Qualitative approach	
	M	SD	M	SD	M	SD
FI	3.98	.87	2.64	1.13	3.89	.84
USA	2.93	1.09	1.93	.69	3.26	1.2
UK	3.71	1.02	2.03	.88	3.2	1.16
NL	3.47	1.16	2.00	.87	3.74	1.00

Table: UX Country Differences. Differences among countries of residence – Credit: Law, 2009.

Indeed, purely by a cursory analysis of both sources we can see that there are major differences in understanding the subjectivity, emotional attachment, and qualitative approaches, even at the coarse-grained level of countries. Indeed, ‘[Table: UX Country Differences](#)’ shows us that the USA sees UX as less subjective than the other countries. However, all countries show a high degree of variance in the answers given. Further, all countries agree on the emotional attachment aspects of user experience but see this as being reasonably low as a factor in the UX landscape. However, all countries seem to agree that UX can be characterised by its qualitative approaches (as opposed to quantitative approaches) to understanding the experience.

#	Statement	N /275	Response Rate	M	SD	95 ci lower	95 ci upper
3	Fleeting and more stable aspects of a person's internal state (e.g., needs, motivations) affect a person's experience of something	261	95%	4.47	.04	4.40	4.54
5	UX occurs in, and is dependent on the context in which the artefact is experienced	265	96%	4.32	.05	4.22	4.42
8	Prior exposure to an artefact shapes subsequent UX	257	93%	4.25	.05	4.16	4.34
18	Designing (for) UX must be grounded in user-centred design	265	96%	4.11	.07	3.98	4.24
23	UX can change even after a person has stopped interacting with the artefact	259	94%	3.93	.06	3.82	4.03
11	UX is based on how a person perceives the characteristics of an artefact, but not on the characteristics per se	251	91%	3.89	.07	3.75	4.03
17	UX should be assessed while interacting with an artefact	260	95%	3.87	.06	3.75	4.00
14	Measuring UX implies determination of merits, values, and significance of an artefact in relation to a person's goals and needs	249	91%	3.84	.06	3.73	3.96
13	We cannot design UX, but we can design for UX	249	91%	3.82	.07	3.68	3.96
1	UX is highly dynamic - it changes constantly while interacting with a product	264	96%	3.76	.07	3.63	3.89
12	Usability is a necessary precondition for good UX	269	98%	3.70	.07	3.56	3.84
2	Imagined use of a product can result in real experiences	235	85%	3.66	.06	3.53	3.78
15	UX refers to affective states, i.e., any combination of valence (good - bad, pleasant – unpleasant) and physiological arousal (calm – excited)	252	92%	3.60	.06	3.48	3.72
22	UX must be approached qualitatively	265	96%	3.59	.07	3.46	3.72
6	UX is not about people's performance (ability to understand and use) in their relation with an artefact, but about the person's perception of that performance	266	97%	3.58	.07	3.44	3.73
16	UX can be quantified and thus compared across similar (or competitive) artefacts	263	96%	3.50	.06	3.38	3.62
7	There is a definite need for a standardized definition of the term UX	268	97%	3.49	.07	3.34	3.63
10	UX should be assessed after interacting with an artefact	255	93%	3.33	.06	3.20	3.45
19	Only an individual person can have an experience. An experience is something personal, something 'within' a person	265	96%	3.16	.08	3.00	3.32
9	People will never have comparable UX - each and every interaction with a product results in a unique experience	268	97%	2.71	.07	2.57	2.84
21	UX is not new, it is already covered by existing engineering approaches	263	96%	2.56	.07	2.42	2.70
20	UX is equal to emotional attachment	261	95%	2.27	.06	2.15	2.39
4	UX is best viewed in terms of marketing	262	95%	1.90	.06	1.79	2.00

Table: Twenty-three Statements About UX. Twenty-three statements about UX sorted by mean agreement (M) – Credit: Law, 2009.

We can also infer (see tables taken from [\[Law et al., 2009\]](#)) that the community sees user

experience as a lens into a person's internal state. A state that affects their experience of the software or system and that this experience must take place within the presence of the software or system they are interacting with. And that it is dependent on their prior exposure to that software or system. Including its longitudinal aspects (see '[Table: Twenty-three Statements About UX](#)'), and that their responses are based on their 'perceptions' of the software or system they are interacting with — as opposed to the true properties of that system.

Further, the community sees that UX must be grounded in user centred design, **we cannot design UX, but that we can design for UX**. We can also see that the community do not think that UX is best viewed in terms of marketing. Or that UX is only equal to emotional responses (affective), or that UX is so individual that there will never be an overlap between people and products. Indeed, extrapolating from these results we can see that most UX specialists do not see the need for a high number of users in their testing and evaluation phases, or the need for quantitative statistical analysis. Instead, they prefer a low number of users combined with qualitative output and believe this can be extrapolated to a large population because people have comparable user experiences.

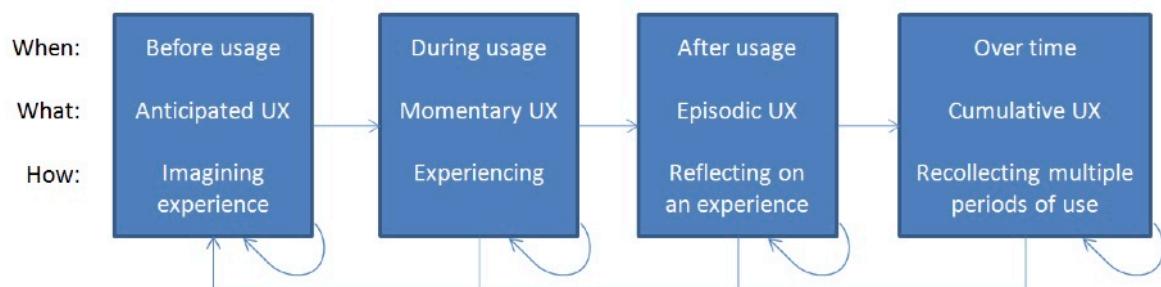


Figure: Time Spans of User Experience. Time spans of user experience, the terms to describe the kind of user experience related to the spans, and the internal process taking place in the different time spans —Image Credit: AllAboutUX.

It seems that the most important parts of the UX landscape (see '[Table: Twenty-three Statements About UX](#)'), its nature and the ideas that are key to its understanding and application, can be summarised from the comments Law received. We could say that the nature of UX is multi-layered and concerns the user's total experience (including their emotions and feelings) based on their current changeable internal state. UX is often socially constructed and represents the cumulative impact of interactions between users and the software or system, these interactions can be qualitatively and quantitatively measured. Key concepts through the UX landscape include the idea that not only the person, but also the artefact, and environment are equally important. And that interactions contain (un)conscious components, intangible aspects, actual and perceived interactions, and that the users' entire and broad experiences are valuable.

How is the Def? (Characteristics)		The Def is for? (Potential Uses)	The Def says UX is? (Nature of UX)	The Def bespeaks? (Key ideas about UX)
Positive	Negative			
<ul style="list-style-type: none"> • <i>comprehensive</i> • <i>easy to understand</i> • <i>simple</i> • <i>clear</i> • <i>concise</i> • <i>accurate</i> • <i>neutral</i> • <i>open</i> • <i>specific</i> • <i>direct</i> • <i>scientific</i> • <i>structured</i> • <i>system-oriented</i> • <i>usable</i> • <i>vague</i> • <i>descriptive</i> • <i>dictionary-like</i> • <i>high-level</i> • <i>integrative</i> • <i>memorisable</i> 	<ul style="list-style-type: none"> • ambiguous • circular • hard to sell • non-scientific • too academic • too broad for practice • too cognitivistic • too detailed • too dogmatic • too esoteric • too logical • too many examples • too strictly focused • wordy 	<ul style="list-style-type: none"> • <i>identify all the important factors to be studied</i> • <i>enable general public to understand UX</i> • identify measurable aspects of UX • drive further research and development • provide a structure of UX • scoping of UX • serve as guidelines • provide a concrete set of attributes that people can relate to • provide pointers to select appropriate combination of methods for a product 	<ul style="list-style-type: none"> • layered • lived-experience • socially constructed • task achievement • total brand experience • user's internal state • emotion • cumulative impact of interactions between users and products/services • cognitive • all feelings • experienced quality 	<ul style="list-style-type: none"> • three dimensions: person, artefact, and environment • types of interactions: (un)conscious • value in a set of affect • intangible aspects of UX • complexity of experience • actual usage • entire user perceived experience • examples • a broad set of experiences with the company • what causes UX • not-marketing related

Figure: UX Picked Definitions. Analysis of the comments on the picked definitions –Image Credit: Law, 2009.

2.4.2 Caveat

As we have [already seen](#), everything we discover should be looked at with a critical eye, including definitions and understanding of the UX landscape. So why may there be a problem with the landscape we have built in the previous section. Indeed from an analysis of Law's paper the responses from the community seemed to be reasonably strong — why should we, then, not believe them?

Firstly take a look at the demographics [[Law et al., 2009](#)]), it seems that this questionnaire was completed by far more industrial practitioners than academic ones — this may have skewed the results. Further, only 27 people said they were educated in art and design therefore we may not be getting a full view of the arts and humanities area — and how it is perceived from a non-technical viewpoint. One-hundred and twenty-three people said that they were interested in understanding UX to design better products, suggests that product designers views, and not software technologists, may have also skewed the results. In addition look at the countries, the authors say '[people from] 25 countries [responded], with larger groups of respondents from Finland (48), USA (43), UK (36), and the Netherlands (32)' so obviously, the views from these countries will dominate. One of the main points to consider is the presumptions of the authors, as expressed in the five definitions used in the survey (see '[Figure: UX Five Definitions](#)'). The creation of these definitions implies the authors desire to elicit broad agreement (or not) and so their creation may introduce some degree of bias. And to some extent represents the authors view and implies that this view has some priority, as opposed a choice made by the UX participants.

D1	All aspects of the end-user's interaction with the company. Its services and its products. The first requirement for an exemplary user experience is to meet the exact needs of the customer without fuss or bother. Next comes simplicity and elegance that produce products that are a joy to own, a joy to use. True user experience goes far beyond giving customers what they say they want, or providing checklist features. [8]
D2	A consequence of a user's internal state (predispositions, expectations, needs, motivation, mood, etc.) the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc.) and the context (or the environment) within which the interaction occurs (e.g. organisational/social setting, meaningfulness of the activity, voluntariness of use, etc.) [7]
D3	The entire set of affects that is elicited by the interaction between a user and a product including the degree to which all our senses are gratified (aesthetic experience) the meanings we attach to the product (experience of meaning) and the feelings and emotions that are elicited (emotional experience). [3]
D4	The value derived from interaction(s) [or anticipated interaction(s)] with a product or service and the supporting cast in the context of use (e.g. time, location, and user disposition). [20]
D5	The quality of experience a person has when interacting with a specific design. This can range from a specific artefact such as a cup toy or website up to larger integrated experiences such as a museum or an airport. [9]

Figure: UX Five Definitions. Five definitions used in the survey –Image Credit: Law, 2009.

This said, it is still my opinion that the authors have done everything possible to be inclusive and to represent the communities view of UX, however disjoint that may be, in an accurate and informative way. By investigating the domain more fully, taking into account other sources, we too can make more accurate appraisals of the UX landscape. And come to our own definition and understanding of what it means to be a UX specialist within that landscape.

As we have already discussed, the concept of UX is not nascent, indeed it has been around for 20-30 years. However, the formalization and professionalization of UX design as a distinct discipline are relatively recent developments.

The term “user experience” was first coined in the 1990s by Don Norman, a cognitive scientist who worked at Apple and other technology companies. Norman’s work on user-centered design and usability had a significant impact on the design industry, and his ideas helped establish the importance of designing products with the user in mind. However, it wasn’t until the early 2000s that UX design began to emerge as a distinct profession with its own set of practices and methodologies. As the importance of user-centered design became more widely recognized, companies began to hire dedicated UX designers and create UX teams to focus specifically on designing products that meet the needs of their users.

In this case, there are certain aspects of the field that are still evolving and can be considered nascent; which include:

- **Emerging technologies:** As new technologies like virtual and augmented reality, artificial intelligence, and the Internet of Things continue to develop, UX designers will need to adapt their design methodologies to account for these new contexts and use cases.
- **Inclusive design:** While inclusive design has been a topic of discussion for many years, it is only recently that it has become a more prominent consideration in UX design. Inclusive design involves designing products and services that are accessible and usable by people of all abilities, ages, and backgrounds.
- **Data-driven design:** The use of data and analytics to inform design decisions is still a nascent aspect of UX design. While many companies are already using data to measure user behavior and improve their products, there is still much to be learned about how to effectively incorporate data into the design process.
- **Ethical design:** With the increasing attention being paid to data privacy and the ethical implications of technology, UX designers are increasingly being called upon to consider the ethical implications of their designs. This includes issues like data privacy, algorithmic bias, and the impact of technology on society as a whole.
- **Cross-functional collaboration:** While UX designers have long worked closely with other members of the design team, there is a growing recognition of the importance of cross-functional collaboration in UX design. This involves working closely with stakeholders from other areas of the business, such as product management, engineering, and marketing, to ensure that the user experience is aligned with business goals.

Overall, there are still many aspects of the discipline that are evolving and changing as technology, society, and business needs continue to evolve.

2.4.3 My View

I've previously written about my idea of UX. Having seen and read a number of definitions that suggest that UX is more about emotion and may be layered on top of other aspects of software engineering and development such as usability. However, the more I dig, the more I realise that I really do not believe any of the definitions as presented; either via the excellent work of Effie Law, or those parties coalesced around AllAboutUX and led primarily by Virpi Roto.

So what do I believe?

1. I believe that UX is primarily about practice and application;
2. I believe it is an umbrella term for a multitude of specialisms;
3. I believe it is a phenomenon in that it exists and is observable;
4. I believe that this phenomenon collects people, methods, tools, and techniques from the wider human factors domain and combines them for practical application;
5. I do NOT believe that UX is a primary research domain but rather that UX is the practical application of a particular combination of tools, techniques, methods, principles, and mindset. And pulled in from primarily human factors and therefore psychology, social science, cognitive science, human-computer interaction, and secondarily product design, and marketing;

6. I do believe that UX is a secondary field of study, if the narrow definition of UX is mainly concerned with emotional indicators is used. However, I believe this is more properly defined as 'affective experience'; further
7. I do not believe that UX is a 'layer' in the software artefact route to development but rather describes that software artefact in a holistic way.

Indeed, I see UX as a combination of the following properties⁹:

- **Utility**
the software in development must be useful, profitable, or beneficial;
- **Effective in Use**
the software must be successful in producing a desired or intended result (primarily the removal of technical barriers particularly about accessibility);
- **Efficient in Use**
the software must achieve maximum productivity with minimum wasted effort or expense (primarily the removal of barriers in relation to usability and interactivity);
- **Affective in Use**
the software must support the emotional dimension of the experiences and feelings of the user when interacting; anticipating interaction, or when remembering interaction, with it; and
- **Engaging in Use**
the software may exhibit an intangible dynamic deliciousness (umami) concerning the fun a system is to use.

So, my definition (and this may evolve) would be:

"User Experience is an umbrella term used to describe all the factors that contribute to the quality of experience a person has when interacting with a specific software artefact, or system. It focuses on the practice of user centred: design, creation, and testing, whereby the outcomes can be qualitatively tested using small numbers of users."

In fact on 26th January 2014 it did evolve to:

"User Experience is an umbrella term used to describe all the factors that contribute to the quality of experience a person has when interacting with a specific software artefact, or system. It focuses on the practice of user centred: design, creation, and testing, whereby the outcomes can be qualitatively evaluated using small numbers of users."

And again on 04th August 2022 to:

"User Experience is an umbrella term used to describe all the factors that contribute to the quality of experience a person has when interacting with a specific technical artefact, or system. It focuses on the practice of requirements gathering and specification, design, creation, and testing, integrating best practice, heuristics, and 'prior-art' whereby the outcomes can be qualitatively evaluated using small numbers of users, placing humans firmly in the loop"

⁹...and so this is how this text is structured.

2.5 Summary

	D1	D2	D3	D4	D5
Total	46	65	44	19	36
% out of 210	22%	31%	21%	9%	17%

Figure: Preferred Definitions. Distributions of the preferred definitions –Image Credit: Law, 2009.

As we can see, HCI is one of the most important aspects of computer science and application development. Further, UX is a mostly applied sub-domain of HCI. This is especially the case when that application development is focused on providing humans with access to the program functionality. But this is not the only concern of UX, indeed for many, it is the augmentation of the interactive processes and behaviours of the human in an attempt to deal with an ever more contemplated world that is the focus. This augmentation does not take the form of artificial intelligence or even cybernetics, but by enabling us to interact with computer systems more effectively, to understand the information that they are processing, and to allow us to focus more completely on the intellectual challenges; as opposed to those which are merely administrative or banal. Indeed, this objective has been Douglas C. Engelbart's overarching aim since 1962:

"By augmenting human intellect we mean increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to problems. Increased capability in this respect is taken to mean a mixture of the following: more-rapid comprehension, better comprehension, the possibility of gaining a useful degree of comprehension in a situation that previously was too complex, speedier solutions, better solutions, and the possibility of finding solutions to problems that before seemed insoluble."

In the context of human factors, HCI is used in areas of the computer science spectrum that may not seem as though they obviously lend themselves to interaction. Even strategies in algorithms and complexity have some aspects of user dependency, as do the modelling and simulation domains within computational science.

In this regard, HCI is very much at the edge of discovery and the application of that discovery. Indeed, HCI requires a firm grasp of the key principles of science, scientific reasoning, and the philosophy of science. This philosophy, principles, and reasoning are required if a thorough understanding of the way humans interact with computers is to be achieved. In a more specific sense if you need to understand, and show, that the improvements made over user interfaces to which you have a responsibility, in fact, real and quantifiable.

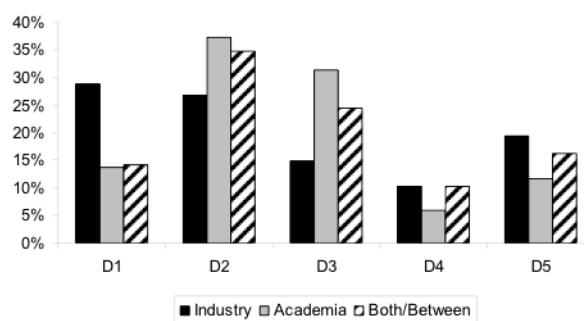


Figure: Preference by Work Place. Definition preference by the work place –Image Credit: Law, 2009.

In reality, it is impossible for a text such as this to give an all-encompassing in-depth analysis of the fields that are discussed; indeed this is not its aim. You should think of this text as a route into understanding the most complex issues of user experience from a practical perspective. But you should not regard it as a substitute for the most in-depth treatise presented as part of the further reading for each chapter. In reality, the vast majority of the work covered will enable you to both develop well-constructed interfaces and systems based on the principles of user experience; and run reasonably well-designed evaluations to test your developments.

2.5.1 Optional Further Reading

- [A. Dix] J. Finlay, G. Abowd, and R. Beale. Human-computer interaction. Prentice Hall Europe, London, 2nd ed edition, 1998.
- [C. Bowles] and J. Box. Undercover user experience: learn how to do great UX work with tiny budgets, no time, and limited support. Voices that matter. New Riders, Berkeley, CA, 2011.
- [R. Unger] and C. Chandler. A project guide to UX design: for user experience designers in the field or in the making. Voices that matter. New Riders, Berkeley, CA, 2009.
- [T. Erickson] and D. W. McDonald. HCI Remixed: essays on works that have influenced the HCI community. MIT Press, Cambridge, Mass., 2008.



Self Assessment Questions

Try these without reference to the text:

1. What is the key focus of HCI?
2. What is the purpose of the UX specialist?
3. What is User Experience and how is it applied?
4. If there are no 100% correct answers in UX, how do we decide what is right and what is wrong?
5. What are the five key properties of UX?

3. UXD and Visual Design

“The public is more familiar with bad design than good design. It is, in effect, conditioned to prefer bad design, because that is what it lives with. The new becomes threatening, the old reassuring.”

– Paul Rand, graphic designer¹

We have already seen, even though 20 years or so old, user experience is still open to many definitions and interpretations; just what UX means, and includes, is a matter of debate and you need to be sure that you are explicit about just what it means for you. Mostly, we can agree that it is about placing emphasis on the user and expanding our concept of their experiences, in terms of technology and surrounding aspects outside of the interface. For this reason, there are different types of focus required from different technical specialists.

3.1 User Experience (UX) and User Experience Design (UXD)

It will be common for you to hear the term UXD (or User Experience Design), which means ‘User eXperience Design—er’. Typically, this specialism is focused on the look and feel of a specific interface, and training the UXD specialist to be a UX designer often occurs after training in graphic design or visual design. You will, however, also note that in terms of other kinds of interfaces, there are typically no well defined interface designers outside of the visual experience, and so the design of conversational and zero UI interfaces is often undertaken by user experience specialists.

In the case of UX, we are thinking typically about user experience in the broader context, not specifically design but more from a technical background of computer science and development. Therefore, UXD is not the natural home for somebody who is trained in the broader UX experience. Indeed, when it comes to specifying systems from an interface and interaction perspective a UX specialist should comply with current best practices.

These are typically set out as guidelines for user interface development specifically around expected look and feel across devices; which are typically codified in what is often understood to be a design language, design system or a look and feel document. These typically are given names that are often outside of the operating system on which they run so that their cross-disciplinarity can be highlighted. For instance, the ‘Metro UI’ was created by Microsoft to support the Windows active tile interface but was seen to be broader than this.

User experience is a broad term that encompasses all aspects of a user’s interaction with a product or service. It includes everything from the user’s initial impressions to

¹<https://www.paulrand.design/>

the ease of use and functionality of the product, to the user's overall satisfaction with the experience. On the other hand, user experience design is the process of designing the (visual) user experience. It is a discipline that focuses on creating products that are not only functional but also intuitive, engaging, and enjoyable to use. UX designers use a variety of research techniques and design principles to create products that meet the needs of their users. In essence, UX is the overall perception of a user's experience with a product or service, while UX design is the intentional and strategic process of designing that experience to be as seamless and satisfying as possible for the user.

UXD often employs graphic or visual designers as part of the team. Visual design is an important aspect of UX design, and the role of a visual designer in UXD is to create visually appealing and functional interfaces that enhance the user experience.

While graphic design and visual design are important components of UX design, UXD goes beyond just visual elements. UXD is a multidisciplinary field that encompasses research, design, and evaluation of user experiences. It involves a range of tasks, including user research, information architecture, interaction design, usability testing, and more.

Again, visual design is just one part of the overall UX design process, and it involves the use of typography, colour, imagery, and other visual elements to create a cohesive and aesthetically pleasing user interface. Graphic design, on the other hand, is focused more on the creation of graphics and images to convey information or messages. Visual designers work closely with UX designers and developers to create interface designs that meet the needs of users while also communicating a clear brand message.

While UX specialists may work closely with graphic designers and visual designers to create compelling and visually appealing products, UXD itself is a broader discipline that encompasses a range of design and research activities aimed at creating user-centred products and services.

However, it's important to note that UXD goes beyond just visual design. UX designers often work closely with user researchers, product managers, and developers to create user-centred products and services. They use a variety of research techniques to understand user needs and behaviors, and then design interfaces that meet those needs while also achieving business goals.

In short, while visual design is an important component of UXD, UX involves a multidisciplinary team working together to create products that meet user needs and provide a great user experience.

3.2 Design Systems and Languages

A design system is a collection of guidelines, components, and assets that are used to create and maintain a consistent and cohesive visual and interactive experience across different products, platforms, and channels. It serves as a single source of truth for design and helps ensure a unified and efficient design process.

Design systems are typically developed by design teams within organizations and are used by various stakeholders, including designers, developers, and product managers. They provide a standardized set of rules, principles, and patterns that guide the creation of user interfaces, interactions, and overall brand experience.

Design systems establish a consistent visual language, including colors, typography, iconography, and other graphical elements. This ensures a cohesive and recognizable look and feel across different applications and platforms. They often include a library of reusable user interface (UI) components, such as buttons, forms, navigation menus, and cards. These components are designed and documented with predefined styles, behaviors, and guidelines, allowing designers and developers to easily implement them in their projects.

Systems articulate a set of design principles and guidelines that help maintain a consistent user experience. These principles may cover topics like accessibility, responsiveness, layout, and content hierarchy, providing a framework for decision-making during the design process. And they include comprehensive documentation that outlines the guidelines, rules, and best practices for using the system. This documentation serves as a reference for designers and developers, ensuring proper implementation and reducing inconsistencies.

Systems often emphasize accessibility, aiming to create inclusive experiences that can be accessed by users with diverse abilities. They provide guidelines and recommendations for creating accessible designs and may include accessible UI components and color palettes. Finally systems often align with the organization's brand identity and guidelines. They define the appropriate usage of logos, typography, colors, and other brand assets, ensuring consistency and brand recognition across all touchpoints.

By adopting a design system, organizations can streamline the design and development process, improve collaboration between teams, reduce redundant work, and maintain a cohesive user experience across their products and services. It also allows for scalability and efficiency, as design updates or changes can be applied globally through the system, ensuring a consistent experience across different platforms and products.

A design language – also known as a design language system or design language framework – is more specifically focused on the operational aspects of the overall system. A design language, refers to a comprehensive and consistent set of design principles, patterns, and guidelines that define the visual and interactive characteristics of a product, brand, or organization. It establishes a unified approach to design, ensuring consistency and coherence across various touchpoints.

Design languages are typically developed and maintained by design teams within organizations, and they serve as a reference for designers, developers, and other stakeholders involved in the creation and maintenance of products or services. By establishing a design language, organizations can achieve consistency, improve user experience, enhance brand recognition, and streamline the design and development process across their offerings.

For instance, the original ‘Material Design’ (see [Figure: Material Page Example](#)) system typically found on Android devices and critically across most Google applications has evolved into its own specification to be applied across any platform Google related or not. Further languages include Microsoft Fluent (see [Figure: Fluent Card Example](#)) and indeed, Apple OSX, and IBM all have their own design systems or languages which you should conform to if you are building or specifying interfaces. These kinds of design systems (or languages) evolved from specifications in real-world architecture, and mutated to interface design patterns (see [Figure: Common UI Patterns](#)) via such repos-

itories as the Portland Pattern Repository (famous for the wiki-wiki web developed by Ward Cunningham).

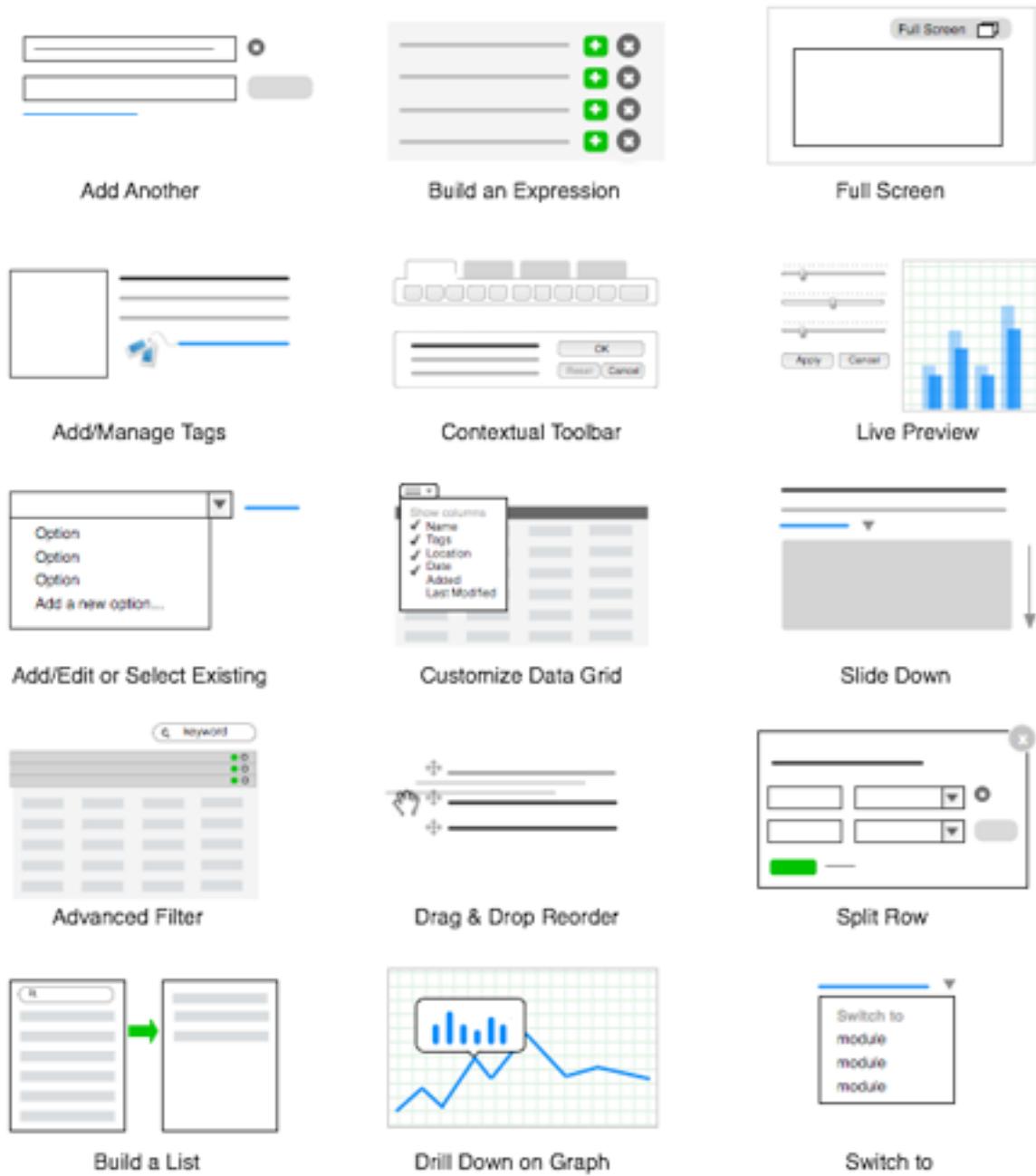


Figure: Common UI Patterns. Common UI patterns refer to the most frequently used and familiar design elements in user interface design. These patterns are used to create interfaces that are easy to use, intuitive, and efficient. – Image Credit: <http://designingwebinterfaces.com>.

Not only technology developers or platform manufacturers create visual languages and guidelines for their applications and platforms. We can see that there are many other organisations who also have their own visual design languages such as Github, Twitter, and Mozilla all have a specification, that their software engineers can use; and this specification is typically contained (or at least modelled) within specific Cascading Style Sheets, directives or visual rendering directives which can be used in whatever

circumstances.

Creation of these languages and specifications should not be attempted by anyone who is not a graphic designer or part of a UXD team and in terms of UX from a technical perspective (you), we are purely the users. So, for instance, we don't create the artificial intelligence or machine learning algorithms, we just use them; we don't build a new ANOVA statistical test, we trust the statisticians to have done their job and we don't reinvent these, and this is the case for using visual design guidance. Conforming also enhances cognitive and perceptual understanding of the user and therefore what to expect is consistent across applications.

3.3 Interaction Design

UXD design and interaction design are closely related but distinct disciplines. Interaction design is a subfield of UX design that specifically focuses on designing how users interact with a product or service.

Interaction design involves designing the flow of user interactions, defining how users navigate through a product or service, and creating the interactions that users have with the interface. This includes designing the placement and behaviour of buttons, menus, forms, and other interactive elements, as well as defining the transitions between screens and the feedback that users receive when they interact with the product.

UX design, on the other hand, encompasses a broader range of activities and disciplines. It includes everything from user research and usability testing to information architecture and visual design. While interaction design is a key component of UX design, it is just one of many elements that contribute to the overall user experience.

In essence, interaction design is a focused subset of UX design that is concerned specifically with designing how users interact with a product or service, while UX design is a broader discipline that encompasses all aspects of a user's experience with a product or service.

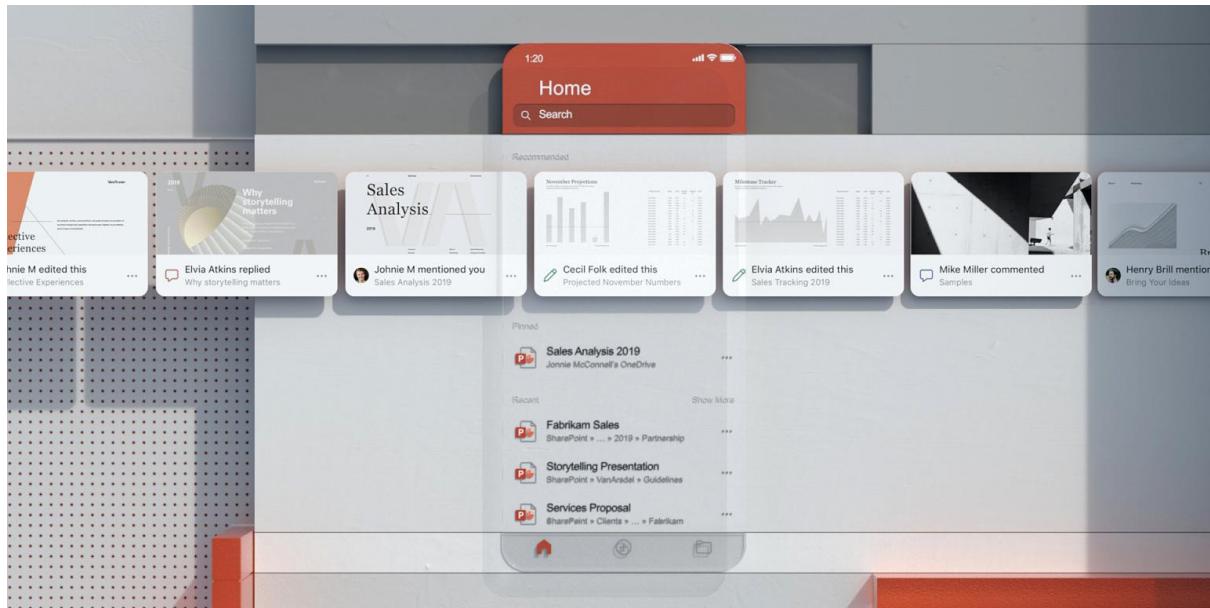


Figure: Fluent Card Example. Fluent Design is a design language and system developed by Microsoft, which provides a set of principles, guidelines, and tools for designing digital interfaces and applications. Fluent Design was introduced in 2017 as a way to create more intuitive and immersive experiences across Microsoft's products and platforms. — Image Credit: Microsoft.

3.4 Zero UI and Conversational Interfaces

It is important to realise that when we talk about UXD we are only talking about visual design. We are not talking about conversational interfaces or Zero UI devices which will still fall under the remit of the UX specialist. Indeed, these kinds of interaction modalities (including those of sonification for instance) have no design language or specifications as we see in visual design and are often highly technical in the way that they are expressed and elaborated, with technology doing the majority of the work, especially in language recognition models for conversational interfaces.

Zero UI interfaces are user interfaces that require little or no user interaction to operate. The goal of zero UI interfaces is to create seamless, intuitive experiences that require minimal effort from the user. They are designed to be intuitive and operate without any visible controls or buttons. Instead of relying on traditional input methods such as touch screens or keyboards, zero UI interfaces leverage technologies such as voice recognition, facial recognition, and gesture recognition to interpret the user's intent and carry out actions. Examples of zero UI interfaces include Amazon's Alexa and Google's Assistant, which use natural language processing to interpret voice commands and smart home devices that can automatically adjust lighting and temperature based on user preferences.

Indeed, Conversational interfaces are a subset of Zero UIs that use natural language to enable communication between humans and machines. These interfaces are designed to emulate human conversation and provide a more intuitive way for users to interact with technology. Conversational interfaces can take many forms, including chatbots, voice assistants, and messaging apps.

Chatbots are computer programs designed to simulate human conversation, typically

through text-based interactions. They are often used in customer service to answer frequently asked questions or provide basic support. Voice assistants, such as Amazon's Alexa, Apple's Siri, and Google Assistant, use speech recognition and natural language processing to enable voice-based interactions between users and devices. Messaging apps, such as Facebook Messenger and WhatsApp, also provide conversational interfaces through which users can interact with chatbots or human agents. Conversational interfaces have become increasingly popular due to their ease of use, accessibility, and ability to provide personalized experiences. They are used in a variety of applications, including customer service, healthcare, finance, and education.

3.5 Summary

In this chapter, I wanted to point out the difference between UX design, visual design and UX as a specialism. We've already covered definitions of UX, and we understood that there are different definitions of what is important in UX, what isn't, and where you want to make sure that you understand that UXD, is really about visual design and UX is broader (you are specialising in UX).

Some texts for UXD as a specialism may emphasize the need for strong visual design skills, but this is not always the case. While

visual design is an important aspect of UX design, many/most of the skills for UX designers prioritize skills such as user research, information architecture, and interaction design over visual design skills. Many UXD texts specifically state that specialists should have experience in user research, user testing, wireframing, prototyping, and other aspects of UX design that are not solely focused on visual design. That said, UX designers need to have a basic understanding of visual design principles and be able to work effectively with visual designers as part of a collaborative team. UX design skills may also include visual design as a desired skill, especially for smaller companies where a UX designer may be expected to handle multiple responsibilities. Overall, however, while visual design is an important aspect of UX design, the emphasis on visual design skills will vary depending on the company, industry, and specific project requirements.

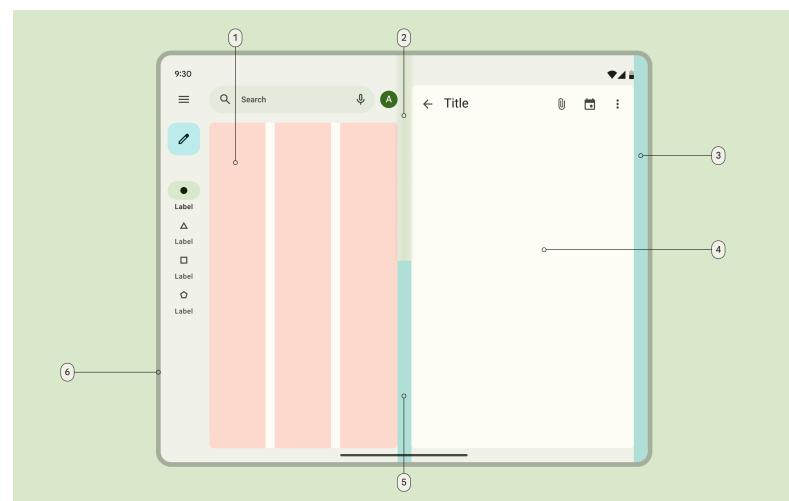


Figure: Material Page Example. 1, Column; 2, Fold; 3, Margin; 4, Pane; 5, Spacer; and, 6, Window. Material Design is a design language and system developed by Google, which provides a set of principles, guidelines, and tools for designing digital interfaces and applications. Material Design was introduced in 2014 and has since become a popular design system used by designers and developers worldwide. — Image Credit: Material.io.

3.5.1 Optional Further Reading

- [Material Design] <https://material.io/>
- [Microsoft Fluent Design] <https://www.microsoft.com/design/fluent/>
- [J Preece, H Sharp & Y Rogers] Interaction Design: Beyond Human-Computer Interaction, Wiley; 6th edition, 2023.
- [A van Boeijen, et al] Delft Design Guide: Design strategies and methods, BIS Publishers; Revised edition 2020.
- [J Johnson] Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines, Morgan Kaufmann; 3rd edition, 2020.
- [J Tidwill, et al] Designing Interfaces, 3e: Patterns for Effective Interaction Design, O'Reilly; 3rd edition, 2020.



Self Assessment Questions

Try these without reference to the text:

1. What is a design language, specification, look & feel, and system; are they different?
2. Describe a conversational interface giving examples.
3. Describe a Zero UI interface, and describe how these might relate to hardware.
4. Who should design interfaces?
5. Pick a design system and describe it.

Part II: Designing the User Experience

4. People are Complicated!

Complexity is your enemy. Any fool can make something complicated. It is hard to make something simple.

– Richard Branson

The key aspect of user experience is its focus on making a system more humane. This means tailoring the system to the user, and not expecting the user to happily conform to the system. We can only achieve this by understanding how humans sense and perceive the outside environment (the system or interface), process and store this knowledge. Making use of it either directly or in the future, and based on past and current knowledge exert control over an external environment (again, in this case, the system or interface).

4.1 UX is Everywhere

User experiences occur in many contexts and over many domains. This variety sometimes makes it difficult to ‘pigeon hole’ UX as one specific thing - as we have discussed - UX is the broad term used to describe the experience of humans with computers both at the interface and system level.

Indeed, the ACM and the IEEE Curriculum ‘Subjects Coverage’ - lists Human Factors as occurring in the following domains:

- AL/Algorithmic Strategies
- NC/Networked Applications
- HC/Foundations
- HC/Building GUI Interfaces
- HC/User Centred Software Evaluation
- HC/User Centred Software Development
- HC/GUI Design
- HC/GUI Programming
- HC/Multimedia And Multimodal Systems
- HC/Collaboration And Communication
- HC/Interaction Design For New Environments
- GV/Virtual Reality
- IM/Hypermedia
- IM/Multimedia Systems
- IS/Fundamental Issues
- NC/Multimedia Technologies
- SE/Software Verification Validation
- CN/Modelling And Simulation
- SP/Social Context

- SP/Analytical Tools
- SP/Professional Ethics
- SP/Philosophical Frameworks

Subject Key: Algorithms and Complexity (AL); Net-Centric Computing (NC); Human-Computer Interaction (HC); Graphics and Visual Computing (GV); Intelligent Systems (IS); Information Management (IM); Software Engineering (SE); Computational Science (CN); Social and Professional Issues (SP).

However, more broadly this is often referred to as human factors that suggest an additional non-computational aspect within the environment; affecting the system or interaction indirectly. This leads on to the discipline of ergonomics, also known as human factors, defined as ... ‘the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimise human well-being and overall system performance.’ At an engineering level, the domain may be known as user experience engineering.

However, in all cases, the key aspect of Human Factors and its application as user experience engineering is the focus on making a system more humane (see [Figure: Microsoft Active Tiles](#)). There is no better understanding of this than that possessed by ‘Jef Raskin’, creator of the Macintosh GUI for Apple Computer; inventor of SwyftWare via the Canon Cat; and author of ‘The Humane Interface’ [[Raskin, 2000](#)]:

“Humans are variously skilled and part of assuring the accessibility of technology consists of seeing that an individual’s skills match up well with the requirements for operating the technology. There are two components to this; training the human to accommodate the needs of the technology and designing the technology to meet the needs of the human. The better we do the latter, the less we need of the former. One of the non-trivial tasks given to a designer of human-machine interfaces is to minimise the need for training.”

“Because computer-based technology is relatively new, we have concentrated primarily on the learnability aspects of interface design, but the efficiency of use once learning has occurred and automaticity achieved has not received its due attention. Also, we have focused largely on the ergonomic problems of users, sometimes not asking if the software is causing ‘Cognitive’ problems. In the area of accessibility, efficiency and Cognetics can be of primary concern.”

“For example, users who must operate a keyboard with a pointer held in their mouths benefit from specially designed keyboards and well-shaped pointers. However well-made the pointer, however, refined the keyboard layout, and, however, comfortable



Figure: Microsoft Active Tiles. Microsoft Active Tiles, More Humane? –Image Credit: Microsoft.

the physical environment we have made for this user, if the software requires more keystrokes than necessary, we are not delivering an optimal interface for that user. When we study interface design, we usually think regarding accommodating higher mental activities, the human capabilities of conscious thought and ratiocination.

Working with these areas of thought bring us to questions of culture and learning and the problems of localising and customising interface designs. These efforts are essential, but it is almost paradoxical that most interface designs fail first to assure that the interfaces are compatible with the universal traits of the human nervous system, in particular, those traits that are sub-cortical and that we share with other animals. These characteristics are independent of culture and learning, and often are unaffected by disabilities.

Most interfaces, whether designed to accommodate accessibility issues or not, fail to satisfy the more general and lower-level needs of the human nervous system. In the future, designers should make sure that an interface satisfies the universal properties of the human brain as a first step to assuring usability at cognitive levels.”

In the real world, this means that the UX specialist can be employed in many different jobs and many different guises; being involved in the collection of data to form an understanding of how the system or interface should work. Regardless of the focus of the individual project, the UX specialist is concerned primarily with people and the way in which they interact with the computational system.



Figure: PET studies of glucose metabolism.. PET studies of glucose metabolism to map human brain's response in performing different tasks. —Image Credit: M. E. Phelps, PNAS, 2000, 97, 9226.

4.2 People & Computers

Understanding how humans take in information, understand and learn from this information, and use it to guide their control of the outside world, is key to understanding their experiences [Weinschenk, 2011] (see [PET scans in Figure: PET studies of glucose metabolism](#)). Here we are not concerned directly with the anatomical, physiological, or psychological aspects of these processes (there are many in-depth treatise that address these areas), but instead, look at them at the point where user meets software or device. In this case we can think, simplistically, that humans sense and perceive the outside environment (the system or interface), process and store this knowledge making use of it either directly or in the future. Moreover, based on both past and current knowledge exert control over an external environment (again, in this case, the system or interface) [Bear et al., 2007].

4.2.1 Perceiving Sensory Information

Receiving sensory information can occur along some different channels. These channels relate to the acts of: seeing (visual channel), hearing (auditory channel), smelling (olfactory channel), and touching (somatic/haptic channel). This means that each of these channels could, in effect, be used to transmit information regarding the state of the system or interface. It is important then, to understand a few basic principles of these possible pathways for the transmission of information from the computer to the human.



Figure: Basketball.. Basketball Awareness Test by Daniel Simons and Christopher Chabris. —Image Credit: 1999 Daniel J. Simons; <http://www.theinvisiblegorilla.com>.

However, before we begin, remember back to our previous discussion – and concerning ‘[Figure: Basketball](#)’ – examining Daniel Simons and Christopher Chabris’ experiments which required you to count passes of a Basketball. Well, this phenomenon is just one example of the complexities involved in understanding users, their perception, and their attention. The phenomenon exhibited is known as ‘Inattention blindness’, ‘attention blindness’, or ‘perception blindness’ and relates to our expectations, perception, and locus of attention. Simply, inattention blindness describes our ability to notice something that is in plain view and has been seen. This normally occurs when we are not expecting the stimulus to occur – why would a Gorilla be moonwalking in a basketball match? If we do not think it should be there, our brains compensate for the perceptual input by not cognitively registering that stimuli. However, there are plenty of other tests that also show this phenomenon. Moreover, there are some different explanations for the why it occurs. As you’ve just read, I prefer the ‘Expectation’ ex-

planation, but others suggest: conspicuity, stimuli may be inconspicuous and therefore not seen as important. Mental workload, we may be too busy focused on another task to notice a stimuli, or capacity this is really the locus of attention explanation; as [we shall see later in on](#). In reality, I expect that there is some combination of each explanation at play, but the point is that the phenomenon itself is not what we would expect, and we don't exactly know why or how it occurs. Keep this level of uncertainty in mind as you read more and as you start you work in the UX domain.

4.2.1.1 Visual Interaction

Visual interaction design is determined by the arrangement of elements or details on the interface that can either facilitate or impede a user through the available resources. This is because the amount of information within the interface that reaches our eyes is far greater than our brain can process, and it is for this reason that the visual channel and, therefore, visual attention is key to good design. Selective visual attention is a complex action composed of conscious and subconscious processes in the brain that are used to find and focus on relevant information quickly and efficiently. There are two general visual attention processes, *bottom-up* and *top-down*, which determine where humans next locate their attention. Bottom-up models of visual attention suggest that low-level salient features, such as contrast, size, shape, colour, and brightness correlate well with visual interest. For example, a red apple (a source of nutrition) is more visually salient, and, therefore, attractive, than the green leaves surrounding it. Top-down models, on the other hand, explain visual search driven by semantics, or knowledge about the environment: when asked to describe the emotion of a person in a picture, for instance, people will automatically look to the person's face.

Both forms of visual attention inevitably play a part in helping people to orientate, navigate and understand the interface. Bottom-up processing allows people to quickly detect items, such as bold text and images, which help to explain how the interface is organised. It also helps people to group the information into 'sections', such as blocks of text, headings, and menus. Top-down processing enables people to interpret the information using prior knowledge and heuristics. For example, people may look for menus at the top and sides of the interface, and main interface components in the middle.

It can be seen from [eye tracking studies](#) that users focus attention sequentially on different parts of the interface and that computational models have been successfully employed in computer graphics to segment images into regions that the user is most likely to focus upon. These models are based upon a knowledge of human visual behaviour and an understanding of the image in question. Indeed, studies exist which record user's eye movements during specific interactive tasks, to find out those features within and interface design that were visited, in which order and where 'gaze hotspots' were found. In these data, we can see an association between the interface components and eye-gaze, but not one as simple as 'the user looked at the most visually obvious interface features. Indeed, sometimes large-bold-text next to an attention grabbing feature, such as an image, was fixated upon. However, we can deduce that, for instance, the information in some text may not itself draw a user's attention but ideally has some feature nearby which do. This idea is supported by other studies that try to create interface design metrics that predict whether an interface is visually complex or not. These studies relate to interface design with complexity explaining that the way an

interaction is perceived depends on the way the interface itself is designed and what components are used.

Through an understanding of the visual channel we can see that there are levels of granularity associated with visual design. This granularity allows us to understand a large visual rendering by segmentation into smaller more manageable pieces or components. Our attention moves between these components based on how our visual attention relates to them, and it is this knowledge of how the visual channel works, captured as design best practice, which allows designers to build interfaces that users find interesting and easy to access. The visual narrative the designer builds for the observer is implicitly created by the visual appearance (and, therefore, attraction) of each visual component. It is the observers focus, or as we shall see, [the locus of attention](#), which enables this visual narrative to be told.

4.2.1.2 The Auditory Channel

The Auditory channel is the second most used channel for information input, and as highly interrelated with the visual channel, however, auditory input has some often overlooked advantages. For instance, reactions to auditory stimuli have been shown to be faster, in some cases, than reactions to visual stimuli. Secondly, the use of auditory information can go some way towards reducing the amount of visual information presented on screen. In some regard, this reduces possible information overload from both interface components and aspects of the interaction currently being performed. The reduction in visual requirements means that attention can be freed to stimuli that are best handled visually, and means that auditory stimulation can be tailored to occur within the part of the interaction to which it is best suited. Indeed, the auditory channel is, often, under-used in relation to standard interface components. In some cases, this is due to the need for an absence of noise pollution within the environment. We can see that consistent sound or intermittent sound can often be frustrating and distracting to computer users within the same general environment; this is not the case with visual stimuli that can be specifically targeted to the principal user. Finally, and in some cases most importantly, sound can move the users attention, and focus it to a specific spatial location. Due to the nature of the human auditory processing system, studies have found that using different frequencies similar to white noise are most effective in attracting attention. This is because human speech and communication occur using multiple frequencies whereas single frequencies with transformations such as sirens or audible tones do not jar the senses but are more difficult to locate spatially.

Obviously, sound can be used for many different and complementary aspects of the user interface. However, the HCI specialist must consider the nature of the sound and its purpose. In addition, interfaces that rely only on sound, with no additional visual cues, may mean the interface becomes inflexible in environments where the visual display is either obscured or not present at all. Sound can be used to transmit speech and non-speech. Speech would normally be via text to speech synthesis systems. In this way, the spoken word can be flexible, and the language of the text can be changed to facilitate internationalisation. One of the more interesting aspects of non-speech auditory input is that of auditory icons and '[Earcons](#)'. Many systems currently have some kind of auditory icon, for instance, the deletion of files from a specific directory is often accompanied by sound used to convey the deletion, such as paper being scrunched. Earcons are slightly more complicated as they involve the transmission

of non-verbal audio messages to provide information to the user regarding some type of computer object, operation, or interaction. Earcons, use a more traditional musical approach than auditory icons and are often constructed from individual short rhythmic sequences that are combined in different ways. In this case, the auditory component must be learned as there is no intuitive link between the Earcon and what it represents.

4.2.1.3 Somatic

The term ‘Somatic’ covers all types of physical contact experienced in an environment, whether it be feeling the texture of a surface or the impact of a force. Haptics describe how the user experiences force or cutaneous feedback when they touch objects or other users. Haptic interaction can serve as both a means of providing input and receiving output and has been defined as ‘The sensibility of the individual to the world adjacent to his body by use of his body’. Haptics are driven by tactile stimuli created by a diverse sensory system comprising receptors covering the skin, and processing centres in the brain, to produce the sensory modalities such as touch and temperature. When a sensory neuron is triggered by a specific stimulus, a neuron passes to an area in the brain that allows the processed stimulus to be felt at the correct location. It can, therefore, be seen, that the use of the haptic channel for both control and feedback can be important especially as an aid to other sensory input or output. Indeed, haptics and tactility have the advantage of making interaction seem realer, which accounts for the high use of haptic devices in virtual and immersive environments.

4.2.1.4 The Olfactory System

The Olfactory system enables us to perceive smells. It may, therefore, come as a surprise to many studying UX that there has been some work investigating the use smell as a form of sensory input directly related to interaction (notably from **[Brewster et al., 2006]**). While this is a much under-investigated area, certain kinds of interface can use smell to convey an extra supporting component to the general, and better understood, stimuli of sound and vision. One of the major benefits of smell is that it has a close link with memory, in this case, smell can be used to assist the user in finding locations that they have already visited, or indeed recognise a location they have already been to within the interactive environment. Indeed, it seems that smell and taste **[Narumi et al., 2011]** is particularly effective when associated with image recognition, and can exist in the form an abstract smell, or a smell that represents a particular cue. It is, however, unlikely, as a UX'er, that smell will be used in any large way in the systems you are designing, although it may be useful to keep smell in mind if you are having particular problems with users forgetting aspects of previously learnt interaction.

4.2.2 Thinking and Learning

The next part of the cycle is how we process, retain, and reuse the information that is being transmitted via the sensory channels **[Ashcraft, 1998]**. In this case, we can see that there are aspects of attention, memory and learning, support the exploration and navigation of the interface and the components within that interface. In reality,

these aspects interrelate and affect each other in more complex ways than we, as yet, understand.

Attention or the locus of attention is simply the place, the site, or the area at which our attention is currently focused. This may be in the domain of unconscious or conscious thought, it may be in the auditory domain when listening to speech and conversation, or it may be in the visual domain while studying a painting. However, this single area of attention is the key to understanding how our comprehension is serialised and the effect that interface components and their granularity have when we perceive sensory concepts within a resource; creating a narrative, either consciously or unconsciously.

Cognitive psychologists already understand that the execution of simultaneous tasks is difficult, especially if these are conscious tasks. Repeating tasks at a high-frequency mean that they become automatic, and this automaticity means that they move from a conscious action to an unconscious action. However, the locus of attention for that person is not on the automatic task but the task that is conscious and ‘running’ in parallel. Despite scattered results suggesting (for example) that, under some circumstances, humans can have two simultaneous loci of spatial attention, the actual measured task performance of human beings on computers suggests that, in the context of interaction, there is one single locus of attention. Therefore, we can understand that for our purposes there is only one single locus of attention, and it is this attention that moves over interactive resources based on the presentation of that resource, the task at hand, and the experience of the user.

While the locus of attention can be applied to conscious and unconscious tasks, it is directed by perception fed from the senses. This is important for interface cognition and the components of the interface under observation. As we shall [touch-on later](#), the locus of attention can be expressed as fixation data from eye tracking work and this locus of attention is serialisable. The main problem with this view is that the locus of attention is specific to the user, the task at hand, as well as the visual resource. In this case, it is difficult to predict how the locus of attention will manifest for every user and every task, and, therefore, the order in which the interface components will be serialised. We can see how the locus of attention is implicitly dealt with by current technologies and techniques and, by comparison to gaze data and fixation times, better understand how the serialisation of visual components can be based on the ‘mean time to fixation’.

It would be comforting to believe that we had a well-defined understanding of how memory and learning work in practice. While we do have some idea, most of our knowledge regarding memory and learning and how these fit into the general neurology of the brain is currently lacking. This said we do have some understanding of the kinds of aspects of memory and learning that affect human-computer interaction (see [Figure: Participants with Minimal Online Experience](#)). In general, sensory memories move via attention into short-term or working memory, the more we revisit the short-term memory, the more we are likely to be able to move this working memory into long-term memory.

In reality, areas of the brain are associated with the sensory inputs: iconic, for visual stimuli; echoic, for auditory stimuli; and haptic for tactile stimuli. However, visual stimuli prove to be the most supported of the senses, from a neurological perspective with over fifty areas of the brain devoted to processing vision, and only one devoted

to hearing. In general, then, the UX specialist should understand that we learn and commit to memory by repetition. This repetition may be via rehearsal, it may be via foreknowledge, or it may be via an understanding of something that we have done before, in all cases however repetition is key. Indeed, this repetition is closely linked to the ability to learn and discover an interface, in the first person, as opposed to second hand via the help system. This makes the ability for users to self-explore the interface and interactive aspects of the system, paramount. Many users whom I talk with often state that they explore incrementally and don't refer much to the manuals. With this in mind, interfaces and systems should be designed to be as familiar as possible, dovetailing into the processes of the specific user as opposed to some generalised concept of the person, and should support aspects of repeatability and familiarity, specifically for this purpose.

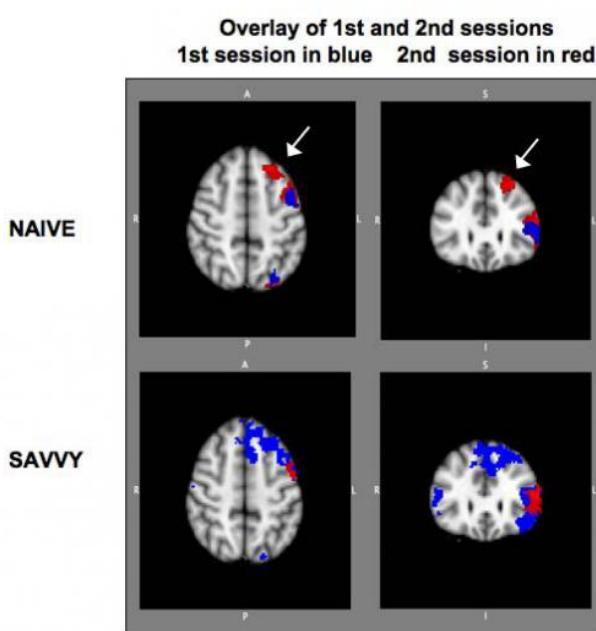


Figure: Participants with Minimal Online Experience. Participants with minimal online experience displayed brain activation patterns very similar to those seen in the group of savvy Internet users - after just a brief period of time. —Image Credit: Faith Brynie, Psychology Today (Brain Sense).

thought of as knowledge of the basic spatial relationships between components within the interface. It is used as a term to suggest a comprehension of an interactive environment or components that relate to exploration within the environment. How a person is oriented is crucial to successful interaction. Information about position, direction, desired location, and route are all bound up with the concept of orientation. Navigation, in contrast, suggests an ability of movement within the local environment. This navigation can be either by the use of pre-planning using help or previous knowledge or by navigating 'on-the-fly' and as such a knowledge of immediate components and barriers are required. Navigation and exploration are key within the UX domain because they enable the user to easily understand, and interact with, the interface. However, in some ways, more importantly, they enable the interface to be learnt by a self-teaching process. Facilitating exploration through the use of an easily navigable interface means that users will intuitively understand the actions that are required of them and that the

Exploration can be thought of like the whole experience of moving from one interface component to another, regardless of whether the destination is known at the start of navigation or if the traversal is initially aimless. Movement and exploration of the interface also involve orientation, interface design, purpose, and mobility. The latter defined as the ability to move freely, easily and confidently around the interface. In this context, a successful exploration, of the interactive elements, is one in which the desired location is easily reached or discovered. Conventionally, exploration can be separated into two aspects: Those of Navigation and Orientation. Orientation can be

need for help documentation is reduced.

4.2.3 Explicit and Implicit Communication

Information can be transmitted in a number of different ways, and these ways may be either implicit (covert) or explicit (overt). In this case let us refer to information transmission as communication; in that information is communicated to the user and the user requirements are then communicated back to the computer via the [input and control mechanisms](#). Explicit communication is often well understood and centres on the visual, or auditory, transmission of both text and images (or sounds) for consumption by the user. Implicit communications are, however, a little more difficult to define. In this case, I refer to those aspects of the visual or auditory user experience that are in some ways intangible. By which I mean aspects such as aesthetic or emotional responses [\[Pelachaud, 2012\]](#) to aspects of the communication.

Explicit Communication. Communication and complexity are closely related whereby complexity directly affects the ease of communication between the user and the interface or system. Communication mainly occurs via the visual layout of the interface and the textual labels that must be read by the user to understand the interactive communication with them. People read text by using jerky eye movements (called 'Saccades') which then stop and fixate on a keyword for around 250 milliseconds. These fixations vary and last longer for more complex text, and are focussed on forward fixations with regressive (backward) fixations only occurring 10-15 percent of the time when reading becomes more difficult. People, reading at speed by scanning for just appropriate information, tend to fixate less often and for a shorter time. However, they can only remember the gist of the information they have read; and are not able to give a comprehensive discourse on the information encountered (see [Figure: Visualising Words](#)). This means that comprehensive descriptions of interfaces for users, when quickly scanning an interactive feature, are not used in the decision-making process of the user. Indeed, cognitive overload is a critical problem when navigating large interactive resources. This overload is increased if the interaction is non-linear and may switch context unexpectedly. Preview through summaries is key to improving the cognition of users within complex interfaces, but complex prompts can also overload the reader with extraneous information.

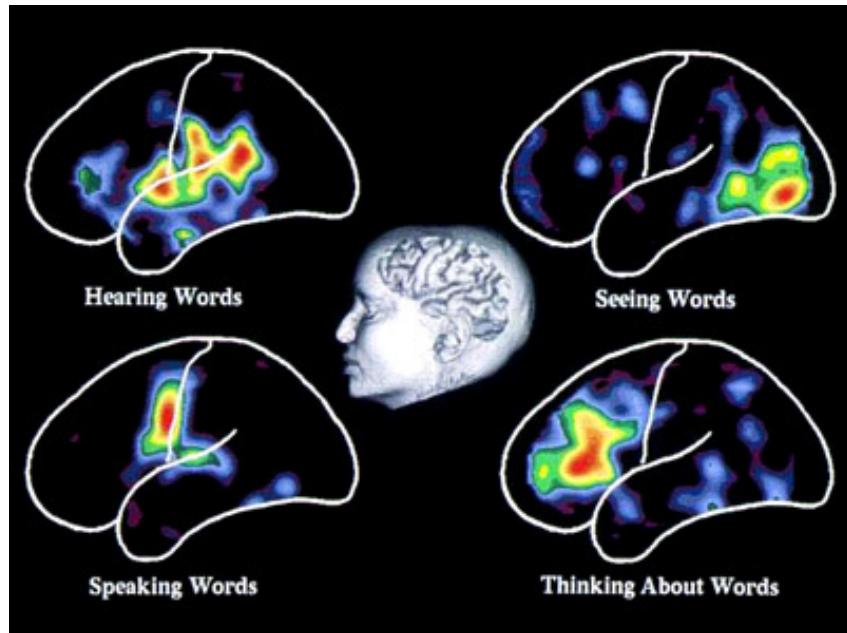


Figure: Visualising Words. Visualising Words. —Image Credit: Marcus E. Raichle, Department of Radiology, Washington University School of Medicine, St. Louis, Missouri.

In addition to the problems of providing too much textual information the complexities of that information must also be assessed. The Flesch/Flesch–Kincaid Readability Tests are designed to indicate comprehension difficulty when reading a passage of contemporary academic English. Invented by Flesch and enhanced by Kincaid, the Flesch–Kincaid Grade Level Test¹ analyses and rates text on a U.S. grade-school level based on the average number of syllables per word and words per sentence. For example, a score of 8.0 means that an eighth grader would understand the text. Keeping this in mind, the HCI specialist should try to make the interface labels and prompts as understandable as possible. However this may also mean the use of jargon is acceptable in situations where a specialist tool is required, or when a tool is being developed for a specialist audience.

Implicit Communication. Aesthetics is commonly defined as the appreciation of the beautiful or pleasing, but this terminology is still controversial. Visual aesthetics is ‘the science of how things are known via the senses’ which refers to user perception and cognition. Specifically, the phrase ‘aesthetically pleasing’ interfaces are commonly used to describe interfaces that are perceived by users as clean, clear, organised, beautiful and interesting. These terms are just a sample of the many different terms used to describe aesthetics, which are commonly used in interaction design. Human-computer interaction work mostly emphasises performance criteria, such as time to learn, error rate and time to complete a task, and pays less attention to aesthetics. However, UX work has tried to expand this narrow ([‘Reductionist’](#)) view of experience, by trying to understand how aesthetics and emotion can affect a viewer’s perception - but the relationship between the aesthetic presentation of an interface and the user’s interaction is still not well understood.

The latest scientific findings indicate that emotions play an essential role in decision making, perception, learning, and more—that is, they influence the very mechanisms

¹The Flesch-Kincaid Grade Level formula is used by the US Government Department of Defence as a standard test.

of rational thinking. Not only too much, but too little emotion can impair decision making [Picard, 1997]. This is now referred to as ‘[Affective Computing](#)’ in which biometric instruments such as Galvanic Skin Response (GSR), Gaze Analysis and Heart Rate Monitoring are used to determine a user’s emotional state via their physiological changes. Further, affective computing also seeks ways to change those states or exhibit control over technology, based upon them².

4.3 Input and Control

The final piece of the puzzle, when discussing UX, is the ability to enter information for the computer to work with. In general, this is usually accomplished by use of the GUI working in combination with the keyboard and the mouse. However, this is not the entire story, and indeed there are many kinds of entry devices which can be used for information input, selection, and target acquisition [Brand, 1988]. In some cases, specialist devices, such as head operated mice have been used, in others gaze and blink detection, however, in most cases, the mouse and keyboard will be the de-facto combination. However, as a UX engineer you should also be aware of the different types of devices that are available and their relationship to each other. In this way, you will be able to make accurate and suitable choices when specifying non-standard systems.

Text Entry via the keyboard is the predominant means of data input for standard computer systems. These keyboards range from the standard QWERTY keyboard, which is inefficient and encourages fatigue, through the Dvorak Simplified Keyboard, ergonomically designed using character frequency, to the Maltron keyboard designed to alleviate carpal tunnel syndrome and repetitive strain injury. There are however other forms of text entry. These range from the T9 keyboard found on most mobile phones and small devices with only nine keys. Through to soft keyboards, found as part of touch screen systems, or as assistive technologies for users with motor impairment, where a physical keyboard cannot be controlled. You may encounter many such methods of text entry, or indeed be required to design some yourself. If the latter is the case, you should make sure that your designs are base on an understanding of the physiological and cognitive systems at work.

The Written Word, cursive script, was seen for a long period as the most natural way of entering text into a computer system; because it relied on the use of a skill already learnt. The main bar to text entry via the written word was that handwriting recognition was computationally intensive and, therefore, underdeveloped. One of the first handwriting recognition systems to gain popularity was that employed by the Apple Newton, which could understand written English characters, however, accuracy was a problem. Alternatives to handwriting recognition arrived in the form of pseudo-handwriting, such as the Graffiti system that was essentially a single stroke shorthand handwriting system. Current increasing computational power and complexity of the algorithms used to understand handwriting have led to a resurgence in its use in the form of systems such as Inkwell. Indeed, with the advent of systems such as the Logitech io2 Digital Writing System, handwriting recognition has become a computer-paper hybrid.

Pointing Devices, for drawing and target acquisition is handled, in most cases, by

²We'll talk more about affective computing later.

the ubiquitous mouse. Indeed, this is the most effective way of acquiring a target; with the exclusion of the touch-screen. There are however other systems that may be more suitable in certain environments. These may include the conventional trackpad, the trackball, or the joystick. Indeed, variations on the joystick have been created for mobile devices that behave similarly to their desktop counterparts but have a more limited mobility. Likewise, the trackball also has a miniaturised version present on the Blackberry range of mobile PDAs, but its adoption into widespread mobile use has been limited. Finally, the touch screen is seeing a resurgence in the tablet computing sector where the computational device is conventionally held close enough for easy touch screen activation, either by the operators finger or a stylus.

Haptic Interaction is not widely used beyond the realm of immersive or collaborative environments. In order to interact haptically with a virtual world, the user holds the ‘end-effector’ of the device. Currently, haptic devices support only point contact, so touching a virtual object with the end-effector is like touching it using a pen or stick. As the user moves the end-effector around, its coordinates are sent to the virtual environment simulator (for example see [Figure: Phantom Desktop Haptic Device](#)).

If the end-effector collides with a haptically-enabled object, the device will render forces to simulate the object’s surface. Impedance control means that a force is applied only when required: if the end-effector penetrates the surface of an object, for example, forces will be generated to push the end-effector out, to create the impression that it is touching the surface of the object. In admittance control, the displacement of the end-effector is calculated according to the virtual medium through which it is moving, and the forces exerted on it by the user. Moving through air requires a very high control gain from force input to displacement output, whereas touching a hard surface requires a very low control gain. Admittance control requires far more intensive processing than impedance control but makes it much easier to simulate surfaces with low friction, or objects with mass.

Speech Input is increasing in popularity due to its ability to handle naturally spoken phrases with an acceptable level of accuracy. There are however only a very few speech recognition engines that have gained enough maturity to be effectively used. The most popular of these is the Dragon speech engine, used in Dragon NaturallySpeaking and Mac Dictate software applications. Speech input can be very effective when used in constrained environments or devices, or when other forms of input are slow or inappropriate. For instance text input by an unskilled typist can be time-consuming whereas general dictation can be reasonably fast; and with the introduction of applicators such as Apple’s ‘SIRI’, is become increasingly widespread.

Touch Interfaces have become more prevalent in recent times. These interfaces originally began with single touch graphic pads (such as those built commercially by



Figure: Phantom Desktop Haptic Device. Phantom Desktop Haptic Device. –Image Credit: Phantom.

Wacom) as well as standard touch-screens that removed the need for a conventional pointing device. The touch interface has become progressively more accepted especially for handheld and mobile devices where the distance from the user to the screen is much smaller than for desktop systems. Further, touch-pads that mimic a standard pointing device, but within a smaller footprint, became commonplace in laptop computer systems; but up until the late 2008's all these systems were only substitutes for a standard mouse pointing device. The real advance came with the introduction of gestural touch and multi-touch systems that enabled the control of mobile devices to become much more intuitive/familiar; with the use of swipes and stretches of the document or application within the viewport. These multi-touch gestural systems have now become standard in most smartphones and tablet computing devices, and their use normalised by providing easy APIs within operating systems; such as iOS. These gestural touch systems are markedly different from gesture recognition whereby a device is moved within a three-dimensional space. In reality, touchscreen gestures are about the two-dimensional movement of the user's fingers, whereas gesture recognition is about the position and orientation of a device within three-dimensional space (often using accelerometers, optical sensing, gyroscopic systems, and/or Hall effect cells).

Gesture Recognition was originally seen as an area mostly for research and academic investigation. However, two products have changed this view in recent times. The first of these is the popular games console, the Nintendo Wii, which uses a game controller as a handheld pointing device and detects movement in three dimensions. This movement detection enables the system to understand certain gestures, thereby making game-play more interactive and intuitive. Because it can recognise naturalistic, familiar, real world gestures and motions it has become popular with people who would not normally consider themselves to be games console users. The second major commercial success of gesture recognition is the Apple iOS. The iOS³ can also detect movement in three-dimensions and uses similar recognition as the Nintendo games console to interact with the user. The main difference is that the iOS uses gesture and movement recognition to more effectively support general computational and communication-based tasks.

There are a number of specialist input devices that have not gained general popularity but nevertheless may enable the UX specialist to overcome problems not solvable using other input devices. These include such technologies as the Head Operated Mouse (see [Figure: Head Operated Mouse](#)), which enables users without full body movement to control a screen pointer, and can be coupled with blink detection to enable target acquisition. Simple binary switches are also used for selecting text entry from scanning soft keypads using deliberate key-presses or, in the case of systems such as the Hands-free Mouse COntrol System (HaMCoS), via the monitoring of any muscle contraction. Gaze detection is also popular in some cases, certainly concerning in-flight control of military aircraft. Indeed, force feedback has also been used as part of military control systems but has taken on a wider use when some haptic awareness is required; such as remote surgical procedures. In this case, force feedback mice and force feedback joysticks have been utilised to make the remote sensory experience more real to the user. Light pens have given rise to pen-computing but have now themselves become less used especially with the advent of the touch-screen. Finally, immersive systems, or parts of those immersive systems, have enjoyed some limited popularity. These

³...and many other tablet computers.

systems originally included body suits, immersive helmets, and gloves. However, the most popular aspect used singularly has been the interactive glove, which provides tactile feedback coupled with pointing and target acquisition.

4.4 Summary



Figure: Head Operated Mouse. Head Operated Mouse. —Image Credit: SmartNAV 4.

Understanding how we use our senses: seeing (visual channel), hearing (auditory channel), smelling (olfactory channel), and touching (haptic channel) enables us to understand how to communicate information regarding the state of the system via the interface. Knowledge of the natural processes of the mind: attention, memory and learning, exploration and navigation, affective communication and complexity, and aesthetics, enables an understanding of how the information we transmit can better fit the users mental processes. Finally, understanding how input and control are enacted can enable us to choose from the many kinds of entry

devices which can be used for information input, selection, and for target acquisition. However, we must remember that there are many different kinds of people, and these people have many different types of sensory or physical requirements which must be taken into account at all stages of the construction process.

4.4.1 Optional Further Reading / Resources

- [M. H. Ashcraft.] Fundamentals of cognition. Longman, New York, 1998.
- [S. Brand.] The Media Lab: inventing the future at MIT. Penguin Books, New York, N.Y., U.S.A., 1988.
- [A. Huberman] Andrew Huberman. YouTube. (2013, April 21). Retrieved August 10, 2022, from <https://www.youtube.com/channel/UC2D2CMWXMOVWx7giW1n3Llg>
- [C. Pelachaud.] Emotion-oriented systems. ISTE, London, 2012.
- [R. W. Picard.] Affective computing. MIT Press, Cambridge, Mass., 1997.
- [J. Raskin.] The humane interface: new directions for designing interactive systems. Addison-Wesley, Reading, Mass., 2000.
- [S. Weinschenk.] 100 things every designer needs to know about people. Voices that matter. New Riders, Berkeley, CA, 2011.



Self Assessment Questions

Try these without reference to the text:

1. Pick one input modality and describe it.
2. What are the four main sensory channels, and briefly explain them?
3. How does UX relate to previous Human Factors work?
4. What does Affective Computing mean?
5. Describe the differences between Gestural Interfaces and Touch Gestures.

5. Practical Ethics

A man without ethics is a wild beast loosed upon this world.

– Albert Camus

The primary purpose of evaluations involving human subjects is to improve our knowledge and understanding of the human environment including medical, clinical, social, and technological aspects. Even the best proven examples, theories, principles, and statements of relationships must continuously be challenged through evaluation for their effectiveness, efficiency, affectiveness, engagement and quality. In current evaluation and experimental practice some procedures, will by nature, involve a number of risks and burdens. UX specialists should be aware of the ethical, legal and regulatory requirements for evaluation on human subjects in their own countries as well as applicable international requirements. No national ethical, legal or regulatory requirement should be allowed to reduce or eliminate any of the protections for human participants which you as a professional UX practitioner are bound to follow, as part of your personal code of conduct.

Evaluation in the human science, of which UX is one example, is subject to ethical standards that promote respect for all human beings and protect their health and rights. Some evaluation populations are vulnerable and need special protection. The particular needs of the economically and medically disadvantaged must be recognised. Special attention is also required for those who cannot give, or refuse, consent for themselves, for those who may be subject to giving consent under duress, for those who will not benefit personally from the evaluation, and for those for whom the evaluation is combined with care.

Unethical human experimentation is human experimentation that violates the principles of medical ethics. Such practices have included denying patients the right to informed consent, using pseudoscientific frameworks such as race science, and torturing people under the guise of research.

However, the question for many UX specialists is, why do these standards, codes of conduct or practice, and duties of care exist. What is their point, why where they formulated, and what is the intended effect of their use?

5.1 The Need for Ethical Approaches

Rigourously structured hierarchical societies, as were common up until the 1960s, and indeed some may say are still common today, are often not noted for their inclusion or respect for all citizens. Indeed, one sided power dynamics have been rife in most societies. This power differential led to increasing abuse of those with less power by those with more power. This was no different within the scientific, medical, or empirical communities. Indeed, as the main proponents of scientific evaluation were, in most cases, economically established, educated, and therefore from the upper levels of

society, abuses of human ‘subjects’¹ became rife. Even when this abuse was conducted by proxy and there was no direct intervention of the scientist in the evaluation.

Indeed, these kinds of societies are often driven by persons who are in a position of power (at the upper levels) imposing their will on communities of people (at the lower levels). Further, these levels are often defined by both economic structure or some kind of inherited status or honour, which often means that citizens at the upper level have no concept of what it is to be a member of one of the lower levels. While it would be comforting to think that this only occurred in old European societies it would be ridiculous to suggest that this kind of stratified society did not occur in just about every civilisation, even if their formulation was based on a very specific and unique set of societal values.

In most cases, academic achievement provided a route into the upper levels of society, but also limited entry into those levels; as education was not widely available to an economically deprived population characteristic of the lower levels. In this case, societies worked by having a large population employed in menial tasks, moving to unskilled labour, to more skilled labour, through craftsmanship and professional levels, into the gentry, aristocracy, and higher levels of the demographic. In most cases, the lack of a meritocracy, and the absence of any understanding of the ethical principles at work, meant that the upper levels of society often regard the lower levels as a resource to be used.



Figure: Old Skool. Old Skool Ethnography had little contact with the people and cultures being studied, and upheld unethical practices that yielded no real scientific gains –Image Credit: Notes from the Ethnoground - <https://ethnoground.blogspot.com/2013/>.

Many cases of abuse have been recorded in experimentation and investigations from the late 17th and early 18th centuries onwards. These abuses took place under the guise of furthering our knowledge of humankind and were often perpetrated against dominated populations. There are many recorded abuses of Australian aboriginal peoples, African peoples, and indigenous south American peoples, in the name of the advancement of our understanding of the world and the peoples within it. These abuses were not purely limited to the European colonies, but also occurred within the lower classes of European society; in some cases involving the murder (see Burke and Hare) and sale of subjects to complicit surgeons for anatomical investigation. Experimenters contented themselves with the rationale of John Stuart Mill, which suggests that experimental evaluation should do more good than harm, or that more people should benefit than suffer. However, there was no concept of justice in this initial rationale as the people who usually ended up suffering were the underprivileged and those who benefited were often the privileged minority.

¹I say subjects here because this is the old name for participants – implicitly indicating a power dynamic with the scientist or experimenter.

The ethical rules that grew up in the late 1950s and 60s came about due to the widespread abuse of human subjects in 1940s Germany:



"Human Experimentation was a series of controversial medical experiments on large numbers of prisoners by the German Nazi regime in its concentration camps during World War II. Prisoners were coerced into participating: they did not willingly volunteer and there was never informed consent. Typically, the experiments resulted in death, disfigurement or permanent disability."

Once these abuses came to light, including the inhumane treatment meted out to adults and children alike, organisations such as the American Psychological Association and the American Medical Association began to examine their own evaluation practices. This investigation did not prove to be a pleasant experience, and while there were no abuses on the scale of the German Human Experiments it was found that subjects were still being badly treated even as late as the 1960s.

This growing criticism led to the need for ethical policies which would enable evaluation to be undertaken (there is clearly a benefit to society in scientific, medical, and clinical evaluation) but with a certain set of principles which ensured the well-being of the subjects. Obviously, a comprehensive ethical structure, to include all possible cases, is difficult to formulate at the first attempt. It is for this reason that there have been continual iterative revisions to many evaluation ethics guidelines. However, revisions have slowed and the key principles that have now become firm, should enable the UX specialist to design ethical studies taking into account the inherent human rights of all who participate in them.

5.2 Subjects or Participants?

You may be wondering why this overview matters to you as an UX specialist. In reality, ethics is a form of ‘right thinking’ which encourages us to moderate our implicit power over the people, and concepts, we wish to understand or investigate. You’ll notice that in all but the most recent ethical evaluation standards the term ‘subject’ is used to denote the human taking part in the experimental investigation. Indeed, I have deliberately used the term subject in all of my discussions up to this point, to make a very implicit way of thinking, explicit. The Oxford English Dictionary² (OED) defines the term ‘subject’ variously as:



"(i) The subject-matter of an art or science. (ii) A person (rarely, a thing) that is in the control or under the dominion of another."

Of course the OED also describes a subject in a less pejorative manner as a person towards whom or which action, thought, or feeling is directed, or a person upon whom an experiment is made. Yet they also define the term ‘participant’, again variously, as:



"(i) That shares or partakes in something; that participates or takes part. (ii) Cognizant or informed of; sharing in the knowledge."

²The definitive record of the English language.

Even taking the most neutral meaning of the term subject (in the context of human experimentation), and comparing it with that of the participant, a subject has an experiment made *upon* them, whereas a participant *shares* in the experimental process. These definitions make it obvious that, even until quite recently, specialists considered the people partaking in their evaluation to be subjects of either the scientist or the experimentation. In reality, most specialists now use of the term participant to enable them to understand that, one-time subjects, are freely and informatively participating in their evaluation and should not be subjugated by that evaluation or the specialists undertaking it. Indeed, as we have seen in ‘evaluation methodologies’ (see [Ethics](#)), the more a human participant feels like a subject, the more they tend to agree with the evaluator. This means that bias is introduced into the experimental framework.

So then, by understanding that we have either evaluation participants, or respondents (a term often used, within sociology and the social sciences, to denote people participating in quantitative survey methods), we implicitly convey a level of respect and, in some ways, gratitude for the participation of the users. Understanding this key factor enables us to make better decisions with regard to participant recruitment and selection, assignment and control, and the definition and measurement of the variables. By understanding the worth of our participants we can emphasise the humane and sensitive treatment of our users who are often put at varying degrees of risk or threat during the experiment procedure. Obviously, within the UX domain these risks and threats are usually minimal or non-existent but by understanding that they may occur, and that a risk assessment is required, we are better able to see our participants as equals, as opposed to dominated subjects.

5.3 Keeping Us Honest

Good ethics makes good science and bad ethics makes bad science because the ethical process keeps us honest with regard to the kind of methodologies we are using and the kind of analysis that will be applied.

Ethics is not just about the participant but it is also about the outcomes of the evaluation being performed. Understanding the correct and appropriate methodology and having that methodology double checked by an evaluation ethics committee enables the UX practitioner to better account for all aspects of the evaluation design. As well as including that of the analysis it also encourages the understanding of how the data will be used and stored. As we have seen (see ‘[Bedrock](#)’), good science is based on the principle that any assertion made must have the possibility of being falsified (or refuted). This does not mean that the assertion is false, but just that it is possible to refute the statement. This refutability is an important concept in science, indeed, the term ‘testability’ is related, and means that an assertion can be falsified through experimentation alone. However, this possibility of testability can only occur if the collected data is available and the method is explicit enough to be understandable by a third party. Therefore, presenting your work to an ethics committee enables you to gain an understanding of the aspects of the methodology which are not repeatable without your presence. In addition, it also enables you to identify any weaknesses within the methodology or aspects, which are not covered within the write-up, but which are implicitly understood by yourself or the team conducting the evaluation.

In addition, ethical approval can help us in demonstrating to other scientists or UX specialists, that our methods for data gathering and analysis are well found. Indeed, presentation to an ethical committee is in some ways similar to the peer review process, in that any inadequacies within the methodology can be found before the experimental work commences, or sources of funding are approached. Indeed, the more ethical applications undertaken, the more likely it is that the methodological aspects of the work will become second nature and the evaluation design will in turn become easier.

It is therefore, very difficult to create a evaluation methodology which is inconsistent with the aims of the evaluation or inappropriate with regard to the human participants; and still have it pass the ethical vetting as applied by a standard ethics committee. Indeed, many ethical committees require that a technical report is submitted after the ethical procedures are undertaken so that an analysis of how the methodology was applied in real-life can be understood; and if there are any possible adverse consequences from this application. These kinds of technical reports often require data analysis and deviations from the agreed methodology to be accounted for. If not there may be legal consequences, or at the very least, problems with regard to the obligatory insurance of ethically approved procedures; for both the specific work and for future applications.

5.4 Practical Ethical Procedures

Before commencing any discussion regarding ethics and the human participant it may be useful to understand exactly what we mean by ethics. Here we understand that ethics relates to morals; a term used to describe the behaviour or aspects of the human character that is either good or bad, right or wrong, good or evil. Ethics then is a formulated way of understanding and applying the characteristics and traits that are either good or bad; in reality, the moral principles by which people are guided. In the case of human-computer interaction, ethics is the rules of conduct, recognised by certain organisations, which can be applied to aspects of UX evaluation. In the wider sense ethics and morals are sub-domains of civil, political, or international law [**Sales and Folkman, 2000**].

Ethical procedures then are often seen as a superfluous waste of time, or at best an activity in form filling, by most UX specialists. The ethical procedures that are required seem to be obtuse and unwieldy especially when there is little likelihood of danger, or negative outcomes, for the participants. The most practitioners reason that this is not, after all, a medical study or a clinical trial, there will be no invasive procedures or possibility of harm coming to a participant, so why do we need to undergo an ethical control?

This is entirely true for most cases within the user experience domain. However, the ethical process is a critical component of good evaluation design because it encourages the UX specialist to focus on the methodology and the analysis techniques to be used within that methodology. It is not the aim of the organisational ethical body to place unreasonable constraints on the UXer, but more properly, to make sure that the study designers possess a good understanding of what methodological procedures will be carried out, how they will be analysed, and how these two aspects may impact the human participants.

Ethics are critical component of good evaluation design because it encourages the UX specialist to focus on the methodology and the analysis techniques to be used within that methodology.

In reality then, ethical applications are focussed on ensuring due diligence, a form of standard of care relating to the degree of prudence and caution required of an individual who is under a duty of care. To breach this standard may end in a successful action for negligence, however, in some cases it may be that ethical approval is not required at all:

“Interventions that are designed solely to enhance the well-being of an individual patient or client and that have a reasonable expectation of success. The purpose of medical or behavioural practice is to provide diagnosis, preventative treatment, or therapy to particular individuals. By contrast, the term research designates and activities designed to develop or contribute generalizable knowledge (expressed, for example, in theories, principles, and statements of relationships).”

In the case of UX then, the ethical procedure is far more about good evaluation methodology than it is about ticking procedural boxes.

As you already know, UX is an interdisciplinary subject and a relatively new one at that. This means that there are no ethical guidelines specifically to address the user experience domain. In this case, ethical principles and the ethical lead is taken from related disciplines such as psychology, medicine, and biomedical research. This means that, from the perspective of the UX specialist, the ethical aspects of the evaluation can sometimes seem like an incoherent ‘rule soup’.

To try and clarify the UX ethical situation I'll be using principles guidelines and best practice taken from a range of sources, including:

- The American Psychological Association's (APA), 'Ethical Principles of Psychologists and Code of Conduct';
- The United States Public Health Service Act (Title 45, Part 46, Appendix B), 'Protection of Human Subjects';
- The Belmont Report, 'Ethical Principles and Guidelines for the Protection of Human Subjects of Research';
- The Council of International Organisations of Medical Sciences, 'International Ethical Guidelines for Epidemiological Studies'; and finally
- The World Medical Association's, 'Declaration of Helsinki – Ethical Principles for Medical Research Involving Human Subjects'.

5.5 Potted Principles of Practical Ethical Procedures

As a responsible UX'er, you are required to conduct your evaluations following the highest scientific and ethical standards **[Association, 2003]**. In particular, you must protect the rights, interests and dignity of the human participants of the evaluation, and your fellow researchers themselves, from harm. The role of your Ethics body or committee is to ensure that any relevant experimentation or testing meets these

high ethical standards, by seeking all information appropriate for ethical assessment, interviewing you as an applicant where possible, and coming to a judgement as to whether the proposed work should be: given ethical approval, refused ethical approval, or given approval subject specified conditions.

5.5.1 In brief...

We can see that the following list of key principles should be taken into account in any evaluation design be it in the field or within the user laboratory.

- *Competence* Keep up to date, know your limitations, ask for advice;
- *Integrity* Have no axe to grind or desired outcome;
- *Science* Follow the Scientific Method;
- *Respect* Assess your participant's autonomy and capability of self-determination, treat participants as equals, ensure their welfare;
- *Benefits* Maximising benefits and minimising possible harms according to your best judgement, seek advice from your organisations ethics committee;
- *Justice* Research should be undertaken with participants who will benefit from the results of that research;
- *Trust* Maintain trust, anonymity, confidentiality and privacy, ensure participants fully understand their roles and responsibilities and those of the experimenter; and finally
- *Responsibility* You have a duty of care, not only to your participants, but also to the community from which they are drawn, and your community of practice.

Some practical considerations must be undergone when an application for ethical approval is being created. The questions are often similar over most ethical applications which involve UX. When creating any ethical procedure, you need to ask yourself some leading ethical questions to understand the various requirements of the ethical application. In some cases understanding these questions before you write your methodology is preferable, in this way you can be led down unethically valid path by understanding what changes you need to make to your evaluation work while you are designing it, and before it becomes too difficult to revise. Here I arrange aspects of the ethical procedure by ethical principles. These aspects describe the kinds of issues which should be considered when creating a methodology and cover the project: overview, participants, risks, safeguards, data protection and confidentiality, reporting arrangements, funding and sponsorship, and finally, conflicts of interest.

5.5.2 You Must Be Competent

1. Other Staff: Assess, and suitable train, the other staff involved; and
2. Fellow Researchers: Assess the potential for adverse effects, risks or hazards, pain, discomfort, distress, or inconvenience to the researchers themselves.

5.5.3 You Must Have Integrity

1. Previous Participation: Will any research participants be recruited who are involved in existing evaluations or have recently been involved in any evaluation before recruitment? Assess the steps you will take to find out;
2. Reimbursement: Will individual evaluation participants receive reimbursement of expenses or any other incentives or benefits for taking part in this evaluation? Assess if the UXers may change their attitudes towards participants, knowing this;
3. Conflict of interest: Will individual evaluators receive any personal payment over and above normal salary and reimbursement of expenses for undertaking this evaluation? Further, will the host organisation or the researcher's department(s) or institution(s) receive any payment or benefits more than the costs of undertaking the research? Assess the possibility of these factors creating a conflict of interest;
4. Personal Involvement: Does the Chief UX Specialist or any other UXer/collaborator have any direct personal involvement (e.g. financial, share-holding, personal relationship, etc.) in the organisation sponsoring or funding the evaluation that may give rise to a possible conflict of interest? Again, assess the possibility of these factors creating a conflict of interest; and
5. Funding and Sponsorship: Has external funding for the evaluation been secured? Has the external funder of the evaluation agreed to act as a sponsor, as set out in any evaluation governance frameworks? Has the employer of the Chief UX Specialist agreed to act as sponsor of the evaluation? Again, assess the possibility of these factors creating a conflict of interest.

5.5.4 Conform to Scientific Principles

1. Objective: What is the principal evaluation question/objective? Understanding this objective enables the evaluation to be focused;
2. Justification: What is the scientific justification for the evaluation? Give a background to assess if any similar evaluations have been done, and state why this is an area of importance;
3. Assessment: How has the scientific quality of the evaluation been assessed? This could include independent external review; review within a company; review within a multi-centre research group; internal review (e.g. involving colleagues, academic supervisor); none external to the investigator; other, e.g. methodological guidelines;
4. Demographics: Assess the total number of participants required along with their, sex, and the target age range. These should all be indicative of the target evaluation population;
5. Purpose: State the design and methodology of the planned evaluation, including a brief explanation of the theoretical framework that informs it. It should be clear exactly what will happen to the evaluation participant, how many times and in what order. Assess any involvement of evaluation participants or communities in the design of the evaluation;
6. Duration: Describe the expected total duration of participation in the study for each participant;

7. Analysis: Describe the methods of analysis. These may be by statistical, in which case specify the specific statistical experimental design and justify why it was chosen, or using other appropriate methods by which the data will be evaluated to meet the study objectives;
8. Independent Review: Has the protocol submitted with this application been the subject of review by an independent evaluator, or UX team? This may not be appropriate but it is advisable if you are worried about any aspects; and
9. Dissemination: How is it intended the results of the study will be reported and disseminated? Internal report; Board presentation; written feedback to research participants; presentation to participants or relevant community groups; or by other means.

5.5.5 Respect Your Participants

1. Issues: What do you consider to be the main ethical issues that may arise with the proposed study and what steps will be taken to address these? Will any intervention or procedure, which would normally be considered a part of routine care, be withheld from the evaluation participants. Also, consider where the evaluation will take place to ensure privacy and safety;
2. Termination: Assess the criteria and process for electively stopping the trial or other evaluations prematurely;
3. Consent: Will informed consent be obtained from the evaluation participants? Create procedures to define who will take consent and how it will be done. Give details of how the signed record of consent will be obtained and list your experience in taking consent and of any particular steps to provide information (in addition to a written information sheet) e.g. videos, interactive material. If participants are to be recruited from any of the potentially vulnerable groups, assess the extra steps that will be taken to assure their protection. Consider any arrangements to be made for obtaining consent from a legal representative. If consent is not to be obtained, think about why this is not the case;
4. Decision: How long will the participant have to decide whether to take part in the evaluation? You should allow an adequate time;
5. Understanding: Implement arrangements for participants who might not adequately understand verbal explanations or written information given in English, or who have special communication needs (e.g. translation, use of interpreters, etc.);
6. Activities: Quantify any samples or measurements to be taken. Include any questionnaires, psychological tests, etc. Assess the experience of those administering the procedures and list the activities to be undertaken by participants and the likely duration of each; and
7. Distress: Minimise, the need for individual or group interviews/questionnaires to discuss any topics or issues that might be sensitive, embarrassing or upsetting;; is it possible that criminal or other disclosures requiring action could take place during the study (e.g. during interviews/group discussions).

5.5.6 Maximise Benefits

1. Participants: How many participants will be recruited? If there is more than one group, state how many participants will be recruited in each group. For interna-

tional studies, say how many participants will be recruited in the UK and total. How was the number of participants decided upon? If a formal sample size calculation was used, indicate how this was done, giving sufficient information to justify and reproduce the calculation;

2. Duration: Describe the expected total duration of participation in the study for each participant;
3. Vulnerable Groups: Justify their inclusion and consider how you will minimise the risks to groups such as: children under 16; adults with learning difficulties; adults who are unconscious or very severely ill; adults who have a terminal illness; adults in emergency situations; adults with mental illness (particularly if detained under mental health legislation); adults with dementia; prisoners; young offenders; adults in Scotland who are unable to consent for themselves; those who could be considered to have a particularly dependent relationship with the investigator, e.g. those in care homes, students; or other vulnerable groups;
4. Benefit: Discuss the potential benefit to evaluation participants;
5. Harm: Assess the potential adverse effects, risks or hazards for evaluation participants, including potential for pain, discomfort, distress, inconvenience or changes to lifestyle for research participants;
6. Distress: Minimise, the need for individual or group interviews/questionnaires to discuss any topics or issues that might be sensitive, embarrassing or upsetting;; is it possible that criminal or other disclosures requiring action could take place during the study (e.g. during interviews/group discussions).
7. Termination: Assess the criteria and process for electively stopping the trial or other evaluation prematurely;
8. Precautions: Consider the precautions that have been taken to minimise or mitigate any risks;
9. Reporting: instigate procedures to facilitate the reporting of adverse events to the organisational authorities; and
10. Compensation: What arrangements have been made to provide indemnity and compensation in the event of a claim by, or on behalf of, participants for negligent harm and non-negligent harm;

5.5.7 Ensure Justice

1. Demographics: Assess the total number of participants required along with their, sex, and the target age range. These should all be indicative of the population the evaluation will benefit;
2. Inclusion: Justify the principal inclusion and exclusion criteria;
3. Recruitment: How will potential participants in the study be (i) identified, (ii) approached and (iii) recruited;
4. Benefits: Discuss the potential benefit to evaluation participants;
5. Reimbursement: Assess whether individual evaluation participants will receive reimbursement of expenses or any other incentives or benefits for taking part in this evaluation; and
6. Feedback: Consider making the results of evaluation available to the evaluation participants and the communities from which they are drawn.

5.5.8 Maintain Trust

1. Monitoring: Assess arrangements for monitoring and auditing the conduct of the evaluation (will a data monitoring committee be convened?);
2. Confidentiality: What measures have been put in place to ensure confidentiality of personal data? Will encryption or other anonymising procedures be used and at what stage? Carefully consider the implications involving any of the following activities (including identification of potential research participants): examination of medical records by those outside the medical facility, such as UX specialists working in bio-informatics or pharmacology; electronic transfer by magnetic or optical media, e-mail or computer networks; sharing of data with other organisations; export of data outside the European Union; use of personal addresses, postcodes, faxes, e-mails or telephone numbers; publication of direct quotations from respondents; publication of data that might allow identification of individuals; use of audio/visual recording devices; storage of personal data as manual files including, home or other personal computers, university computers, private company computers, laptop computers
3. Access: Designate someone to have control of and act as the custodian for the data generated by the study and define who will have access to the data generated by the study.
4. Analysis: Where will the analysis of the data from the study take place and by whom will it be undertaken?
5. Storage: How long will the data from the study be stored, and where will this storage be; and
6. Disclosure: Create arrangements to ensure participants receive any information that becomes available during the research that may be relevant to their continued participation.

5.5.9 Social Responsibility

1. Monitoring: Assess arrangements for monitoring and auditing the conduct of the evaluation (will a data monitoring committee be convened?);
2. Due to Diligence: Investigate if a similar application been previously considered by an Ethics Committee in the UK, the European Union or the European Economic Area?
3. Disclosure: State if a similar application been previously considered by an Ethics Committee in the UK, the European Union or the European Economic Area. What was the result of the application, how does this one differ and state why you are (re)submitting it;
4. Feedback: Consider making the results of evaluation available to the evaluation participants and the communities from which they are drawn.

5.6 Summary

Remember, Ethical applications are focussed on ensuring due diligence. Simply this is a standard of care relating to the degree of prudence and caution required of UX specialist who is under a duty-of-care. As a responsible UX'er, you are required to

conduct your evaluation by the highest ethical standards. In particular, you must protect the rights, interests and dignity of the human participants of the evaluation, and your fellow UX'ers, from harm.

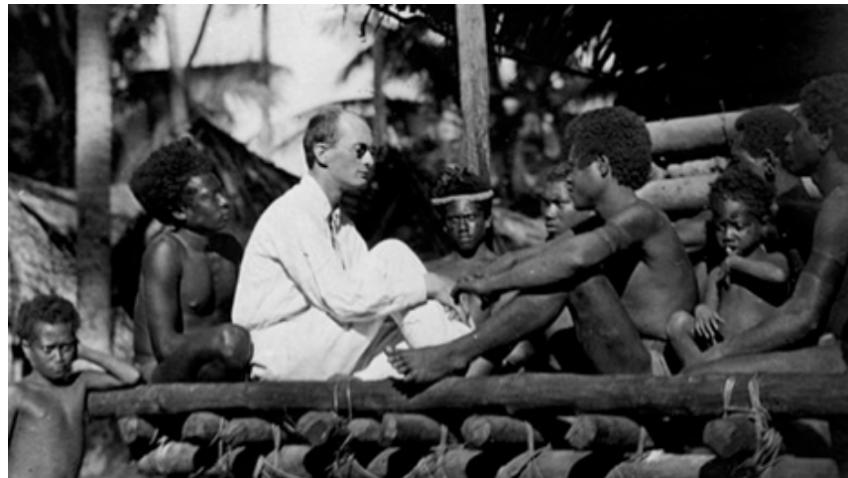


Figure: Off the Verandah. Detailed participant observation is important and anthropologists must have daily contact with their informants if they are to adequately record the “imponderabilia of everyday life” that are so important to understanding a different culture. —Image Credit: Bronislaw Malinowski. *Argonauts of the Western Pacific*.

To help you accomplish this duty-of-care, you should ask yourself the following questions at all stages of your evaluation design and experimentation:

- (1) Is the proposed evaluation sufficiently well designed to be of informational value?
- (2) Does the evaluation pose any risk to participants by such means as the use of deception, of taking sensitive or personal information or using participants who

cannot readily give consent?

- (3) If the risks are placed on participants, does the evaluation adequately control those risks by including procedures for debriefing or the removal or reduction of harm, guaranteeing through the procedures that all information will be obtained anonymously or if that is not possible, guaranteeing that it will remain confidential and providing special safeguards for participants who cannot readily give consent?
- (4) Have you included a provision for obtaining informed consent from every participant or, if participants cannot give it, from responsible people acting for the benefit of the participant? Will sufficient information be provided to potential participants so that they will be able to give their informed consent? Is there a clear agreement in writing between the evaluation and potential participants? The informed consent should also make it clear that the participant is free to withdraw from an experiment at any time.
- (5) Have you included adequate feedback information, and debriefing if deception was included, to be given to the participants at the completion of the evaluation?
- (6) Do you accept full responsibility for safety?
- (7) Has the proposal been reviewed and approved by the appropriate review board?

As we can see these checks cover the range of the principles informing the most practical aspects of human participation in ethically created evaluation projects. By applying them, you will be thinking more holistically about the ethical processes at work within your evaluations, especially if you quickly re-scan them at every stage of the evaluation design. Finally, this is just an abbreviated discussion of ethics; there are plenty more aspects to consider, [and some of the most salient points are expanded upon in the Appendix](#).

5.6.1 Optional Further Reading

- [Barry G. Blundell]. Ethics in Computing, Science, and Engineering: A Student’s Guide to Doing Things Right. N.p.: Springer International Publishing, 2021.

- [Nina Brown], Thomas McIlwraith, Laura Tubelle de González. *Perspectives: An Open Introduction to Cultural Anthropology 2nd edition 2020*³
- [Rebecca Dresser]. Silent Partners: Human Subjects and Research Ethics. United Kingdom: Oxford University Press, 2017.
- [Joseph Migga Kizza]. Ethics in Computing: A Concise Module. Germany: Springer International Publishing, 2016.
- [Bronislaw Malinowski]. Argonauts of the Western Pacific (London: Routledge & Keegan Paul, 1922), 290.
- [Perry Morrison] and Tom Forester. Computer Ethics: Cautionary Tales and Ethical Dilemmas in Computing. United Kingdom: MIT Press, 1994.
- [David B. Resnik]. The Ethics of Research with Human Subjects: Protecting People, Advancing Science, Promoting Trust. Germany: Springer International Publishing, 2018.
- [B. D. Sales] and S. Folkman. Ethics in research with human participants. American Psychological Association, Washington, D.C., 2000.

5.6.2 International Standards

- [APA] Ethical principles of psychologists and code of conduct. American Psychological Association, 2003.
- [BELMONT] The Belmont Report, ‘Ethical Principles and Guidelines for the Protection of Human Subjects of Research’.
- [CIOMS] The Council of International Organisations of Medical Sciences, ‘International Ethical Guidelines for Epidemiological Studies’.
- [HS-ACT] The United States Public Health Service Act (Title 45, Part 46, Appendix B), ‘Protection of Human Subjects’.
- [HELSINKI] The World Medical Association’s, ‘Declaration of Helsinki – Ethical Principles for Medical Research Involving Human Subjects’.



Self Assessment Questions

Try these without reference to the text:

1. What is the importance of Ethics in the context of UX?
2. Why did Ethics evolve pre-World War 2?
3. Why did Ethics evolve post-World War 2?
4. How would you get Practical Ethical Approval?
5. **In brief...** What are the 8 main Ethical Principles?

³<https://perspectives.americananthro.org/>

6. Gathering User Requirements

I'll sit down to work on an assignment, start sketching screens or composing an outline, then suddenly stop and say to myself, these are all 'hows!' What is the 'what?' What am I really trying to deliver?

– Marc Rettig

In his 1992 article, Marc Rettig, defines the methods we use for understanding requirements as ‘Hat Racks for Understanding’ [Rettig, 1992] -

“There is an implicit ‘what’ underlying most software projects ‘understanding.’ The users of your software are trying to learn something from the, data they are seeing. They need to understand the process of working with the software, and how it relates to the rest of their job.

They need to understand how to use the software itself. When understanding is made an explicit part of the requirements, the ‘how’ that is, the design will change. How can we help our users gain understanding? ... As the collection grows, a few common themes are emerging. Some of the themes are often discussed in computing circles, such as user centred design, iterative development, user testing, and concern for human factors. These are all ‘big ideas.’ However, what I really need for my day-to-day work are a few easy-to-handle tools – ways, of thinking that will help me to do well at all those small decisions whose cumulative effect is so large...”

When writers and designers want to explain something, they use visual ‘hat racks’ (maps, diagrams, charts, lists, timelines) that help us understand how our world is organised. Information hats may be hung on these racks to reveal patterns, connections, and relationships. As the scientific visualisation community has found out, one of the exciting things about computers is the way they let you dynamically change the racks on which your information hats are hanging – you can see the same information in many different ways... Although there seems to be an infinity of ways to organise information, [Wurman] notes there are really only five general ways: time; location; continuum or magnitude; and category.”



Links to Evaluation Analysis

Before starting, it is useful to point forwards to Evaluation Analysis). The gathering and analysis of user data shares similarities and also some of these techniques can be used in Evaluation Analysis and vice-versa.

6.1 What is Requirements Analysis?

Requirements engineering is a systematic and critical process in software and system development that focuses on identifying, documenting, and managing the needs and

expectations of stakeholders for a particular system or product. It plays a crucial role in ensuring that the final product meets its intended purpose and satisfies user needs. Requirements engineering (also known as ‘elicitation’) is a crucial phase in the software development or system engineering process, focused on gathering, documenting, and understanding the needs and expectations of stakeholders. The primary goal of requirements elicitation is to collect information about what the system or software should do and how it should behave. This information serves as the foundation for the project and helps ensure that the final product meets stakeholder requirements.

“Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is an important aspect of project management.”

– TechTarget Network 2020

Further, requirements analysis is a phase within the broader process of requirements engineering. It involves a detailed examination and evaluation of the gathered requirements to ensure that they are complete, accurate, and well-understood. The primary goal of requirements analysis is to refine the initial set of requirements, identify potential issues or conflicts, and establish a solid foundation for the subsequent phases of the software or system development process. There are many types of requirements: Customer requirements; Structural requirements; Behavioural requirements; Functional requirements; Non-functional requirements; Performance requirements; Design requirements; Derived requirements; and Allocated requirements. Indeed, how we gather requirements can also affect the outcomes we will discuss [Laboratory settings](#) later, but briefly Laboratory settings controlled but not ecologically valid. ‘Free living’ or ‘in the field’ are ecologically valid but not controlled. Simply, ecological validity is a critical consideration in UX work, as it assesses the extent to which our findings accurately represent and can be applied to real-world situations. UX specialists strive to design studies, experiments, interfaces, interactions that balance the need for control and rigour (Laboratory) with the goal of mimicking the complexities and nuances of everyday life (free-living). This ensures that the knowledge generated from UX work has relevance and practical implications when the work is actually deployed and used.

6.1.1 Observation, Analysis, Discussion, Interpretation (OADI)

There are many different takes and methods on the steps required to gather and make sense of user requirements which will cover over the next two or three chapters. However, we can summarise these as: **‘Observation’** (Observe / Monitor / Collect) Look at what people do, and note it down; **‘Analysis’** (Analyse / Correlate) Analysis the ‘stuff’ they produce; **‘Discussion’** (Discuss / Reflect-Back) Try to find out what these observations and analysis means; and, **‘Interpretation’** adds your knowledge to the analysed data. These are not necessarily disjoint steps but can be conjoined and (or) iterative.

6.2 Digital Phenotyping

Digital Phenotyping is the “moment-by-moment quantification of the individual-level human phenotype in situ using data from personal digital devices”, in particular smartphones.

The data can be divided into two subgroups, called active data and passive data, where the former refers to data that requires active input from the users to be generated, whereas passive data, such as sensor data and phone usage patterns, are collected without requiring any active participation from the user.

The term was coined by Jukka-Pekka Onnela in 2015 but it has been undertaken for over a decade before it was named.



Figure: How Digital Phenotyping Works. How Digital Phenotyping Works. —Image Credit: Huckvale, K., Venkatesh, S. & Christensen, H. <https://doi.org/10.1038/s41746-019-0166-1>

As can bee seen in [Fig: How Digital Phenotyping Works](#) the cycle of of data collect and then analysis is highly conjoined in digital phenotyping, and so for this reason we will cover it in more detail later. You can [skip forward to a more in-depth discussion](#) if you'd like to understand this earlier.

6.3 User Centred Design

Old-style software engineering often forgot, or ignored, the ‘what’ and, therefore, the user. As a remedy to this failing a design method – variously called user centred design, human centred design, co-operative evaluation, participatory design, or user experience design – was created which would place a user at the centre of the system. In reality, this means that there is a long user consultation period before any software is developed such that the designers can more accurately represent the requirements

– the ‘what’ – of those users. It would be fair to say that there are many different sub-styles of, what I will now call, User Centred Design (UCD) and the texts and standards that are listed in this section are good first steps [9241-210:2010, 2010]. However, there are many overriding questions and principles that are appropriate to them all and in some regard can be used without strict reference to a specific UCD methodology. As a trained UX'er, you should not be following any particular method by rote, but you should rather understand the strengths and weaknesses of each and select the tools that are most appropriate to your specific requirements within your current design process¹.

The term “user-centred design” was popularized and coined by Donald A. Norman, a cognitive scientist and usability engineer. In the 1980's, Norman played a key role in advocating for a shift in design philosophy that prioritized the needs, goals, and experiences of users in the design process. Norman's book “The Design of Everyday Things,” originally published in 1988 under the title “The Psychology of Everyday Things,” explored the principles and concepts of user-centred design. In the book, he emphasized the importance of designing products and systems that align with users' mental models, cognitive abilities, and behavioural patterns. Norman's work highlighted the significance of considering the users' perspectives, abilities, and context when designing interfaces, products, and environments. He advocated for intuitive, understandable designs that minimize the user's cognitive load and facilitate ease of use. Since the publication of “The Design of Everyday Things,” user-centered design has gained widespread recognition and acceptance as a fundamental approach in various fields, including product design, software development, and user experience design. The principles and methodologies derived from user-centred design have shaped the development of intuitive, user-friendly, and effective products and systems that meet the needs and expectations of users.

Springing from UCD came “universal usability”, a concept created by Ben Shneiderman, an American computer scientist and professor. Shneiderman is known for his significant contributions to the field of human-computer interaction (HCI) and user interface design. Shneiderman introduced the concept of universal usability in the late 1990's as an extension of the principle of UCD. While universal design primarily focuses on physical and environmental accessibility, universal usability expands the concept to encompass digital interfaces and technologies. Universal usability emphasizes designing interactive systems and interfaces that are accessible, usable, and effective for a wide range of users, including individuals with diverse abilities, backgrounds, and levels of expertise. It aims to make technology inclusive and user-friendly for everyone, regardless of their physical, cognitive, or sensory characteristics. Shneiderman's work on universal usability highlighted the importance of designing interfaces that are intuitive, learnable, and efficient, enabling users to accomplish their tasks effectively and efficiently. He advocated for the consideration of users' needs, preferences, and capabilities in the design process to create interfaces that are accessible and enjoyable for all users.

Universal usability and user-centred design share common goals and principles, although they approach them from slightly different perspectives. UCD focuses on designing products, systems, or interfaces that meet the needs, goals, and preferences of the target users. It emphasizes understanding the users' requirements, behaviours,

¹You should also remember that changes and tweaks to methods or processes may indeed be useful and based on long experience – however you should also be realistic – they may also be based on the desire to sell more, and different, books and training texts.

and contexts through user research and involving users in the design process. UCD aims to create intuitive, usable, and satisfying experiences for specific user groups. On the other hand, universal usability expands the concept of user-centred design to consider a broader range of users with diverse abilities, backgrounds, and characteristics. It emphasizes designing interfaces and systems that are accessible and usable by the widest possible audience, regardless of individual differences. Universal usability seeks to remove barriers and accommodate the needs of individuals with disabilities, ensuring equal access and usability for all.

In essence, while user-centred design focuses on meeting the needs of a specific target user group, universal usability takes a more inclusive approach by considering the needs of a broader range of users. Both approaches recognize the importance of understanding users, involving them in the design process, and creating interfaces that are intuitive, effective, and satisfying to use.

Universal usability can be seen as an extension of user-centred design, incorporating accessibility and inclusivity as essential components. By considering the principles of universal usability, user-centred design can ensure that the products and systems being developed are accessible and usable by as many people as possible, regardless of their abilities or characteristics. In practice, user-centred design and universal usability can be complementary approaches, with user research and usability testing helping to uncover and address the diverse needs and abilities of users, including those with disabilities. By integrating the principles of universal usability into user-centred design processes, designers can create more inclusive and user-friendly experiences for a broader range of individuals.

UCD represents a set techniques that enable participants to take some form of ownership within the evaluation and design process. It is often thought that these participants will be in some way knowledgeable about the system and process under investigation, and will, therefore, have an insight into the systems and interfaces that are required by the whole organisation. UCD methods are closely linked to the [think aloud protocol](#), but instead of entirely focusing on evaluation the users are encouraged to expand their views with suggestions of improvements based on their knowledge of the system or interfaces that are required. Indeed, the participants are encouraged to criticise the proposed system in an attempt to get to the real requirements. This means that – in some cases – a system design is created before the UCD have begun so that the participants have a starting point.

The UX specialist must understand that a UCD process is not a fast solution; UCD often runs as a focus group based activity and, therefore, active management of this scenario is also required. Enabling each individual to fully interact within the discussion process while the UX'er remains outside of the discussion; acting as a facilitator for the participants views and thoughts is a key factor in the UCD process. Indeed, another major difference between UCD and conventional Requirements Capture is that UCD becomes far more of a conversation between the software engineer and the user/customer.

In terms of requirements analysis², we can think of the functional aspects to be those which are uncovered within the UCD process, while it may be convenient to think of the

²Now you'll start to see some 'bleed' between terms. Requirements Engineering (Elicitation and Analysis) is pretty formal and is 'well' known in the software engineering domain. UCD isn't as well known in the engineering community and uses less formal methods to convey understanding. I'm agnostic to an extent but I think a combination of both is a better way of building our 'Hat Rack for Understanding' than one type alone. In this case, I try to marry terms and use interchangeable language to get the process. In the end, we want to pick tools and techniques – 'magpie-like' – which suit us.

remaining sections of this text as more general-purpose non-functional requirements. In this case, we can see that functional requirements are those which are specific to the development in question. While non-functional requirements might be aspects such as a development should be usable; certain functionality should be reachable within a certain time frame; the system should be accessible to disabled users all users of mobile devices etc.

Of course, this is a very rigid delineation, and there is obviously some exchange between functional and non-functional requirements as defined here; but for our purposes it is simpler to suggest that functional requirements relate to a specific system while non-functional requirements relate to good user experience practise in-general. For instance, in the case of software engineering, usability is an example of a non-functional requirement. As with other non-functional requirements, usability cannot be directly measured but must be quantified using indirect measures or attributes such as, for example, the number of reported problems with the ease-of-use of a system.

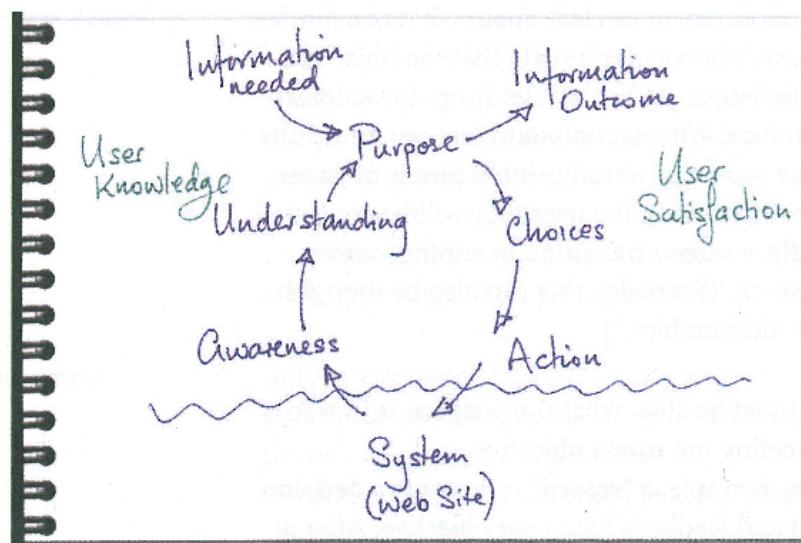


Figure: Awareness, Understanding, Action. Awareness, Understanding, Action. —Image Credit: Cato, 2001.

For most practical purposes, the user centred design uses an iterative looping methodology (for example [Cato, 2001] see [Figure: Awareness, Understanding, Action](#)) not dissimilar to the common software engineering [iterative methodology](#). In this case, a requirements elicitation and analysis process, traditionally called systems requirements engineering, is undergone. This process should enable the UX'er to understand the requirements of the user and to understand how to service the tasks required by those users. It also serves to highlight aspects of the interaction or task that are important or required and gives an indication of how the user design should proceed. However at this stage, the design often does not include any decision about the user experience. Indeed, systems requirements analysis is far more about understanding the user and their place within the wider context of the system as opposed to the specifics of how the interface looks and feels.

The major difference between the traditional engineering methods and UCD is that the users participate far more in UCD and that the cycles are not so rigid. Further,

requirements engineering is often more concerned that all functionality is present and that this functionality works correctly – important, to be sure. However, UCD is interested in making sure the functionality elicited in the requirements capture is the ‘right’ functionality for the users – it is ‘What People Want!’³.

Once the capture has occurred it is time to communicate that information to the software engineers. This is always tricky because words and specifications tend to reduce the information to a manageable amount to communicate, but the richness of the requirements capture can easily be lost in this reduction. UCD was created exactly to address this failing – traditional methods reducing requirements to functional specifications of programmatic functionality – by adding new informal (agile) people first modelling techniques to the mix.

A modelling phase is also included when the concrete requirements for the human facing parts of the system are captured as part of a formal definition using a well-known modelling methodology such as user task analysis, Unified Modelling Language (UML) Use Case Diagrams, or general use case diagrams, etc. These modelling formalisms enable the information that has been captured in the requirements analysis phase to be transmitted to the software engineers for development and inclusion within the system⁴. They also enable the software engineers to tell if their system meets the UXers specification as defined within these models; and, therefore, enables a separation of concerns between the UX engineer and the software engineer. As is implicit in the discussion of modelling, the final stage is the development of the human aspects of the system as well as the design of the interface. In this case, a sub-iteration is often undertaken in an agile manner, often with the user being directly present. At a minimum, the user should have a high input into both the interface artefacts and should include views on how they work based on their direct knowledge of the job at hand. Once these phases have been undertaken the iterative cycle begins again until no further changes are required in any of the requirements elicitation, analysis, system design, and development.

In all cases, you must think: what information do I need to build a system, how can I get this information, and how can I disseminate this information into the software engineering process, once acquired? Cato simply summarises this as Discover, Design, and Use (see [Figure: Discover, Design, Use](#)), but understanding the ‘what’ is the most important part of the equation in the requirements analysis.

One final, but important, thought before we move on; Fred Brooks in his definitive book ‘The Mythical Man Month’

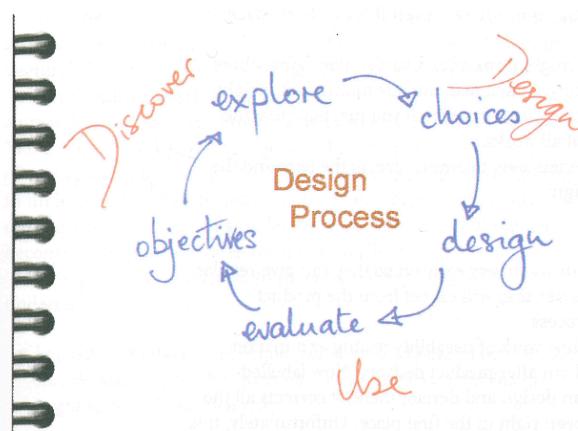


Figure: Discover, Design, Use. Discover, Design, Use.
—Image Credit: Cato, 2001.

³There's a caveat here, in that in some cases users do not always know what they want or need, or what is appropriate or inappropriate. Some organisations (such as Apple) do not even rely on UCD or focus groups and the like; instead preferring to rely on their market knowledge and expertise to design and produce devices.

⁴Currently, some organisations get their software engineers to elicit requirements, although I would contend that this is not necessarily in their area of expertise.

states that ‘When designing a new kind of system, a team will design a throw-away system (whether it intends to or not). This system acts as a ‘pilot plant’ that reveals techniques that will subsequently cause a complete redesign of the system.’, focus on the phrase ‘whether it intends to or not’, which really means plan for a pilot - because you *will, without question*, be building one⁵.

6.4 What Information Do I Need?

Requirements elicitation is the first step within the requirements engineering process. This engineering process is often separated into three aspects that of elicitation, specification, and validation. Requirements engineering can also be known as requirements analysis that is, by coincidence, the term sometimes used to describe data analysis after the elicitation phase and before the specification phase commences. Requirements engineering encompasses all parts of the system design however here we will focus on UCD aspects and the kinds of methods that are often used early within the development cycle to elicit understanding of the users and their requirements.

The main concern for the UX'er is that the elicitation process is all-encompassing and that the specification is an accurate representation of those elicited requirements. Here we will be concerned with the elicitation; we will not be focusing on validating these requirements for one simple reason – the validation of models within the software engineering domain is neither rigorous or well understood. Most validation processes suggest that the user is involved at each stage. However, validation techniques are often more in keeping with a formal validation of the functional requirements (via Test Driven Development for instance) by the developer, as opposed to expecting high user participation. The exception to this rule is the use of prototyping techniques for understanding the correctness of the tasks and the human facing aspects of the system. If an agile method is used the validation processes is no longer required in this context. However, we can see that the UX'er must use some kind of research methodology for a final evaluation of the system.

There are many terms that are common and run through many different kinds of requirements elicitation processes, often these are used interchangeably between the different processes, and while some methods strictly refer to specific terminology, developers seem to be less pedantic than methodology designers. The basic format when trying to elicit a user’s requirements – the what – is often regarding the role of the user, the action a user (a role) wishes to enact, and the information or object on which that action should be elected. There are different names for this kind of model: roles, actions, and objects; and users, actions, and information are but two.

Further, **there is a common vocabulary to describe a user role**. These user roles are often described using terms such as actor, stakeholder, role, or proxy, and all users have the desired goal, thus:

- ‘Actor’ refers to a specific instance of the users such as a customer, manager, or sales clerk and indeed in some cases the term actor is prefixed by their priority within the specific methods such as a ‘primary’ or ‘secondary’ actor. However, in all cases the actor is someone who actually initiates an action within the system;

⁵We'll discuss these real world implications more fully later.

- ‘Stakeholder’ is similar to an actor in that they are a class of user but, in this case, they are less involved within the actual initiation of the action, instead having an interest in the outcome;
- ‘Role’ is also closely related to an actor or a stakeholder in that it describes the persona the user will be taking, such as a purchaser or a seller. Moreover, in some cases may also be used to refer to the similar class of users as in the actor, such as a customer, manager, or sales clerk for instance; finally, the term
- ‘Proxy’ is used to describe a person who is not a specific user but is playing that role. In this case a proxy sales clerk may actually be a manager who is relating the requirements of the sales clerk to the developer based on their knowledge of the job but actually who do not perform the job directly; mostly this is because the actual user is not available.

Finally, the term ‘*Goal*’ is used to describe the outcome that is required by the actors or stakeholders and these may vary based on the role that each of the users is playing.

The next term to consider is the ‘action’, and, in this case, it is often listed as a plain English phrase such as ‘track my order’, or ‘buy the book’, or ‘compare functionality’. We can see that these terms make a very good descriptors for function or method statements within the main code, and so are often used as such by the software engineers. Finally, an ‘object’ is required for these actions to work upon. In this case, we can derive objects by looking at the nouns within these action phrases such as ‘order’ or ‘book’. In other cases, we must create the object-name if this is not explicitly described – such as ‘functionality’ – in which we might want to create a ‘specification’ object of each item.

So then we can see that in most cases these kinds of terms – and the way you should be thinking – applies to most UCD methodologies; it turns out that thinking about the ‘what’ is not as complicated as we may have imagined.

6.4.1 The Humble Post-It!

There are many tools to use in the user-centred requirements elicitation process, but strangely one of the most effective is the humble post-it note. Through the 70s 80s and 90s, a vast amount of effort was directed into creating computerised design tools specifically for the purpose of conveying design information to the software engineer⁶. The thought was that by computerising the elicitation process we could capture and easily link user requirements that would then serve as the basis of the generated software model; thereby removing the possibility of introducing errors into the mapping from design to software. However, along with this logic for computerisation was introduced a certain inflexibility into the design. Once the text was entered into the system it was very rarely changed or updated, once typed, once printed, it was seen in some way as immutable.

⁶Indeed, I even worked on one such tool called MOOSE ‘Model-based Object Oriented System Engineering’; with Derrick Morris, Gareth Evans, and Peter Green.

Then in the late 1990s and early 2000 came the use of the post-it note. The post-it note offers the flexibility that anything can be written on it, anything can be designed upon it, and it can be placed and moved in many and varied different ways, and on many different surfaces; mostly walls and whiteboards. People could scrawl something and decide later to throw it away, people could write something place it in a certain position – clustered with a post-it notes – and at a later point easily change it to a different position.

The post-it note facilitates the evolutionary design process, and in my opinion, this is without a doubt one of the most important tools to arise in the requirements capture space for a long time.

Now the post-it note is used in many different scenarios: from creating a conference grid (see [Figure: Conference Grid](#) – in this case at the WebArtScience Unconference in London 2010), to building an agile development, through to mapping the emergent requirements of a user centred design (see [Figure: Post-Its in Development](#)).

The first tool you should reach for when starting any design elicitation and mapping process is the post-it note. It will enable a design⁷ to emerge from the users, using a tool that they consider unthreatening, obvious, and familiar. Further, post-it notes have the advantage over other mediums, such as the whiteboard, in that with post-it notes everyone can have his or her say. Everybody has a pen/marker, and everybody has a means of conveying his or her ideas. Remember, once the text is removed from the whiteboard information is lost, a post-it note can be moved to one side such that the idea is not lost. Moreover, the evolution of the design is still recreateable – but maybe does not stay within the focus of the discussion.

Once you have completed this elicitation phase you'll be left with clusters of post-its on a wall or whiteboard, in some cases these will be reasonably neat and orderly, in other cases they will be in informal ‘scrappy’ clouds. You should now photograph those clusters, possibly after you have numbered each post-it note such that you can recreate the clouds at any point in the future, if necessary. Both information on a post-it note and their arrangement on the wall have to mean, and you should consider this meaning when trying to formalise these designs for the software process.



Figure: Conference Grid. Conference Grid. –Image Credit: WebArtScience Unconference.

⁷Within the UCD process the post-it note is very similar to standard index card often used for a process known as ‘Card Sorting’ and, in this case, provides a familiarity to the UX'er, which may not be possible with different kinds of technical software tools.

6.5 How Do I Get ‘The Information That I Need’?

Most methods for collecting data come from a combination of anthropology and sociology⁸ mashed-together with some minor alterations to make them more amenable to the UX'er. This ‘modified-mush’ owes much to social anthropologists following the theoretical tradition of ‘interactionism’; interactionists place emphasis on understanding the actions of participants by their active experience of the world and the ways in which their actions arise and are reflected back on the experience. This is useful for the UX'er as the interactionists component of the method makes it quite suitable for requirements capture focused on tasks and systems.

To support these methods and strategies, many suggest the simultaneous collection and analysis of data. This implies the keeping of substantive notes consisting of a continuous record of the situations events and conversations in which the UX'er participates; along with methodological notes consisting of personal reflections on the activities of the UX'er as opposed to the users –‘[Memoing](#)’. These notes should be preliminary analysed within the field and be indexed and categorised using the standard method of ‘[Coding](#)’ different sections of the notes to preliminary categories which can be further refined once the requirements capture has concluded.

The most used methods for requirements capture are participant observation, interviewing, archival and unobtrusive methods. I suggest that these methods are mainly used for building a body of understanding that is both deep and narrow in extent and scope. This knowledge can then be used to help guide our development. You can expect a better outcome if a variety of methods is used in combination – however, in reality you will mostly be limited by time.

⁸Anthropologists and sociologists describe data capture variously as fieldwork, ethnography, case study, qualitative research, interpretative procedures, and field research. However, among anthropologists fieldwork is synonymous with the collection of data using observational methods. For the sociologist, the term often describes the collection of data using a social survey.

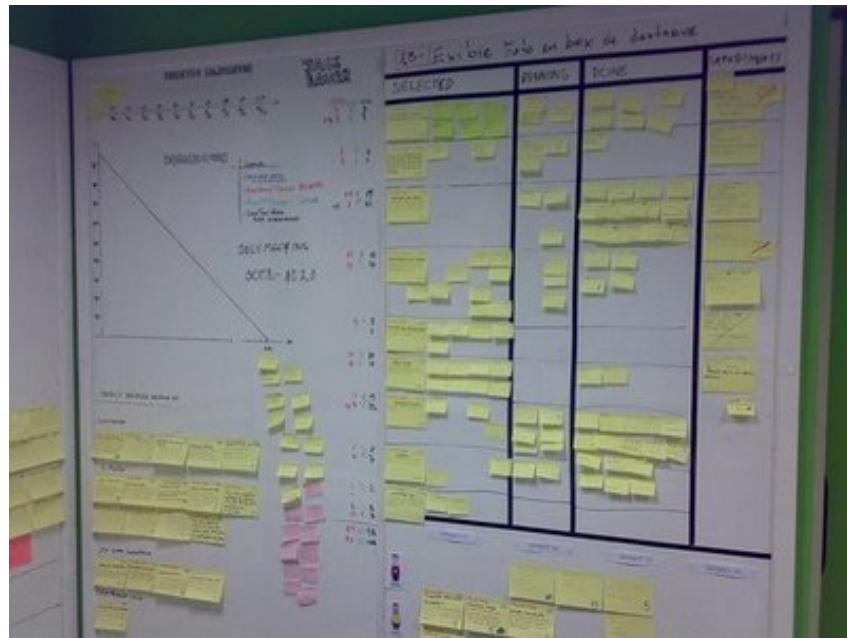


Figure: Post-Its in Development. Agile Development using Informal Post-Its. —Image Credit: Agile.

6.5.1 Got Six Months?

Imagine you are UX'er employed within a company that has just acquired a different organisation. Your company needs to know the processes that are followed within this newly purchased company. You need to understand what kind of jobs their employees do, the systems that they follow, and the level of computerisation, you then need to build an assessment of how their systems and business processes will fit into yours. Even if you only have a small amount of time (six months, say) you will be able to gain a high level of detailed understanding, and scope out the problems of integration, by utilising the ethnographic methodology of observation or participant observation.

Participant observation, is the process by which data is gathered by participating in the daily life of the group or organisation under study. The UX'er watches the people in the study to see what situations they normally meet and how they behave in them and then enters into discussion with some or all of the participants in the situations to discover their interpretations of the events that have been observed. These discussions are often called 'conversations with a purpose' and are very (very) informal interviews – in that you want clarification of certain points without being too prescriptive so as not to bias the users answer. Special classes of participants, called 'informants', may also be called upon, informants are usually 'people-in-the-know'. The person or persons selected to be informants, therefore, have a broad knowledge of the organisation, its services, and its people and can provide clarifications and corroboration for data collected while in-situ. The 'key informant' has an additional significance to the UX'er, which may be because they have extensive knowledge of the user roles, the actions they perform, or the objects on which they perform those actions. They may also host the UX'er in their team, or sometimes because they were the initial contact for the UX'er and served to introduce them into the department or team. Key informants are an excellent way to recover information about past events or organisational culture and memory that are no longer observable. Once in the team the observations of the UX'er should be

methodical and verbose in their note taking and thoughts. Eventually, you will begin to take fewer notes, witness less novel situations, and start to have fewer novel insights; this is an indicator that your observations are coming to an end.

You should remember that your sampling must be representative and that you should remain in the team/organisation until all requirements are captured. Behaviour should be sampled in natural settings and whenever possible be comparative; with the samples and sampling procedure made public. Sampling strategies include probability sampling, in which every unit of the universe under study has the same calculable and nonzero probability of being selected; and non-probability sampling, where there are no means of estimating the probability of units being included in the sample. Non-probabilistic techniques include judgement and opportunistic sampling, in which samples selected are based on opportunity or the need for a certain kind of sample. Snowball sampling, in which an initial set of key informants provide access to their friends and colleagues, thereby forming a larger set. Finally, theoretical sampling, often used in natural settings, is the process of data collection for generating theory whereby the analyst jointly collects, codes, and analyses, the data and decides what to investigate next to develop the theory as it emerges.

While there may be some roles the UX'er may take, these are constantly negotiated and re-negotiated with different informants throughout the requirements capture. Therefore, it can be seen that the interpretive nature of the work means that your role is also important including the influence of your specialism, experience, age and gender, and ethnicity, concerning that of the user group. Remember, as a UX'er you need to observe events by causing as little affect as possible and develop trust and establish relationships often by learning a shared vocabulary that is appropriate to various organisational settings.

You may wish to be more prescriptive than observation if time is limited and focus on ‘Task Analysis’. Task analysis is used to refer to a set of methods and techniques that analyse and describe the way users do their job regarding the activities they perform, including how such activities are structured, and the knowledge that is required for the performance of those activities. The most common and simplistic form is hierarchical task analysis in which a hierarchy of subtasks are built to describe the order and applicable conditions for each primary task.

The reality is that these look like very simple procedures for accomplishing tasks. However, there is a large amount of implicit information captured within these constructs. For example, if a user wishes to borrow a book from the library then part of the task analysis may be that a new checkout card is required, and subtasks may be that the current date needs to be entered on that card as does the borrower's name and catalogue number of the book. In this case, we can see that any automated system must make provision for a pseudo-checkout code to be created and that there must be the fields available for capturing the date, name, and catalogue number within that pseudo-checkout card.

In reality, you can use task analysis to analyse the notes you have collected in your more free-form observations, or if time is limited, focus only on tasks. In this case, you sacrifice completeness, richness, and nuance of time savings (there may not be a cost saving if the requirements capture is incomplete).

In summary then, we can see that the time-scale for observation work is quite long;

between six and twelve months in some cases. This means that when conducting requirements capture in an industrial setting cost constraints often conspire to remove observation from the battery of available techniques. Even so, the use of the key informant as the principal means of soliciting information concerning organisational jargon, culture, and memory can be useful to enhanced other types of research method.



Observation: Participant Observation – ‘Learn by Doing’; Longterm / Interpretive; Embedded / Imbedded (Invisible & Embedded); Copious Notes and Analysis (Ethnography); Conversations with a Purpose; Deep but Narrow Understanding; using Strategies to NOT Standout.



Observation: Task Analysis – ‘Learn by Observing’; Can be Short Term; Invisible / Remote, but can be Close & Discursive; Notes with ‘Formal’ Models; Mid-Deep / Mid-Scope; using Less Interpretation.

6.5.2 Got Six Weeks?

Imagine working for a company that is trying to specify the functionality for a new computer system. This computer system will move the individual paper-based systems of the sales department and those of the dispatch department into one streamlined computerised sales and dispatch system. In this case you know some of the pieces, you know some of the systems that already exist, you know your own company’s business processes and have an understanding of the organisational knowledge. However, you do not know the detailed processes of the sales and dispatch departments separately; and there is no documentation to describe them. Also, you do not know the interface that currently exists between these two departments and so you cannot understand the flow of information and process that will be required in a unified system. In this case, it is useful to employ focus group techniques followed up by structure or semi-structured interviews.

Group interviews – Focus Groups – are used to enable informants to participate and discuss situations with each other while the UX'er steers the conversation and notes the resultant discussions. In the role of moderator (and scribe) the UX'er needs to think of the most practical considerations of working with an organisation. For instance, logistics, piggybacking on existing meetings, heightened influence of the key informers, group dynamics, adhering to the research agenda and spontaneity, and the difficulty in ensuring confidentiality.

A variation on the focus group is the respondent, or member, validation. This is the process whereby a UX'er provides the people on whom the requirements capture is conducted, with an account of the findings; with the aim of seeking corroboration, or otherwise, through the various interview methods. Leading on from these validatory discussions, is the concept of action research which is related to participatory design, participatory research and the practitioner ethnographer. Here, a UX'er along with the users collaborate on the diagnosis of a problem and the development of a solution based on that diagnosis. In reality, this means that the trained UX'er guides the un-trained

users in an attempt to evolve a knowledge of the problem and the possible solutions to that problem.

Interviewing is a useful method in requirements capture. However, you can deliver these in both a structured or a semi-structured interview style. These often take place as friendly exchanges between the user and the UX'er to find out more about the organisation or the users job. Interviewing techniques are best applied when the issues under investigation are resistant to observation. When there is a need for the reconstruction of events. When ethical considerations bar observation. When the presence of a UX'er would result in reactive effects. Because it is less intrusive into the user's day, because there is also a greater breadth of coverage; and finally, because there can be a specific focus applied.

Both single participant interviews or focus group discussions are rarely conducted in isolation and are often part of a broader requirements capture drawing on the knowledge of the UX'er in their application. It is sometimes normal for an aide memoir to be used concerning the interview often. However, all topics are not covered. Three main types of questions are used; 'descriptive questions' that enable users to provide statements about their activities; 'structural questions' which tend to find out how users organised their knowledge; and finally, 'contrast questions' which allow users to discuss the meanings of situations and provide an opportunity for comparison to take place between situations and events in the users world.

In this way, both structured and unstructured interviews are useful to the UX'er because they enable a direct confirmation of partial information and ideas that are being formed as the requirements capture continues. These interviews are often conducted as part of focus group discussions – or afterwards for clarification purposes – because in some cases the communal discussion enables joint confirmation of difficult or implicit aspects of the area under investigation.

Social Processes The social processes that go along with the requirements elicitation phase are often overlooked. For the UX analysts, this is one of the most important aspects of the work, and it should be understood that implicit information mentioned as part of social interactions in an informal work environment may have a high degree of impact upon the requirements engineering phase. Aspects of the work that is not self-evident, errors that often occur, and annoyances and gripes with a system that is not working correctly often come out in an informal social setting; as opposed to the most formal setting of a systems design interview.

In this case, it should be abundantly clear that spending social time with prospective users of the system is useful for catching these small implicit but important pieces of information which are often ignored in the formal process. A full understanding of the system requirements is often not possible in a formal setting because there may be political expediency in not bringing negative aspects of the system to the attention of the requirements engineer; especially if these aspects of the system were designed by management. Further, users may also consider aspects of their job, or the system, too obvious to be explicitly expressed.

In summary, then, we can see that the time-scale for interviewing, focus group discussions, and participatory work can be short to medium term; between one and three months in some cases. This short timescale in data gathering is the main factor attributed to the popularity of these techniques. The data returned is not as rich as that

gathered by observation because the UX'er already has some idea of the domain, and enacts some control over it; for instance, in the choice of questions asked. However, interview methods and variants often provide an acceptable balance between cost, time, and quality of the data.



Observation / Discussion: Social / Shorten 6 Months – Social; Just Talking; Cannot get the full picture; Relaxed - Informal; or Observation on a smaller scale; but More Constrained - More Shallow, than PObs.



Diary Study (in the Field) – Qualitative data; User behaviours, activities, and experiences; Self-reported by participants; Longitudinally; Log specific information; Can be periodically prompted; Can be Biased; Can be remembered incorrectly; and Can be ‘recall’ what they think the UX specialist wants them to.

6.5.3 Lack of Users?

How do you understand the organisational processes that need to be computerised if the people with the most experience have retired or been made redundant? Further, if a computer system does not already exist, how do you also know what the look and feel of the system should be like. If there is no electronic reference point, or reference point for the processes that need to be undertaken, how can you derive this information. Simply, by an investigation of the archival materials of the organisation, paying specific attention to the processes that you are trying to replicate, you should be able to rapidly prototype an initial system that can then be refined by real world user interaction.

An archive is a collection of historical records that have accumulated over the course of an individual or organisation’s lifetime. The archives of an individual may include letters, papers, photographs, computer files, scrapbooks, financial records or diaries created or collected by the individual - regardless of media or format. This means that other computational resources can also be used, such as online photos, avatars, WebCams, and video recording, as well as discussion forums, email archives, mailing-lists or electronic distribution lists. Administrative files, business records, memos, official correspondence and meeting minutes can also be used. There are two different kinds of archival records, being, the running record and including material such as wills, births, marriages, deaths, and congressional records, etc. In contrast to this there are episodic or private records such as sales records, diaries, journals, private letters, and the like. Archival records are normally unpublished and almost always unique, unlike books or magazines, in which many identical copies exist. Archives are sometimes described as information generated as the ‘by-product’ of normal human activities.

The importance of historical records is precisely because of their by-product nature. These records can be classified as either primary sources or secondary sources based on whether they are summarised or edited; touched by someone after their creation. Primary sources are direct records such as minutes, contracts, letters, memoranda, notes, memoirs, diaries, autobiographies, and reports; and secondary sources include

transcripts, summaries, and edited materials. The second distinction is between public documents and private documents and solicited and unsolicited responses. Life histories, autobiographies from an insiders point of view are also very useful as well as diaries and diary interviews. Letters are considered to be an important source as they are primary, often private, and mostly unsolicited, therefore they are written for an audience and captured, within the writing, is an indicator of the different kinds of social relationships within the letter style. However, a degree of circumspection is required when evaluating personal documents as to the authenticity, the authors possible distortion and deception.

Once selected, an archival material can be subjected to analysis as part of the process of understanding the work and placing it in context; you could think of this as being similar to data-mining. Content analysis is an approach to the analysis of archival material that seeks to quantify content regarding predetermined categories; a process called ‘coding’. Aspects that may be sampled are significant actors, words, subjects and themes, and dispositions. Also, secondary sources may be subject to a secondary analysis, in that data, which has been pre-collected, undergoes a second analysis possibly having completely different research questions applied than the original creator of the secondary source had in mind.

Organisations run on forms, order forms, summary sheets, time-sheets, work rosters, invoices, and purchase orders, in fact, any written systemised approach to fit data into a well understood homogenised format. In this way, forms provide an implicit understanding of the kinds of work that prospective users will carry out, the terms that will be familiar to them, and information requirements that are not used as well as those that are overused. Therefore, it is easy to see that forms are a structured collection of system requirements. The positive aspect of understanding forms is that their interpretation is often very straightforward. Whereby users neglected to make certain obvious aspects of the work they do forms do not, further, the acquisition of forms and their processing is easy because forms are commonly available within the organisation. Indeed, in many cases forms can be used to reverse engineer a database schema including attributes entities and relationships that go to make up that schema. This means that the results of this kind of analysis are far more amenable to formal modelling than other kinds found in the elicitation processes.

As with all good computer science, data reuse is a primary concern, and this is also the case with requirements capture. The UX analyst should look out for requirements that have already been captured for some other application. These may be reusable in specifying a similar application or can be reanalysed to add extra depth to the requirements elicitation process already underway. Indeed, even though previous requirements documents may be out-dated, some aspects may serve as a useful starting point to begin interview discussions or focus group activities. Here questions about the old system, in comparison to the new system, can be posed, and the responses of the users gauged to inform the creation of the new system. In addition to standard requirements reuse, aspects of the specification can be reverse engineered such that some of the analytical information can be identified and indeed parts of the current software or systems processes can also be reverse engineered to identify their constituent parts enabling the creation of models that form a higher level of abstraction.

In summary, then, grabbing and analysing archival; material – old forms, reports, past memos, and the like – all help the UX'er without users to create a picture the needs of

the organisation and users they are working with. It is rare that archival material will be your only source for requirements capture, but it may be a good initial starting point if your users do not have the time, initially, to spend bringing you up to speed.



Archive – Describe an organisation by the things it produces; The Letters and Trails it Creates; Important Documentation; Logs and Logbooks; in short The artefacts of the systems already created by the users.

6.5.4 Relationship to the Digital Twin

A digital twin is a virtual representation of a real-world physical object, system, or process. It's a digital counterpart that mirrors the physical entity in a digital environment, often in real-time or near-real-time. This concept emerged from the convergence of technologies like the Internet of Things (IoT), data analytics, and simulation.

Digital twins are used across various industries, including manufacturing, healthcare, energy, aerospace, and more, to provide insights, analysis, and predictions about the real-world counterpart. They can range from simple models representing individual objects to complex models representing entire systems or environments.

Digital twins provide a visual representation of the physical entity, making it easier to monitor and understand its status and behaviour. They incorporate data from sensors, IoT devices, and other sources to reflect the real-time conditions and performance of the physical counterpart. Digital twins enable simulations and “what-if” scenarios, allowing businesses to test changes, optimizations, and potential outcomes before implementing them in the physical world. By analysing historical and real-time data, digital twins can provide predictive insights into potential issues, failures, or performance improvements. Digital twins allow remote monitoring and management of physical entities, facilitating maintenance, troubleshooting, and adjustments from a distance. As more data is collected and analysed, digital twins can be refined and improved, leading to a better understanding of the physical entity’s behaviour.

The concept of digital twins continues to evolve, driven by advancements in technology and the increasing need for efficient data-driven decision-making in various industries. So in reality when you are gathering requirements you are often intent on being able to build a digital twin of the real world process or system. You may of course wish to improve things but in the first place you need to understand and at least digitally model the current system be that digital or analogue. You can [skip forward to understand how digital twins and digital phenotyping relate](#) if you'd like to understand this earlier.



Tread Lightly, Tread Carefully!

Please ‘Be Careful!’ All of the requirements capture processes, listed above, require the UX’er to become a major part of the UCD requirements elicitation process. This means you are in a very privileged position in which you can inadvertently influence the outcomes of the requirements capture by unintentionally exerting influence on your users. You must be very careful not to be biased when it comes to soliciting feedback on anything you have captured; including any analysis you have made. It is not for you to solicit confirmation of your work from the users but for you to understand their needs without exerting any influence whatsoever. Only in this way can we be sure that the requirements you have captured are the users’ – and not yours or your company. Remember, there are **many cautionary tales as to bias within the requirements gathering process:**

One of the most well-known examples is the launch of New Coke. “*Coca-Cola invested in cutting-edge customer research to ensure that New Coke would be a big success. Taste tests with thousands of consumers clearly showed that people preferred it. The reality was somewhat different, however. Hardly anyone bought it.*”

6.6 Summary

Once the elicitation and analysis phase has been complete it maybe time to move this knowledge into a more formalised description of the user, which can be understood by the software engineers and, therefore, used to develop the user-facing aspects of the system. In this case, we can see that requirements elicitation is not dissimilar to knowledge acquisition. The major problem within the knowledge acquisition and requirements elicitation domain are that analysts have difficulty acquiring knowledge from the user. This can often lead both domains into what has become known as analysis paralysis. This phenomenon typically manifests itself through long phases of project planning, requirements gathering, program design and data modelling, with little or no extra value created by those steps. When extended over a long timeframe, such processes tend to emphasise the organisational and bureaucratic aspects of the software development, while detracting from its functional, value-creating and human facing portion.

6.6.1 Optional Further Reading

- [A. D. Abbott] Methods of discovery: Heuristics for the social sciences (contemporary societies). WW Norton & Company, 2004.
- [S. W. Ambler.] The elements of UML 2.0 style. Cambridge University Press, Cambridge, 2005.
- [R. G. Burgess] In the field: An introduction to field research. Routledge, 2002.
- [J. Cato]. User Centered Web Design. Addison Wesley, 2001.
- [P. Loucopoulos] and V. Karakostas. System requirements engineering. International Software Engineering Series. McGraw-Hill, 1995.



Self Assessment Questions

Try these without reference to the text:

1. How would you go about getting the ‘what’?
2. Why are Post-its so important?
3. What is User Centred Design?
4. What do you do if there are no participants for a corporate system build?
5. What are the two main danger points to remember when undertaking UXD?

7. Modelling Requirements

In all cases you must think: what information do I need to build a system, how can I get this information, and how can I disseminate this information into the software engineering process, once acquired?

– Anon

There are some kinds of requirements documentation techniques that can be applied to the software development task. While the following is not a complete list, it does outline the kind of techniques that are often used and which normally provide adequate feedback from users. The conscientious UX analysts, however, should not rely only on these methods but should also include ones from the research literature such that a greater depth of understanding can be derived. In any case, the following list gives a subset of common techniques that both the UX analysts and the software engineer should know and understand.



Links to User Evaluation

Before starting, it is useful to point forwards to [User Evaluation](#). The collection and modelling of user data shares similarities and also some of these techniques can be used in the User Evaluation and vice-versa.

7.1 Informal Methods

The most common representation medium for informal systems is natural language; this is the case across organisations and across representational contexts. Forms are represented in natural language as are letters, memos, procedural documentation, and the like. Further, scenarios are often represented in natural language, and interviews with users and participants within the system's creation process are also conducted using natural language. Therefore, an analysis of the language used within these kinds of artefacts is eminently useful for requirements elicitation purposes. Practical methods of understanding requirements from a natural language analysis are by the use of nouns and verbs to represent actions and functionality. We can see that from an analysis of natural language, constraints, targets, owners, and actors can also be identified. Additional aspects of the functionality, which may be missed in general question-and-answer sessions with users, become self-evident when natural language analysis is applied to transcripts. In this case, the UX'er should link about understand the types of actions that are required and the interrelationships between those actions in a way that does not require the user to make changes in how they are discussing the requirements - and informal methods are very useful for this.

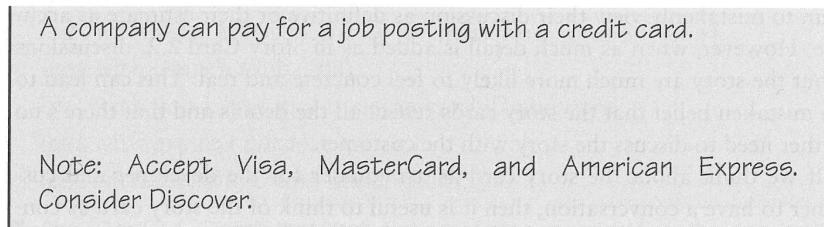


Figure: Story Card with Notes. Story Card with Notes Providing Additional Detail. —Image Credit: Cohn, 2004.

7.1.1 User Stories

User stories simply describe functionality that will be available to the users of the system [Cohn, 2004]. They are comprised of a written description, conversations about the story, details of the story, and a series of tests that enable us to understand if the document is complete and, therefore, the story is an accurate representation of the desired user experiences. User stories can be varying lengths, and these are determined by the way in which the story is created and summarised and also based on the purpose for which it is intended. In some cases, it can be the size of a full A4 sheet and for others it can fit onto two sides of a standard index filing card (see [Figure: Story Card with Notes](#)).

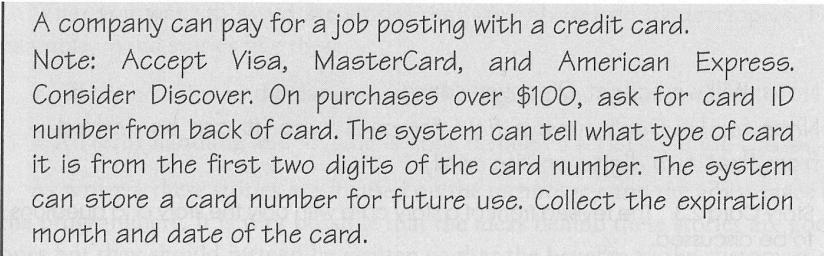


Figure: Story Card with Too Much Detail. Story Card with Too Much Detail. —Image Credit: Cohn, 2004.

The story is the visible part with the background information, which relates to the conversation between the customer and the developer of the story, being one of the most important aspects. The stories are often prioritised based on the value to both the organisation and the needs of the users.

Stories should be independent in that references to other stories of the other examples should be removed. In this way, the simplicity of the work can be embraced, and the story becomes easily understandable, in this way, developers are not required to investigate complex cross-referencing to understand the primary message of a story. Stories should also be negotiable in that they should not be thought of as fixed contracts but can be altered or modified based on new information or a negotiation with other stakeholders of the story. Obviously the story should be valued by both the user and the customer, and, besides, the story should be estimate-able. This means that the amount of time required to create the functionality to enact the story can be easily gauged by the developer even based on the developers lack of domain or technical knowledge. This really suggests that the story should be small enough for its development to be easily quantified (see [Figure: Story Card with Too Much Detail](#)). Removing complexity also

aids the estimate-ability of the work required to enact the story and so it may be useful to split stories into different parts to maintain their brevity (see [Figure: Revised Story Card](#)).

A company can pay for a job posting with a credit card.

Note: Will we accept Discover cards?

Note for UI: Don't have a field for card type (it can be derived from first two digits on the card).

Figure: Revised Story Card. Revised Story Card with only the Story and Questions to be Discussed. – Image Credit: Cohn, 2004.

Finally, the story should be testable; some developers suggest that successfully passing these tests prove that the story has been successfully developed, however for the UX analyst this is nowhere near the level of support that is required to suggest that success has been achieved, and comprehensive summative post-testing should also be carried out.

7.1.2 Use Cases

A use case differs from a user story because the user story is far less complex, shorter, and with an estimate-able outcome [\[Cockburn, 2001\]](#). A use case describes a system under various conditions and forms an implicit contract between the developers of the system and the commissioners of that system about the behaviour it will exhibit. Within the context of human-computer interaction, we can see that use cases can be effective when focused on human behaviour and the interactive environment. Indeed, use cases can also be applied to non-user facing parts of the system that makes them more effective. In this case, a unified language is built up such that the software engineers understand the use cases regardless of whether they are created by the UX'er, business manager, organisation analyst, information analyst, knowledge analyst, or any other of the number of specialisms that are required to develop the system. Use cases can be created to fill many different purposes and may take many different forms. As long as the form is well understood and all use case creators are consistent then there should not be any problem regarding the uniformity of understanding which is transmitted via these cases. Mostly, however, use case will consist of: the primary initiator, for example, the purchaser, claimant, or user; the use cases scope; its level of importance; and a list of stakeholders who may also be interested. Often success scenarios are also listed, and possible extensions to the system can be discussed (see [Figure: Tabular Use-Case](#)). The thing to remember about use cases is that they really are a requirement and should not require translation into a secondary format. However, be warned, they do not account for all requirements such as business rules, data formats, etc. However in the case of the UX analyst they serve a perfectly adequate purpose for the necessity of specifying user-facing aspects of the system.

Table 11.1. One-Column Table Format of a Use Case

USE CASE #	<the name is the goal as a short active verb phrase>	
Context of Use	<a longer statement of the context of use if needed>	
Scope	<what system is being considered black box under design>	
Level	<one of summary, primary task, subfunction>	
Primary Actor	<a role name for the primary actor, or a description>	
Stakeholder and Interests	Stakeholder	Interest
	<stakeholder name>	<put here the interest of the stakeholder>
	<stakeholder name>	<put here the interest of the stakeholder>
Preconditions	<what we expect is already the state of the world>	
Minimal Guarantees	<the interests as protected on any exit>	
Success Guarantees	<the interests as satisfied on a successful ending>	
Trigger	<the action upon the system that starts the use case>	
Description	Step	Action
	1	<put here the steps of the scenario from trigger to goal delivery and any cleanup after>
	2	<...>
	3	
Extensions	Step	Branching Action
	1a	<condition causing branching> : <action or name of sub use case>
Technology and Data Variations		
	1	<list of variations>

Figure: Tabular Use-Case. Possible Format for a Tabular Use-Case. –Image Credit: Cockburn, 2001.

In addition to the standard aspects of the use case, other information can be added such as preconditions that describe how the system should be before a use case commences; triggers that will initiate that use case; and guarantees that the use case will actually have been enacted correctly at the end. Use cases make use of the common software engineering scenario. However, it is normally created as a series of steps or bullet points as opposed to the standard scenario that is often more conversational. There are many different styles of use case ranging from formal, through casual, through to different styles created for different software engineering methodologies such as the Rational Unified process. Use cases may also take the form of if-statements and include extensions to common diagrammatic methods such as the Unified Modelling Language. The advantage of using this kind of modelling is that a software engineer will be familiar with UML. Therefore, there is less chance of misinterpretation than if the engineer were required to learn a new formalism.

7.1.3 Personas and Scenarios

According to Lene Nielsen, “*the persona method has developed from being a method for IT system development to being used in many other contexts, including the development of products, marketing, planning of communication, and service design. Despite the fact that the method has existed since the late 1990s, there is still no clear definition of what the method encompasses. A common understanding is that the persona is a description of a fictitious person, but whether this description is based on assumptions or data is not clear, and opinions also differ on what the persona description should cover.*” [Nielsen, 2013]

7.1.3.1 Personas

A Persona Is a fictional character created to convey a certain set of user properties and behaviours to a technical recipient, such as a software engineer (for example <http://elderlypersonas.cure.at/>¹). These personas are often created to flesh out the roles of actors, stakeholders, or proxies as we have discussed previously. In reality, personas are highly related to scenarios and user stories, however for a persona a set of user characteristics are created to describe the needs and requirements of that person². In this case, a persona can be quite short, or very deep (see [Figure: Expanded Persona](#)) – and there is much discussion as to how best to create them and what properties they should have (Yahoo! have some nice approaches to Persona development³). Personas are often created by user experts, the users themselves, or key informants to place the technical expert into the role of the user, allow them to understand the user better, and to convey information that may be lost in the reduction often applied to other approaches.

¹<http://elderlypersonas.cure.at/>

²Persona Example: *Mary* has a learning disability. She finds looking at images on a webpage very distracting. Mary would like to see all images rendered in the following order. First, for images with long descriptions have the long description rendered in place of the image. If the long description does not exist, she wants the ‘alt text’ to be rendered. If neither is available, Mary wants the file name rendered. Added functionality would allow Mary to right click (context menu) on an image to list and select the rendering of the available alternatives (thumbnail, original size, full screen, low resolution, high resolution, alt text, long description, file name). *Juan* is hard of hearing. He wants always to see a video on the page. Also, Juan would like the Spanish language track used if available, along with Spanish captions as a default. If these are not available, he wants to see the video with English audio and captions. If no captions are available, Juan wants the video and English audio. Added functionality would allow Juan to right click (context menu) on a video to list and select the rendering of the available alternatives. (W3c WAI UAAG Version 2).

³<http://www.uie.com/brainsparks/2006/05/18/yahoos-approach-to-keeping-personas-alive/>

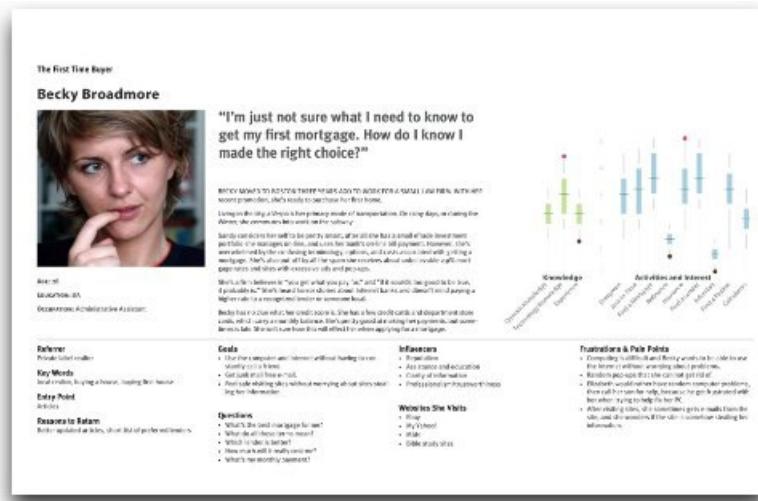


Figure: Expanded Persona. Todd Zaki Warfel's Persona Description With His Persona DNA Chart. – Image Credit: <http://www.uie.com>.

7.1.3.2 Scenarios

A Scenario is a detailed description of the user's interaction with the computer; these scenarios are not the same as those found in a use case because they are often larger and more encompassing than a use case, and are written in plain English as opposed to a bullet point. Within a standard scenario, there may be many actors within the system and many interrelated stories often criss-crossing throughout the scenario. This means that they do not conform to either the user story that requires there be a level of independence within the story or to the use cases because they do not separate out the stories into well-defined task requirements. In reality, a scenario represents a detailed definition of a specific instance of interactivity. Scenarios are very flexible and narrative in nature which means that they are easier to create than the more formalised user story or use case. Indeed, most scenarios are created from direct conversations with users and stakeholders of the system, and they are then transmitted for modelling without much additional modification. Often scenarios can be used to clarify issues and implications of system development when there is no other way to do so.

7.2 Semi-Formal Methods

Methods don't neatly fit into informal and formal categories - there are of course semi-formal methods - and in reality most methods sit on a spectrum. You can, however, decide where you think a method fits based on its reproducibility - and in the case of very formal methods their ability to generate code. The following methods sit on this border, but could of course fit into either; to a greater or lesser extent.

7.2.1 Wireframes

Wireframes are simple, often hand drawn, sketches of the proposed user interface (see **Figure: Calendar Wireframe**). In this case, wireframes can be created with the user at

the time a requirements capture discussion is taking place so that the UX'er can make fast and direct corrections and developments to that interface – based on immediate feedback from the user. By drawing these wireframes freehand the technology barrier between the user and the UX'er are reduced, and there is a greater immediacy to the capturing of these requirements. Wireframes may also be annotated using bubbles and arrows that may normally point to interface components such they can be explained better.



Figure: Calendar Wireframe. —Image Credit: The Thought Balloon.

As well as the immediacy of the hand drawn wireframe it is sometimes appropriate to create a computerised wireframe if this is being related to a different set of stakeholders, such as those responsible for the business requirements of an organisation – as opposed to the users directly – in reality this means for presentation to higher management.

7.2.2 Mock-ups & Storyboards

Mock-ups & Storyboards are just extensions of wireframes, but with more flesh on the bones. Again both mock-ups and storyboards may occur as hand drawn sketches, or as computer-generated interface elements. These mostly don't have the ability to run real code – being presented mainly for their visual look and feel – indeed, both mainly sit between the wireframe and the 'prototype'. As with wireframes, mockups are mostly a singular interface screen or component (or loosely-related series of screens) which can then be discussed (see [Figure: iPhone Application Mock-up](#)).

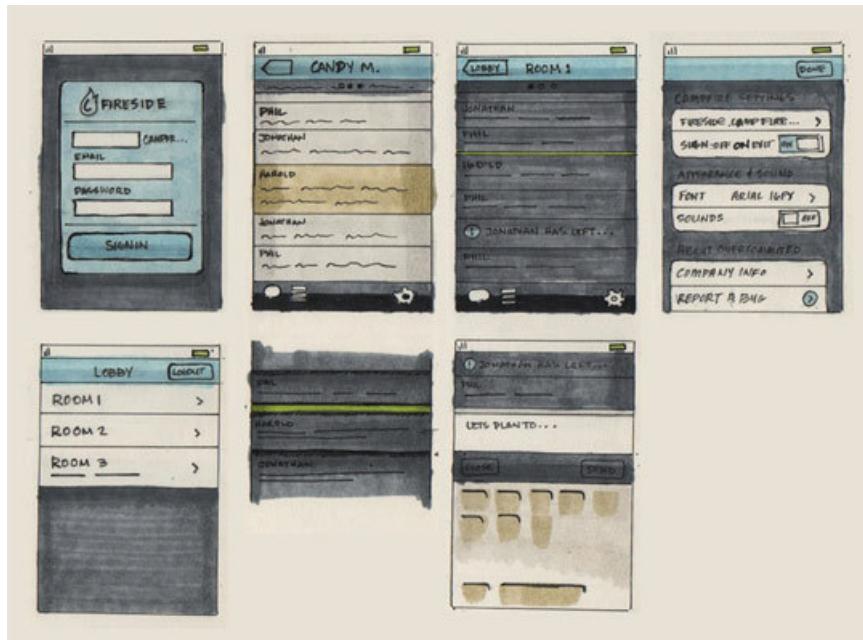


Figure: iPhone Application Mock-up. iPhone Application Mock-up, Early Ember Sketches. –Image Credit: Web Design Ledger.

A storyboard (see [Figure: Post-It Storyboard](#)), on the other hand, adds interactive elements and interaction design to wireframes or mockups – showing the outcomes of certain user selections. These are mainly very simplistic interactions – being all that can be modelled with pen and paper – and do not run an interactive code. You often see a storyboard running top left to bottom right in a cartoon style, or in a flowchart-style whereby following different arrows has different effects.

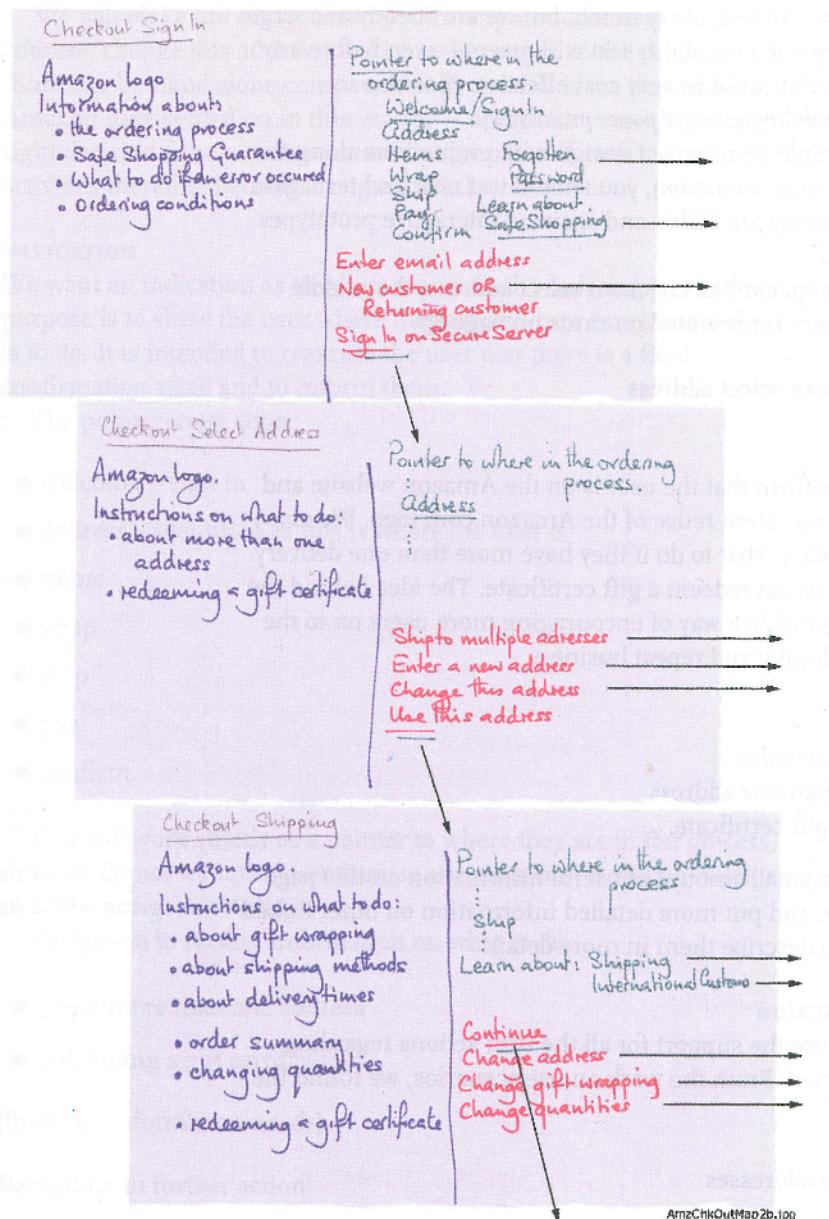


Figure: Post-It Storyboard. Example Post-it Interaction Storyboard. —Image Credit: Cato, 2001.

7.2.3 Priority Poker

Priority (Planning/Scrum) Poker is a design game for collaboratively prioritising interface / UX specific items. Simply select the items that you think need to be prioritised and using a UX card deck (see [Figure: Priority UX Poker Card Deck](#)) for each person place your cards face down for each. Once there are no cards left, upturn and add them up, order the results gives the priority for development; highest first. You can then use these in an agile style MoSCoW prioritisation or MoSCoW⁴ analysis:

- M - MUST: Describes a requirement that must be satisfied in the final solution for the solution to be considered a success.

⁴MUST, SHOULD, COULD, or WON'T.

- S - SHOULD: Represents a high-priority item that should be included in the solution if it is possible. This is often a critical requirement but one that can be satisfied in other ways if strictly necessary.
- C - COULD: Describes a requirement that is considered desirable but not necessary. This will be included if time and resources permit.
- W - WON'T: Represents a requirement that stakeholders have agreed will not be implemented in a given release, but may be considered for the future. (note: occasionally the word "Would" is substituted for "Won't" to give a clearer understanding of this choice)

UX priority poker, is a variant of planning poker (see [Figure: Priority UX Poker Card Deck](#)), also called Scrum poker, and this is why it fit nicely into the MoSCoW analysis model. Again, this is a consensus-based technique for estimating software development effort in user stories. As in priority poker, members of the group make estimates by playing numbered cards face-down to the table, instead of speaking them aloud. The cards are revealed, and the estimates are then discussed. By hiding the figures in this way, the group can avoid the cognitive bias of anchoring⁵. Neil Turner describes priority poker's benefits in use as:

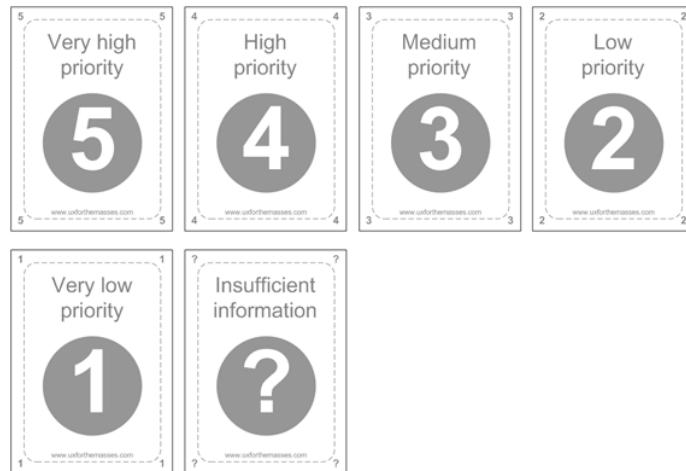


Figure: Priority UX Poker Card Deck. Priority UX Poker Card Deck. —Image Credit: <http://www.uxforthemasses.com/>.

- During requirements gathering and analysis to prioritise goals and objects; functional and non-functional requirements; and to prioritise features.
- During design to prioritise personas, scenarios and user journeys.
- Following usability testing and expert reviews to priorities usability issues and recommended changes.
- During testing to prioritise bugs, defects and fixes.
- Within Agile projects to prioritise user stories and the product backlog.

7.3 Formal Methods

Software requirements are described in the standard IEEE 830-1998 as “A *software requirements specification (SRS)* is a complete description of the behaviour of the system to be developed. It includes a set of use cases that describe all of the interactions that the users will have with the software. Use cases are also known as functional requirements. In addition to using

⁵Anchoring or focalism is a cognitive bias that describes the common human tendency to rely too heavily on the first piece of information offered (the “anchor”) when making decisions —Wikipedia.

cases, the SRS also contains nonfunctional (or supplementary) requirements. Non-functional requirements are requirements that impose constraints on the design or implementation (such as performance requirements, quality standards, or design constraints).”



Figure: Planning Agile (SCRUM) Poker Card Deck. Planning Agile (SCRUM) Poker Card Deck. –Image Credit: Wikimedia.

Within the context of modelling user requirements (requirement specification), our choices of methodology become limited. Some elicitation practices such as user stories and use cases have an endpoint that includes the specification of requirements and the modelling of the user-facing aspects of the system. However, this is not the case for all requirements elicitation processes and indeed, even for those who include

their own specification, it may be necessary to translate from these to a format that is more readily understandable by a software engineer. In this case we will study modelling languages based on well-understood engineering techniques and focusing on the universal modelling language, state transition diagrams, and flow diagrams: flow charts, data flow diagrams and control flow diagrams.

7.3.1 Flowcharts

Flowcharts, and diagrams that describe the flow of actions, control, and information are an old, tried, and trusted form of conveying information about the way the system should behave concerning its users and the processes that those users enact [**Loucopoulos and Karakostas, 1995**]. While the flowchart has some detractors and is seen by some as an old form of modelling, you can guarantee that an examination of any whiteboard in any software engineering company will most likely have flow charting components written upon it. A flowchart is a common type of diagram and represents the system behaviour as a series of boxes with inter-connecting arrows indicating transitions between steps. These boxes have a specific order, and each one has a specific shape such that a certain action can be described; for example a diamond-shaped box represents a choice to be made, a cylinder represents a data storage device, etc. Each box has text written within it which describes either the process, choice, or artefact. Flow is represented by the adjoining arrows, and there may be return flows as well as standard waterfall type cascades. Flowcharts are used in analysing, designing, documenting or managing a process or program in various fields (see [Figure: Data Flow Model](#)). In addition to the standard flowchart, there are two types of diagrams which the HCI analyst or software engineer should be familiar with and these are the data flow diagram and the control flow diagram.

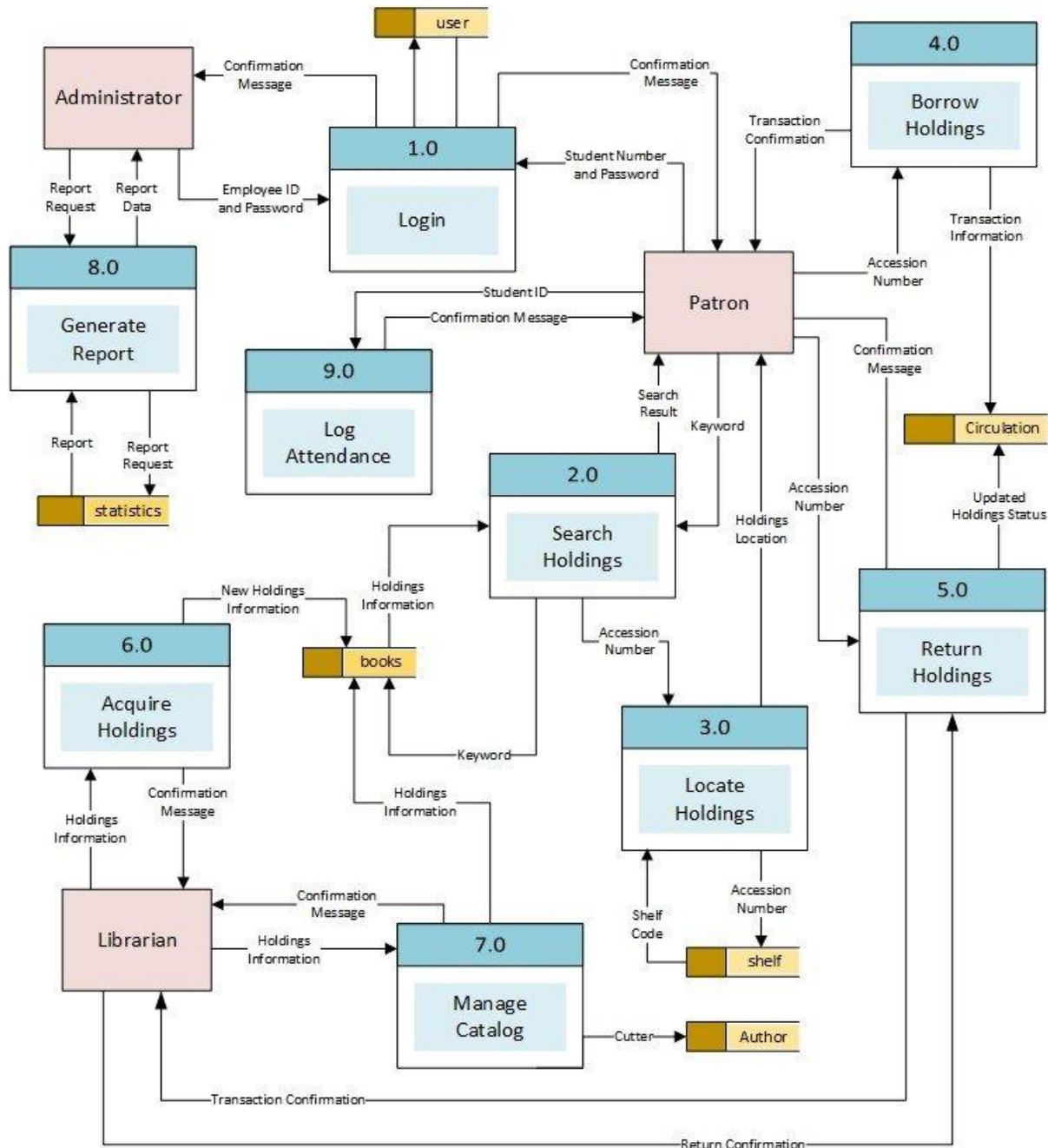


Figure: Data Flow Model. An Example of a Data Flow Model. —Image Credit: Manuel B. Garcia 2019.

The data flow diagram (or model) is instrumental in the waterfall-based SSADM methodology and represents the flow of data through an information system in a graphical format. In this way, the data processing aspects of the system can be better understood, and a detailed analysis of the way information moves around the system, including its storage, can be derived. This is important for the UX'er because an understanding of how information moves, along with the kind of information that will be required by the system, indicates the type of inputs that will be required from the interface and thereby implies a certain interaction and behaviour that will be exhibited by the user. The data flow diagram provides no information about the timing or ordering of processors, or whether processes will operate in sequential or parallel, this differentiates it from a

standard flowchart.

The control flow diagram represents a sequence of subdivided steps often representing well-known programming concepts such as if-then-else conditions, repetition, and/or case conditions. Annotated boxes and shapes are used to represent operations, data, or equipment, and as with the standard flowchart, arrows are used to indicate the sequential flow from one to another. There are several types of control flow diagrams used: configuration decision control flow diagram are used in configuration management; quality control flow diagram are used in quality control; change control flow diagram are used in project management; and process control flow diagrams are used in the software engineering practice. Obviously the last two types of flow chart diagram, change and process, can be used and adapted in the UCD process and are useful tools for understanding behavioural aspects of the software engineering system in the large. In software and systems development control flow diagrams can be used in control flow analysis, data flow analysis, algorithm analysis, and simulation. Control and data flow analysis are most applicable for real time and data driven systems. In reality these flow analyses, transform logic and data requirements text into graphical flows that are easier to analyse, and convey to the software engineer, than the text itself.

Each of these different flow diagramming techniques are useful in-and-of themselves, however, in combination the flowchart, data flow diagram, and control flow diagram of very effective in conveying human facing aspects of the system in the context of the input and output requirements of that system.

7.3.2 State Transition Diagram

A state transition diagram, also known as a state diagram, is different from flowcharting because each node of the diagram represents a specific system state, not a fact within that system, and is used to describe the behaviour of a system. This behaviour is analysed and represented as a series of events that could occur in one or more possible states. State transition diagrams require that the system be composed of a finite number of states which in some cases are true and tangible states and at other times are more abstract. Diagrams are made up of nodes and arcs (circles with connecting arrows). Each node is divided in half with the state of being written in bold within the top half, and the entry action written with in the bottom half. This entry action specifies the state that must currently exist for this new state to be entered. The directed arrows represent transitions between states, and these are all labelled with the transition condition. There are many forms of state diagrams which differ slightly and have different semantics. One such possible representation is the state transition table.

State transition tables are normally two-dimensional, most commonly with a series of events denoted as the columns and a series of states denoted as the rows. At the intersection of the state and event, a specific action is denoted along with the new state that will be reached if that action is undergone. A second, but a less common way of denoting the table, is that the next state is denoted within the column and the current state is denoted within the row. The intersection of these two states has the action, and the event placed within it.

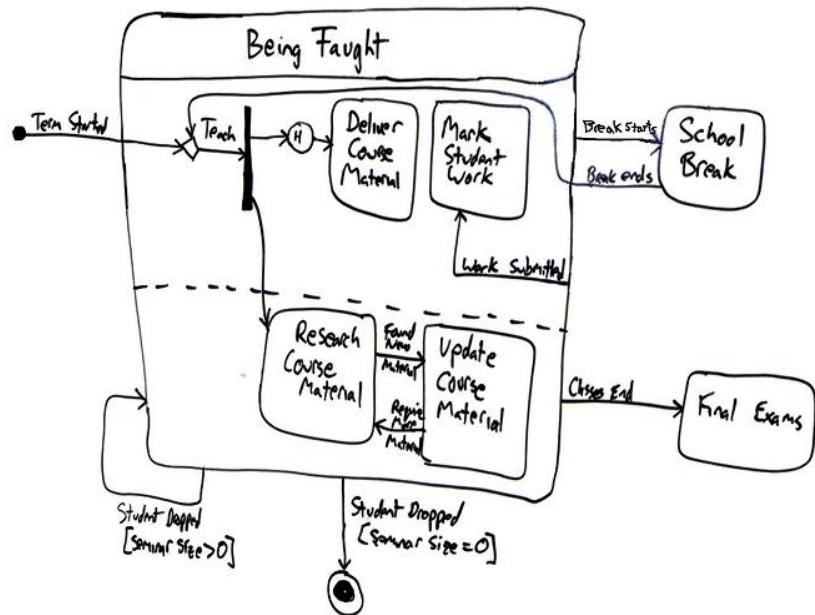


Figure: State Transition Diagram. An Example of a State Transition Diagram (or State Machine). –Image Credit: <http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm>.

State transition diagrams and tables are used to specify any behaviour a system may exhibit (see [Figure: State Transition Diagram](#)). This may be changed within the program logic or within the actual user interface itself. Within the context of the UCD, state transition diagrams and tables can be used to specify the user-facing aspects of the system and the states that the interface should move between as the user's interactive behaviour involves. It is therefore very easy to understand how the system should be left after specific events have occurred and, therefore, it is easy to test if those events and conditions have been met in a rigorous and specific way. This kind of technique also dovetails into the natural engineering processes that will be familiar to most software engineers and means that it is a powerful technique for conveying interactive behaviours between the UX domain and the software engineering domain.

7.3.3 The Unified Modelling Language (UML)

UML was developed in the mid-1990s to overcome the problems with other diagrammatic representations of object-oriented software [[Ambler, 2005](#)]. Since then it has to evolve and become the most popular modelling language within the software engineering process and has been extended from its initial object-oriented focus to enable an entire system to be modelled. It includes primitives to represent use case diagrams, class and package diagrams, sequence and state machine diagrams, communication, components, and deployment diagrams along with activity diagrams and diagrammatic conventions associated with database creation and relational modelling. In this case, it is highly likely that any software engineer creating modern applications will understand the modelling language, and what is more there are many tools for describing UML models and some tools that generate software from these descriptions. In this case, UML models may be automatically transformed to other representations (e.g. Java) using Query/View/Transformation QVT-like transformation languages. UML is a de facto industry standard and is evolving under the auspices of the Object Man-

agement Group with many industry leaders helping create the standard. Finally, UML is extensible, offering profiles and stereotypes as the main mechanisms for customisation.

Many books have been written describing UML (and there are many different diagrammatic notations, see [Figure: UML Diagram Collage](#)), all running into hundreds of pages, and so attempting a complete description here would be ineffectual. However, a brief overview of the diagrammatic conventions specifically focused to the user-facing aspects may be worth considering; in this way you will be able to read at least these diagrams and have a certain level of understanding as to their meaning. The most common UML modelling element is called the class diagram and looks like a rectangular box divided into three separate sections, at the top the name of the class is given, in the central part each variable is listed, and in the bottom part the methods that work on those variables are described. Next to each definition you may see a plus symbol (+) to describe public access, a hash symbol (#) describes protected access, a minus symbol (-) to describe private access, and a tilde (~) to describe a package. A diagram resembles many these rectangular boxes linked together via directed arrows with additional diagrammatic primitives of a stick man representing an actor. A circle with an arrow at the top representing a process/controller class, a circle resting upon a horizontal line representing a domain class, a circle with the letter T placed at the ‘nine o’clock position’ representing an interface class, and finally a horizontal line representing an association. Notes can be added to diagrams in the form of a rectangular box with the top right corner folded over, and a UML frame, which is a bounding box, can be used to logically group aspects of the diagrammatic components together. Finally, the diagramming primitive called a stereotype is signified by its name being written in lowercase between double greater than and less than symbols (<>). The stereotype primitive is used to denote a variation on an existing modelling element, with the same form, but with a modified intent; in object-oriented languages these can often be used to signify inheritance classes.

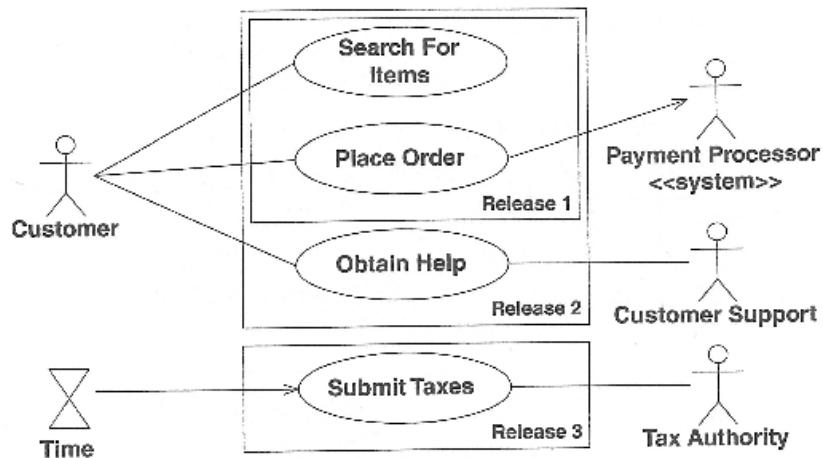


Figure: UML Use Case Example. Online Shopping UML Use Case Example. –Image Credit: Ambler, 2005.

The most important aspect for the UX'er is the UML use case diagram (see [Figure: UML Use Case Example](#)). Use case diagrams show relationships among actors and use cases within a system. They provide an overview of usage requirements, communicate the scope of a development project, and model the analysis of usage requirements in

the form of a system use case model. Use cases names are often begun with a strong verb such as ‘open’, ‘close’, ‘save’, ‘deposit’, or ‘withdraw’, etc. Moreover, are normally represented by an actor being linked to a series of oval ‘boxes’ containing the use cases. Use cases that are stacked in a vertical manner imply a sequenced top to bottom. Use cases can be enclosed in rectangles to imply a logical grouping; an actor should be drawn outside of these use case grouping elements. An actor can be associated with one or more use cases, and a single use case can be associated with more than one actor. Associations may exist between an actor and a use case and two use cases, and a generalisation can likewise exist between two actors, or between two use cases. Stereotypes are represented using broken lines to make these connections, with the standard stereotype convention being written next to that line. Also, conditions can be placed upon the associations, and these are denoted by joining that association with a standard note. There are very many more aspects of the mapping of use case diagrams in UML than can be described here. However, this brief synopsis should help you get started and enable you to derive some understanding from informal diagrams you may come across.

As we have seen, more formal modelling can be an important part of the UCD/-software creation process, particularly when trying to convey rigorous information regarding user-facing aspects of the system. This is mainly because this kind of information is difficult to quantify especially because users are notoriously flexible and inconsistent in their approaches to applications and systems use.



Caveat

Alan Dix gives us a very real warning as to the dangers of mediocracy and ‘Group-Think’ in his 2010 paper [Dix, 2010] “Imagine you have a group of children and want to give them lunch. In the UK, you might well choose baked beans. Not the most exciting choice but few children actively dislike baked beans; they are acceptable to everyone. However, give each of those children a euro (or maybe two) in a sweet shop... they will all come away with a different chocolate bar, the chocolate bar that is ‘OK’ for everyone gets chosen by none. Much of traditional HCI design is like baked beans - a word processor installed for the whole company, a mail program used by every student, good enough for everyone. However, increasing personal choice, especially for web-based services, makes the design more like the chocolate bar; different people make different choices, but what matters is that the product chosen is not ‘good enough’ for all of them, but best for some.”

I’ve also covered this aspect of design [Harper, 2007] and Dix has a very good way of phrasing it. I think that the salient point here is that systems run in a combinatorial way, where as individual findings can exist with minimal confounding variables muddying the waters. Different operating modalities are useful for providing a personalised experience, and a one-size fits all (consensus) approach is not always the best way forward. Indeed, if I were to play ‘devils-advocate’, I’d wonder if universal design or participatory design just encourage a product that is acceptable to all but desired by none – expanded upon later); it is something you should think about.



Cautionary Note

Testing mostly occurs to see whether the developed system meets the specification set out in the models. If this is the case, many software engineering practices declare that the system has been successfully developed for the user. In this case, there is obviously an over-reliance on the assertion that correct requirements elicitation and analysis have been undertaken, and that this process has enabled the model to be developed correctly.

At this point then, it is only the model that is being tested as opposed to the user-facing aspects of the system along with the user interface after the entire system has been developed. The UX'er should pay particular attention to this cautionary note. If, at all possible, you should mitigate this over-reliance by testing the final system with users regardless of the conformance of the software to the UCD you have created.

In effect, you must differentiate between the design you have created and its creation in software. By declaring success, that your software meets your design specification, you do not allow for errors in the design – or its translation into the software.

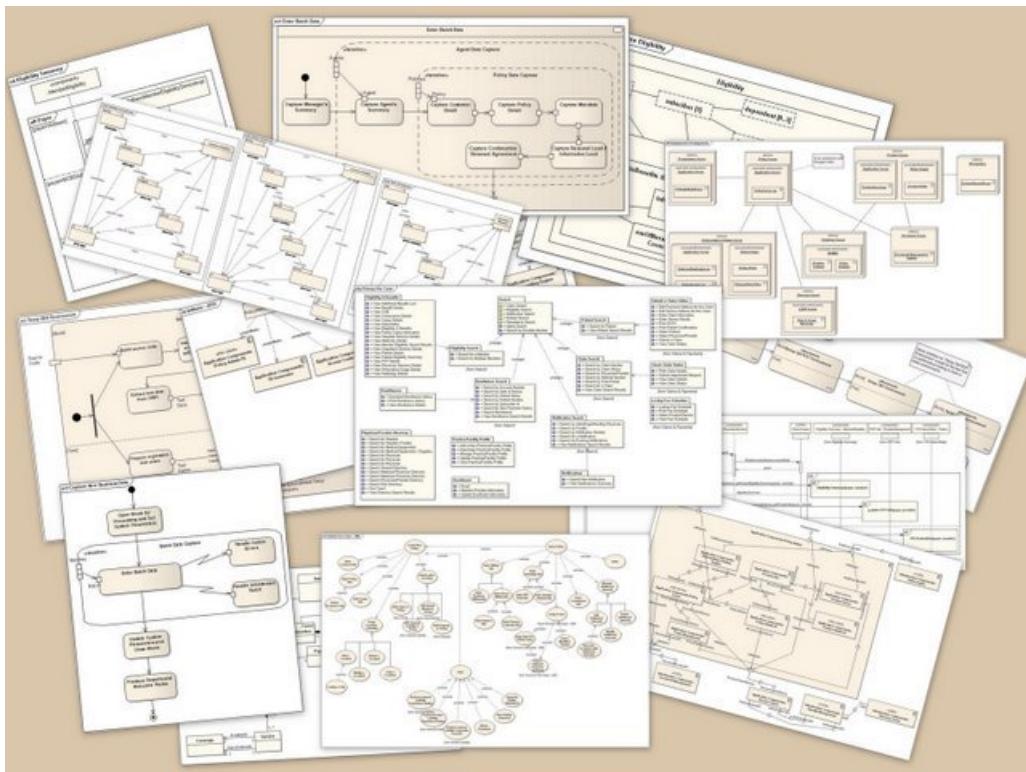


Figure: UML Diagram Collage. UML Diagram Collage. —Image Credit: Wikimedia.

7.4 Summary

This chapter has highlighted the methods of communication to enable a formal dialogue to be entered into between users, UX'er, and the software engineer. In this way, the needs of the users can be accurately communicated between both parties.

Interestingly, interaction design is at the forefront of Web development because there is no common interface or set of interface tools applicable to the many different needs of the Web user. In this case, it can be seen that the Web provides a blank canvas for the interaction designer and enables many different design and interaction modalities or schemas to be applied. This is in stark contrast to a standard desktop applications that force the user into certain standardised practices as discussed above.

Designing the interaction is usually accomplished in combination with user experience specialists as well as graphic designers and interface engineers. Because this is an interdisciplinary process, there are very few fixed modelling techniques for designing these interactions. Often the most basic methods are used which would apply to all, and in many cases whiteboards and chalkboards are used to discuss possible layouts with post-it notes used to order functionality and design and events within the framework. Storyboarding techniques may also be used to relate better the intended interactions, and the graphic designer may flesh out these models with some initial ideas about how the interface should look or be skinned.

Remember, “*In all cases you must think: what information do I need to build a system, how can I get this information, and how can I disseminate this information into the software engineering process, once acquired?*”

7.4.1 Optional Further Reading

- [B. Buxton] Sketching user experiences: getting the design right and the right design. Morgan kaufmann, 2010.
- [A. Cockburn.] Writing effective use cases. Addison-Wesley, Boston, 2001.
- [M. Cohn.] User stories applied: for agile software development. Addison-Wesley, Boston, 2004.
- [R. Krum] Cool infographics: Effective communication with data visualization and design. John Wiley & Sons, 2013.
- [B. Shneiderman] Designing the user interface: strategies for effective human-computer interaction. Addison-Wesley, Reading, Mass. 1987 (this is a famous book now in its fourth edition)

7.4.2 International Standards

- [ISO 9241-210:2010.] Ergonomics of human-system interaction – part 210: Human-centred design for interactive systems. TC/SC: TC 159/SC 4 ICS: 35.180; 13.180 / Stage: 60.60 (2010-03-03), International Organisation for Standardisation (ISO), Geneva, Switzerland, 2010.



Self Assessment Questions

Try these without reference to the text:

1. What are the differences between informal and semi-formal methods?
2. Pick an informal methodology and describe it?
3. Pick a formal methodology and describe it?
4. Elaborate the 'Cautionary Note'.
5. Differentiate between Personas and Scenarios

Part III: Building the User Experience

8. Developing for UX

Happy developers are productive developers; productive developers produce beautiful code as well as beautiful interfaces.

– Anon.

User Experience is an umbrella term used to describe all the factors that contribute to the quality of experience a person has when interacting with a specific technical artefact, or system. It focuses on the practice of requirements gathering and specification, design, creation, and testing, integrating best practice, heuristics, and ‘prior-art’ whereby the outcomes can be qualitatively evaluated using small numbers of users, placing humans firmly in the loop.

In my opinion, the user experience process starts with the ‘User Experience Engineer’ as end-user. By this I mean that the user experience of the UX engineer is often ignored; as is the user experience of software engineers who may also assist in the creation of the artefact. But it is critical not to miss out aspects of the software engineering process. Start with the right tools that fit both the development and the developer. This rationale extends to the methodology/lifecycle used, and the ethos of the development in general [Martin, 2011; 9241-100:2011, 2011].

8.1 UX Development

User Experience departments look different in different organisations and corporations. In some, the UX engineer is responsible mainly for the interactions with the users both before the development begins. Also in the requirements gathering phase, and after the development is ‘finished’, in the testing and evaluation stage. However, in some departments the separation may be more flexible, and you may also be tasked with building parts (or mock-ups) of the interface and interactivity yourself.



My picks are:

- Methodology – Agile Development (if we know the end point) / Cowboy Coding (if we don’t know the ned point)
- Separate Concerns: Microservices or REST (Service lead)
- Interfaces: Use a toolkit and skin-it. Don’t create your own!

If you are lucky, you may be part of a ‘Skunkworks’ team or an organisational research department. Where you will be part of a small and loosely structured group of people

who research and develop a project primarily for the sake of novel innovation¹. The skills-set you will be required to use in each of these situations are different. However, your training should allow you to undertake any².

Of these three scenarios, the one that requires the most careful navigation and communication is the first. Here you are responsible just for the human facing aspects of the work and are seen as the liaison between the users and the software engineers. This requires good communication skills and the ability to understand complex technical issues while translating these to more simple concepts for user dissemination.

In this scenario, where you are not allowed to develop the software yourself, it is difficult to overstate the importance of the general software engineers. Those who are building the system functionality and by implication, your interface and interactivity. If these technical staff are not ‘on your side’, then your project is likely to fail. The difference between a good software engineering and bad software engineering is critical. Even more critical for a good user experience we would expect much of the complexity of the interface to be removed to the underlying software functionality. Or generate underlying software functionality to enable a more complete and interesting user interaction.

Let me illustrate with an example, in this case, the development of the Apple Newton in the early 1990s. This system was widely seen at the time as having an excellent interface (see [Figure: Apple Newton UI](#)). But the main reason this interface was so powerful was because all applications could share data between them. This meant that writing a name in your calendar would automatically link to that name in your contacts list and bring through relevant information on addresses and telephone numbers etc. into the calendar application. This was possible because of the farsightedness of the underlying software engineering in which data was stored as ‘soups’. Soups are a very simple flat database in which the specification of the data is stored with that data. In this case, all applications can delve into the different soups created by different applications, using their data at will; termed ‘cross soup compatibility’. Without this underlying, but often invisible, mechanism the user experience would have been much reduced.

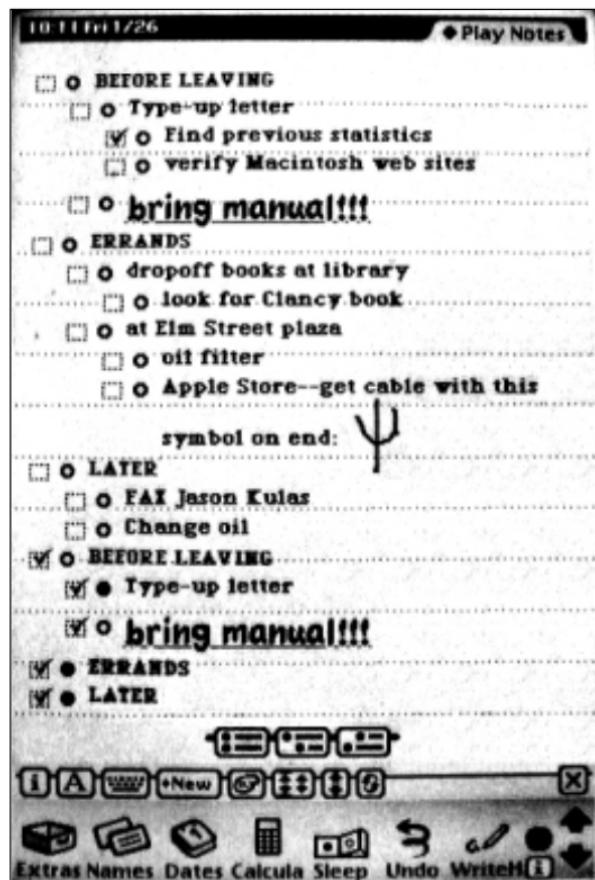


Figure: Apple Newton UI. Apple Newton UI –Image Credit: Wikimedia.

such as Microsoft's - now defunct - Pioneer Studios.
²We'll discuss these real world implications more fully [later on](#).

The point I'm trying to make is that the underlying functionality is as much a part of the user experience as the interface (either virtual or physical) itself. The software engineer can bring this level of innovation to a UX project, but this takes good communication, good involvement, and often the opportunity for the engineer to build elegance³ into the code base [**Oram and Wilson, 2007**].

Finally, it is not my intention to describe the kinds of tools a developer should use, these are often personal choices or defined by the organisation in which they work. A developer's choice of tool often makes more sense to the developer than to anyone else. Indeed, I have witnessed many arguments between developers as to the relative benefits of code generation using; say, the Eclipse Framework as opposed to NetBeans (see [Figure: Netbeans Debugger](#)), or vice-versa.

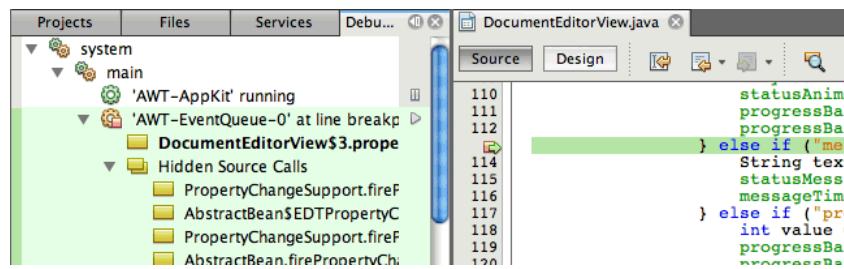


Figure: Netbeans Debugger. Netbeans Debugger –Image Credit: Sun/Oracle.

What is more important is the development methodology and the development ethos which should support the kind of interactions and feedback naturally provided by users.

8.2 Development Methodologies and Lifecycles

It is often the case that the UX engineer will need to fit into the design methodology and lifecycle that has been created by the project managers. In this case, there is little one can do to alter or modify this decision and all that remains is to understand the advantages and disadvantages of that lifecycle within the context of the UX work. Indeed, the decision as to the methodology and lifecycle chosen is often very personal and may also be based on the experience of the project manager's previous successful use of that methodology. It may also rest on the desires of the organisation and the needs that they may have concerning maintenance and dissemination within the organisation at a later stage.

In this case, there are a number of models of the development lifecycle accompanied by a number of engineering methodologies that conform to aspects of these models [**Sommerville, 2011**]. But, these are often more specific about the tools and techniques that are used in the different stages of the underlying model than we may like. I do not suggest that the following descriptions are in any way exhaustive. They should give you an initial understanding of the types of development that are often undertaken. As well as an overview of some of the most popular methodologies that are used in

³Indeed, code with is open to the scrutiny of peers (such as open-source code) is often found to be more elegant and less error prone than code that is hidden from scrutiny.

common software engineering practice. Let us start with the four most common models currently in use.

8.2.1 Waterfall

The worldview of the waterfall model is that each aspect of the development lifecycle occurs in a tight and rigidly strict order. Here which each phase must be fully completed before the next can begin (see [Figure: Waterfall Methodology](#)). In this case, imagine a series of phases starting from the top left and descended diagonally to the bottom right. Normally we would start in the top left with requirements and finish in the bottom right with maintenance. Usually, there are phases of design, implementation, and verification that make up the remainder of the waterfall development lifecycle. However, these can vary and be extended based on the specific kind of waterfall model that is being used.

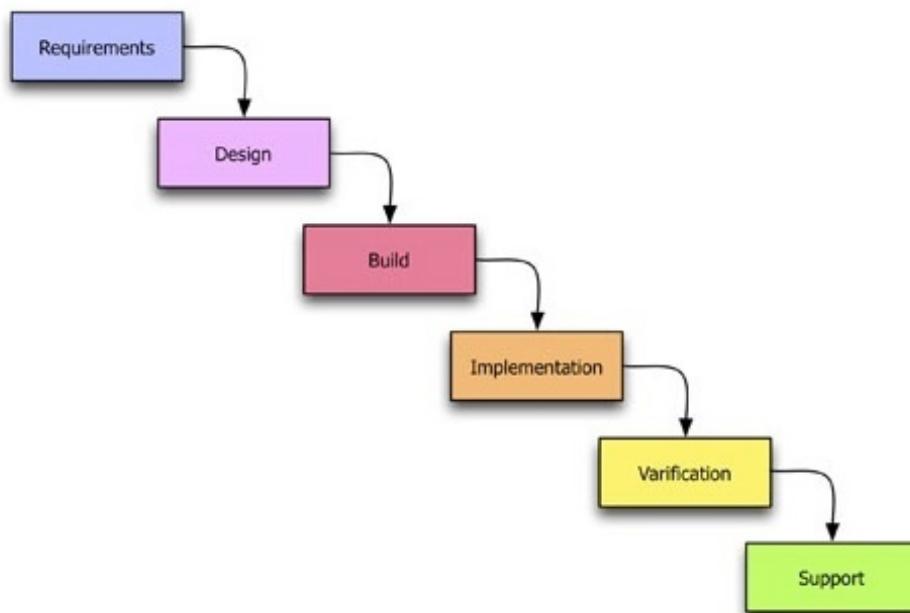


Figure: Waterfall Methodology. Waterfall Methodology –Image Credit: Linear Blue.

In this way requirements inform design, design informs implementation, implementation informs verification, and finally, verification informs maintenance. In some cases, there is a feedback that runs from the last phase such as maintenance or verification directly up to requirements. This is used to show that there is some cyclical behaviour between the elements of the waterfall model. In my personal opinion, this is the worst way of doing human factors work because each part of the human-centric development is confined to each specific lifecycle phase. This means that it is very difficult to feedback and change the requirements of the human facing aspects of the project to fulfil the user needs and their changing requirements. Or indeed redesign to remove any development miss-perceptions of those user requirements. In reality, it is my opinion that the waterfall model is far more suited to rigid software engineering. Whereby the key focus is accomplishing the project plan such that the contract between the commissioner and the developers is filled on time and to budget. The user

requirements are only valuable in that they are met as part of the formal development specification regardless of their applicability to the end user. It is my opinion then that the waterfall model is far more targeted to contractual fulfilment than system usability. I would also suggest it is very difficult to ad-hoc modify the waterfall methodology to suit the requirements of the user experience ‘process’ - try not to use this model if at all possible. If you have no choice, try to iterate requirements and design as much as your organisation will allow - and suggest from the outset that they link verification to implementation; it will be a tough sell!

8.2.2 Spiral

The spiral model is different to the waterfall model in that it splits up each task into some tasks that are iteratively undertaken. In this way, the worldview of the spiral model is that each aspect of the development lifecycle requires multiple iterations before proceeding to the next phase. Each phase requires the determination of objectives, the identification and resolution of risks, development and testing, and finally planning for the next iteration. The spiral model is named because the development track can be drawn as a spiral emanating from a central point. To visualise this, imagine that a piece of paper is divided into quarters with a central point being the start of the development lifecycle (see [Figure: Spiral Methodology](#)). The spiral tracks through each quarter starting from the top left in a clockwise manner represent each of the four stages of the iteration phase (objectives, risks, development, plan).

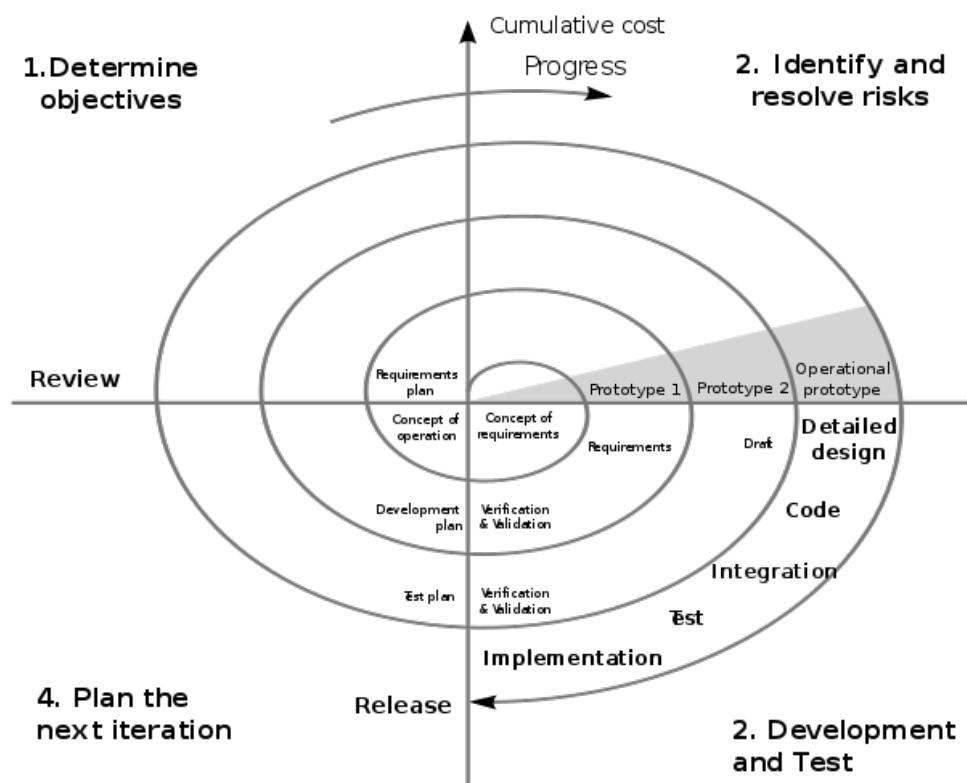


Figure: Spiral Methodology. Spiral Methodology –Image Credit: Wikimedia.

As the development tracks spirals out it is annotated with the larger aspects of the

development lifecycle. These are aspects that will be required at the time such as requirements, design, development, validation and testing, and maintenance, similar to the waterfall model. However, the iterations are tighter at the centre when designers are gathering the requirements and designing the system and become longer as they spiral out towards the edge where we arrive at detailed design coding and implementation. In the context of UX development, this is a better model than the waterfall model described previously. This is mainly because it acknowledges that some iterations will be required for each of the phases. And it also acknowledges that these phases may need to be redeveloped based on new information uncovered in different parts of the evolving development. Not perfect for the UX process but acceptable without much ad-hoc modification!

8.2.3 Iterative

The worldview of the iterative model is different than the waterfall or the spiral models. This is because it sees software development as a circular process comprising requirements, analysis and design, implementation, testing, evaluation, and then planning for the next iteration. This means that it sees the software engineering lifecycle as iterative and cyclical as opposed to a rigid sequence of phases that have no impact on each other once the development is underway (see [Figure: Iterative Methodology](#)).

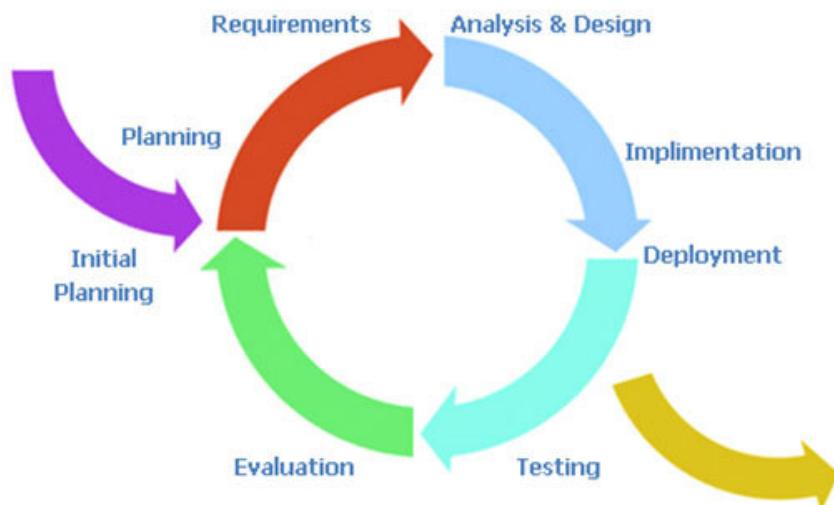


Figure: Iterative Methodology. Iterative Methodology –Image Credit: Wikimedia.

The iterative system is very easy to understand. All the user needs to do is imagine a circular process running in a clockwise direction with each of the main phases of the development equally spaced. An initial planning phase feeds into the initial entry point before the requirements elicitation is undertaken, and an exit point is available after implementation and signifies deployment. In reality, the deployment exit could be linked to the initial planning entrance to signify the continual maintenance the system will undergo. For UX development, this is my second most favourite model because it takes into account the impact that the user testing will have on the next iteration of development and planning. It supports the view that, from a user perspective, we cannot understand all aspects of the user experience before some development, even

prototypical development, has been undertaken. In this case software that is far more suited to the user is created and modifications can be made to refine the design on subsequent iterations. The major drawback of the iterative approach is that the endpoint can sometimes become lost in the focus on an ever more refined software artefact. In this case, the software development manager must make sure that they have some understanding of the number of iterations they expect to occur for any specific development. Making sure to plan time for an extra one or two iterations to account for unforeseen circumstances. A good choice for the UX process which can progress without ad-hoc modifications.

The two key models that have been left out of this general discussion is the Agile Method and Cowboy Coding. This is because these are my two preferred methods for user-facing development. I want to talk about them [in more detail later](#) and within the context of research and development in the human factors domain.

In addition to these three overarching models of the software development lifecycle, there are a number of well-known methodologies. These make specific recommendations as to the kind of practical processes, tools, and techniques used for requirements gathering, design, and development. Five of the most common are covered in more detail next.

8.2.4 Structured Systems Analysis and Design Method

SSADM is the first methodology I learned when initially training to become a software engineer. It is based on the waterfall model however it has one notable difference, being that there is far more time devoted to the initial requirements engineering process. Indeed, in the standard design there is a feasibility study, an investigation of the current environment, an understanding of the business organisation and systems, and finally a requirement specification phase. This means that the system is very top-heavy when it comes to modelling the user. In most cases, this is a good way of understanding the context of the development as well as the real system aspects that are required by the organisation commissioning that development. Although in some cases, commissioning organisations start to become uneasy at the amount of time which elapses between starting a project and seeing some tangible technical developments. But, if you have no choice but to use a waterfall type model then SSADM would be the one I would think of first because of its heavy user involvement from the start.

8.2.5 Rational Unified Process

RUP is a methodology that you will probably encounter at some point in your career because it is so commonly used. The Rational software company, who originated this process, were bought by IBM and so the tools and techniques to support this process have been created because it is IBM's preferred development method. One of the main advantages of RUP is the fact that the process is flexible, in this way it can be tailored and modified to suit the development being undertaken with various aspects being included or left out. In general, there are four phases in the lifecycle being the: inception, elaboration, construction, and transition phases. The only real phase to discuss in more detail is the transition phase the others being reasonably similar across all methodologies. In reality, the transition phase is about deployment and the move

from the development into a production setting; making it available and understood by the end user. While RUP is based on an iterative lifecycle, its support for the user-facing aspects of the system is less well-defined than those dealing with project management and software creation. If I had a choice, I would not use RUP directly but would look for an agile variant such as the [Agile Unified Process](#).

8.2.6 Scrum

Scrum is again an agile and iterative methodology that is focused on managing the software development lifecycle in a rapid iterative fashion. Scrum development has two types of roles, the pigs, are the people who are performing the software development process, and the chickens, who are the people for which the development is being created. Understanding that chickens are a first-class citizen within the development cycle is very encouraging for the UX engineer. Scrum accomplishes its iterative releases using sprint's which are broken down into 30 days or 24-hour cycles. Projects are initially broken down into things that need to be accomplished. There is then a plan developed for how these things are to be created. A sprint occurs such that an entire 'plan of accomplishment' can be realised as an incremental development after a 30-day period. In this way, chunks of the system are created piece-by-piece as opposed to a more waterfall like separation of concerns in which the cycle is longer because the project resources are divided into different development teams. This is a useful way to go about UX development because it enables aspects of the software to be released to the user in a fast incremental manner. Indeed, use and testing can already start before the software is near completion. Therefore, changes requested by the chickens can be integrated by the pigs in the next sprint.

8.2.7 Rapid Application Development

RAD can be thought of in two contexts, the first being as a model and the second being a methodology. In the case of UX work, I favour thinking of it more as a methodology than as a model. In reality, there are many forms of RAD but the main factor that differentiate it for UX work are that it uses minimal planning. RAD is far more about rapid software prototyping with high user input than it is a fixed and delineated software development cycle. In this case rapidly created models of the software are tested with users before the main functionality and interaction design are fixed. In this way, the user has a high input into the system. Once the system is created, there is an implicit knowledge of how it will work and what it will look like, along with the functionality it includes.

8.2.8 eXtreme Programming

XP is one kind of rapid application development which requires developers to work in pairs without a detailed design. The lifecycle is cyclical and involves frequent small (point) releases of the software for consumption by the users. XP asserts that by programming together as a pair more can be achieved than by working alone. And that by frequent releases the users will always be kept involved in the lifecycle as opposed to at a distance. This means that user comments can be accommodated very quickly and at the views of users are taken into account as the implementation proceeds. In reality, this means that the users end up with a finished piece of software which

by halfway, or three-quarters of the way, through the project has attained a level of stability. Therefore, additional modifications may not be immediately perceptible to the end user. In this way, the lifecycle minimises the ability to miss undocumented user requirements within the specifications. It also enables changes or errors to be addressed immediately and removes the tendency for users and user testing to be seen as an afterthought to a contractual deadline.

In addition to these methodologies, I can think of at least another twenty which you may come across. However, the five listed above represent the general dichotomy of methodologies and management practices that you should minimally be aware of. SSADN is well-known, mature, and contractually safe. RUP allows flexibility and has a number of tools to aid the development process, however, in using these tools you will be conforming IBMs specific view of the method. Scrum places the user as a first-class citizen, is often used in open source software development, but has a lower uptake in the commercial sector. RAD is characterised by its lack of design and heavy reliance on incremental prototype development which is often seen as suboptimal in the business view of a development lifecycle. Finally, XP adds structure to rapid, agile methods but is intensive and hungry for user time in the development cycle.

8.3 Methodologies More Suited to the UX Process

I find that the more development I do the more I come back to two very different kinds of methodologies for two very different kinds of UX work. In a way these methodologies are very specific to the way I normally do things and in this case you may disagree and have your own preferences. However, I would suggest that when you do not know what you need in the development. Or that the development is purely focused on research, and therefore very little is predictable or known from the outset, Cowboy coding is the only real tangible method of getting started. Once you can begin to predict different aspects of the code development with specific requirements for the software processes cowboy coding is no longer appropriate. Then agile methods become a far more favourable way of creating software that dovetails into business processes and enables a specific timeline to be followed. Indeed, you may wish to use cowboy coding if you are incubating ideas and designs, moving to agile process when you start to align your prototypes with the business requirements of the development.

8.3.1 Don't Know What You Need... Cowboy Coding

Cowboy coding is used to describe software development whereby the developers have autonomy over the process. There is no project plan or direct timelines, and the schedule, development style, and algorithms are all decided by the individual developer or the development team for the specific project. In the case of research development, it is often very difficult to understand the problems that will be encountered; or a timeline for the solution to be developed. In this case, it is inadvisable to conform to a specific fixed lifecycle methodology because understanding what will be required and how long that will take to develop is not possible. Indeed in some cases the experimental outcome of the research is not a fully working system. Rather, a better understanding of the things that can be done and the things that can't be done; you find this method in use in many skunkworks or R&D labs.

This is a key difference to the standard commercial view of software development. Here most of the problems will be soluble because many aspects of the system have already been created and are well understood. It is merely a matter of automation that is required. This is not to say that the development process is not a difficult or taxing one, and we can see that there are many examples of this process failing. Indeed, the reason we have so many lifecycle designs is specifically because projects often fail. However these are very separate, and different, concerns than that of research coding. This is because research coding expects the challenges to be so significantly difficult that the research developer will need to rapidly change their algorithms and processes. Complete and whole systems are often not required with just aspects of the software development being tested. Research software does not have to be tested to the same level as commercial developments, and positive project outcomes are still possible even if the software system is not developed in its entirety.

It is for these reasons that cowboy coding is more practical. The development must commence without wasting time on project planning that will - in the end - serve no purpose. In some ways, the research developer expects their development to not ever reach deployment maturity, or that the problems inherent in good research should be so insoluble that a complete development will never really be possible. Cowboy coding is useful for experimental and skunkworks work because the coding team is often one person or a very tight and focused team. In this way, it is difficult to see how maintaining a large and unwieldy methodology, with the added administrative overheads would be in any way useful. This is not the case for commercial developments or developments that are moving out of the prototypical phase, in which a more rigorous methodology is required.

8.3.2 Know What You Need... Agile Development⁴

The Original Manifesto Signatories are: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Stephen J. Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas.

It is often easy for the UX'er to remain at a distance from the development lifecycle. This is especially the case where they are from a psychology background as opposed to a computer science - or software engineering - background. Many different opinions exist as to the 'best' methodology to use for developing commercial software. However, it is my contention that agile methods are far more useful and flexible concerning the human facing aspects of the development than the others described above. This being the case you should realise that this section on agile methods is my very personal view on performing practical UX work within the context of a larger software development.

"In February 2001, 17 software developers met at the Snowbird, Utah resort, to discuss lightweight development methods. They published the Manifesto for Agile Software Development to define the approach now known as agile software development. Some

⁴You never really know what you need - you just think you do - so this section should be titled 'Kinda Know What You Need...'.

of the manifesto's authors formed the Agile Alliance⁵, a nonprofit organisation that promotes software development according to the manifesto's principles."

Agile software development [**Cockburn, 2002**] is based on iterative methodologies where requirements and solutions evolve through collaboration. The teams that implement these solutions are often self-organising and cross-disciplinary or interdisciplinary in nature. Development is not free-form however, and there is still a development lifecycle and project planning effort which accompanies agile development (see [Figure: Agile Methodology](#)). However, the time between cycles is often very short such that software can be delivered at weekly or monthly intervals. And indeed time periods within the model are often measured in weeks rather than months or quarters; the work being performed in a highly collaborative manner.

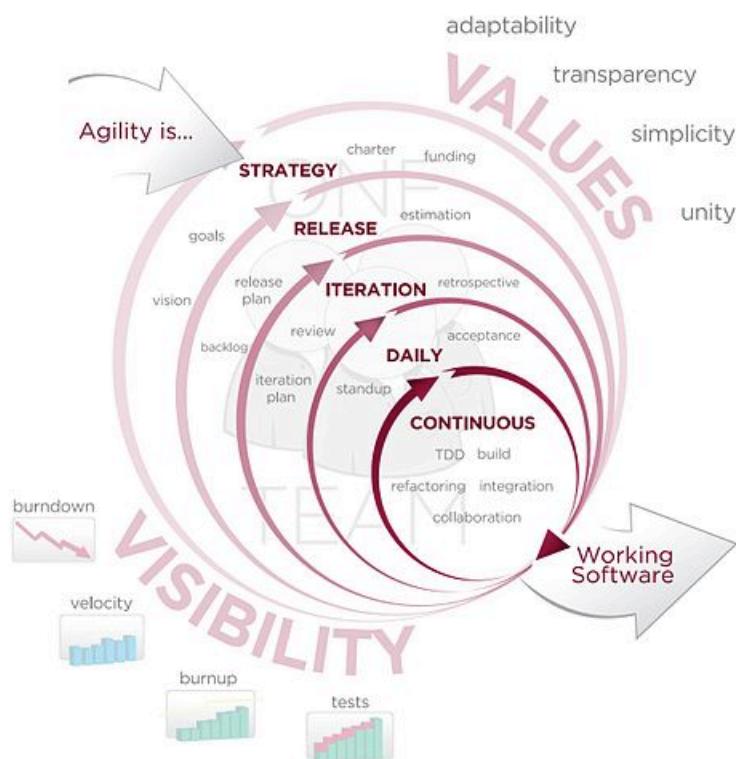


Figure: Agile Methodology. Agile Methodology –Image Credit: Wikimedia.

The Agile Manifesto⁶ aims to ‘uncover better ways of developing software by doing it and helping others do it’ and has four main values:

1. Individuals and interactions over processes and tools;
2. Working software over comprehensive documentation;
3. Customer collaboration over contract negotiation; and
4. Responding to change over following a plan.

Stating that ‘while there is value in the items on the right, we value the items on the left more’.

⁵<http://www.agilealliance.org/>.

⁶<http://agilemanifesto.org>.

We can see that the agile method's lifecycle is highly tailored towards the individual user and keeps these users within the system, being integral parts of the deployment process from start to finish. It is for this reason that agile methods represent, in my opinion at least, the best development lifecycle and methodology for human facing developments. By understanding that the user is placed above contracts and project timelines we can be sure that the final version of the software, which in some cases will be used for many years, fulfils the users needs and through its maintenance cycles fulfils their future needs too.

In addition, there are 12 practical principles behind the Manifesto:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software;
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage;
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
4. Business people and developers must work together daily throughout the project;
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done;
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation;
7. Working software is the primary measure of progress;
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely;
9. Continuous attention to technical excellence and good design enhances agility;
10. Simplicity—the art of maximising the amount of work not done—is essential;
11. The best architectures, requirements, and designs emerge from self-organising teams; and
12. At regular intervals, the team reflects on how to become more effective, and then tunes and adjusts its behaviour accordingly.

As you will be aware from our previous discussions, there are different forms of agile development and these different forms can be useful in different contexts. There are no de-facto winners when it comes to these methods because the context and the desired user outcomes are different for each specific agile method. In this case, you may decide to choose XP, RAD, or any of the other methods listed above, or indeed those created after this text was written. In any case, the focus should always be that the individual and their interaction requirements should be considered before the needs of the development team.

8.4 Separation of Concerns

The separation of concerns is a key aspect of software engineering practice in which each logical part of the system is dissected and separated as much as possible from the other parts of the system. The interactions between the system components are well specified and take place through well-defined interfaces. By separating system concerns it is possible for the development of each part of a complex system to be

undertaken in relative isolation from the other parts. Complexity often occurs in systems because of the combinatorial nature of the tasks required to be undertaken. By decomposing those tasks and separating them into isolated components, complexity — or the possibility of increased combinatorial complexity — is reduced. This means that developments can occur in parallel, and it also means that complex aspects of the system can be more easily handled. An example of a real-world development system that makes use of these separations is Nokia's Qt Framework (see [Figure: Nokia's Qt Framework](#)).

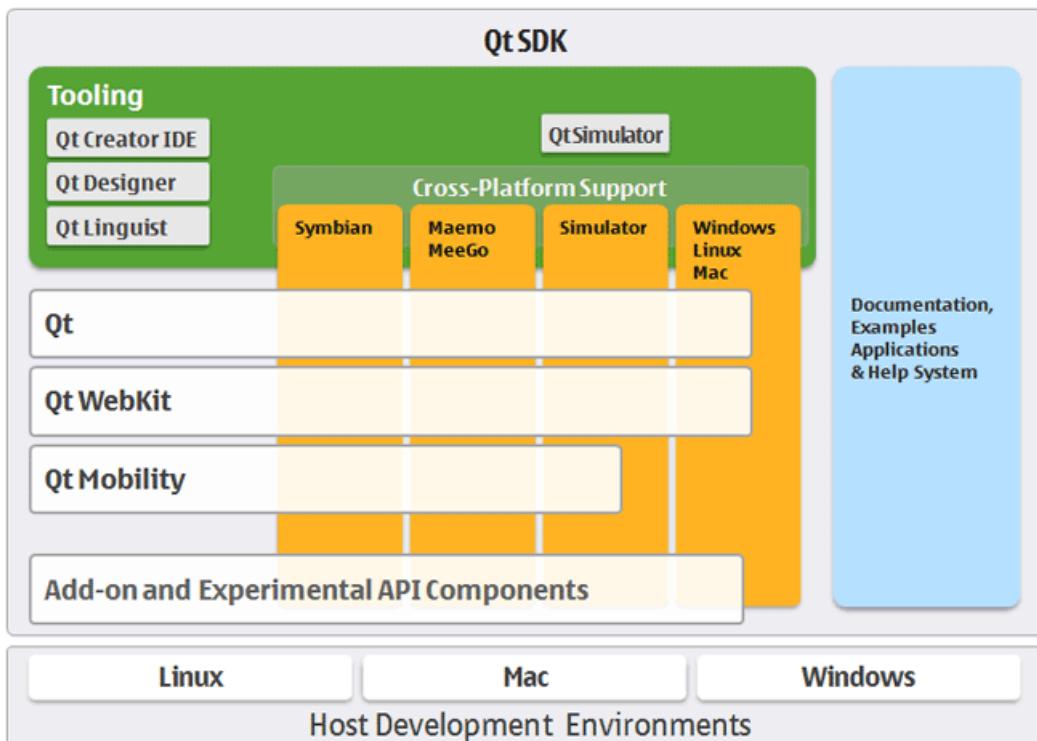


Figure: Nokia's Qt Framework. Nokia's Qt Framework — Qt is a cross-platform application and UI framework. It includes a cross-platform class library, integrated development tools and a cross-platform IDE. Using Qt, you can write web-enabled applications once and deploy them across many desktops and embedded operating systems without rewriting the source code. —Image Credit: Nokia.

For the UX engineer, separating the presentational concerns from the other logical components of the system has a high degree of utility. This is because the interface and the tasks that surround it can then be created in isolation from the other components of the system, only calling on deeper program logic through predefined and well-understood interfaces. Therefore, the interface dialogue components and human aspects of the system can be developed and tested in an agile development cycle until the human-centric parts have been fully tested and evaluated. Finally, the underlying logic can be linked and so the development - even using the Waterfall Methodology - can proceed with the knowledge that the interface components are fit for purpose.

Following this ethos also enables automatic generation of certain aspects of the system. This means that different presentational components can be generated by combining a model of the functional tasks that are required to be undertaken along with a model of the user requirements. In this way, either model can be changed without requiring each presentational aspect to be altered. Small model changes percolate through to the real-world practical deployment without necessitating full systems redevelopments.

Many different systems have evolved out of this desire to separate concerns and to generate aspects of the system automatically for both different platforms and different user types. While the following are not an exhaustive list, they give the reader a general idea of why and how these techniques can be so powerful.

8.4.1 Model View Controller Architecture

The MVC architecture suggests a reasonably simple separation of concerns which isolates aspects of the user input (control), from aspects of the user interface (view), from the underlying description of the programming application logic (model). This means that input is handled by the controller that passes on that input to the model. The model handles this input and updates the underlying data structures and associated output accordingly. Finally, the view queries the model such that changes that have occurred, based on the most recent input in combination with the state of the model before the input, are related back to the user via the interface. Some implementations of the MVC architecture also create a dependency graph that enables only aspects that have changed to be updated through the view. The most complex aspect of this architecture is the model because it handles moving between specifically defined states. This is based on the user input and the previous state of the model. State transition diagrams can be very useful within this context. The view itself is the main focus for the UX engineer. And by abdicating much of the responsibility of the input, event detection, and changes within the model state to other concerned parties within the system, the focus can remain firmly on enhancing user interaction. At the minimum making sure that user's interaction is not hindered by a badly built interface. Also by correctly modelling a view, many different interaction requirements can be supported without the need to individually code each one.

The Model View Presenter (MVP) architecture is a variation upon the MVC architecture in which the model is the interface defining the data to be displayed or otherwise acted upon. The view displays the data while the presenter acts to both retrieve and format data. There is no consistency between implementations of this architecture and the degree of logic permitted within each view varies. However, all implementations have the aim of facilitating automated unit testing and improve the separation of concerns in the presentation logic.

Finally, the **Presentation Abstraction Control (PAC)** architecture is similar to the MVC architecture however it occurs in layers. Each layer contains a number of so-called agents, and the agent comprises presentation, abstraction, and control elements. These agents only communicate with each other through the control element and in this way each agent completely isolates both the presentation and abstraction elements. This means that presentation components can be further disconnected from each other with their underlying program logic housed within the same agent. This has some benefits in the UX domain however it also means that additional communication via the control element needs to be undertaken, and consistent abstraction across layers and agents must also be present.

8.4.2 Multilayered Architecture

Multilayered architecture breaks each aspect of the system into a separate layer running from the user interface layer at the top, through the application layer, through the

domain layer, finally reaching the infrastructure layer. Indeed, it is this domain layer that the model view controller architecture uses as the model to drive the generation aspects of the system. Sometimes the domain and application layers are conjoined as there can often be some overlap between the description of the business rules within the domain and the actual application of those rules within the real software architecture. The infrastructure layer focuses on interactions with the underlying platform, and the services which that platform provides. In this way, the infrastructure layer can be used to enact network connections, memory handling, and all the other system elements you would expect to be present in a modern operating system or platform.

For the UX engineer, the most important aspect of this architecture is the division of the user interface from the other layers. By understanding the flow of data and control in this hierarchical manner, it is sometimes easier for developers to think about the interface creation in isolation. The use of layers is common within software engineering and so it is more in keeping with the thought processes that the developer is already familiar with. The separation of concerns can still be maintained in this model because as long as the application interfaces between the different players are well defined, isolation between the components can still be maintained. In this way, the user interface layer can be changed to suit different user requirements without the need to redevelop the other layers, which are isolated from each other, and the interface layer. However, in this case, the ability to change the interface layer based on an abstract description of the presentation and user requirements of the system is difficult. This is because multilayered architectures are often not based on abstract models of the system that are enacted at runtime. But instead are created to facilitate enhanced development. This means that the machinery required to generate these dynamic adaptive interfaces is not present directly within the system.

8.4.3 Service Oriented Architecture

SOA is a network architecture that requires a high degree of loose coupling of the services provided and the operating systems on which they run. SOA separates the functionality of the application into distinct uses and isolates that functionality from the operating system. This means that an application running on a desktop can call and interact with different services orchestrating the user interaction and the interface aspects of the system separately from those which are used to enact the program functionality. Therefore, services can be called from different nodes on the network, and in the case of the World Wide Web, across servers such that in some cases the developers do not even need to create the application functionality directly.

This kind of architecture is obviously very powerful for the UX engineer. However, it also comes with the problem of the persistence of the services that are offered and their availability from the supplying organisations. Creating a SOA type architecture by orchestrating local services means that specific computer resources can be used for tasks that they are most suited to. Also, a form of parallelism can be accomplished because different services can run at the same time on different computational resources all orchestrated by a single user from a single application interface.

Therefore, SOA's preserve the separation of concerns to a high degree, reduce the complexity of combinatorial functionality because services are isolated from each

other and are often focused on providing a specific service. Orchestrating the services is the most complex task. However, by maintaining this degree of loose coupling the knock-on problems of testing and debugging can be mitigated against those which arise in a system with a high degree of coupling.

We can, therefore, see that there are many different architectures that could be used by the UX engineer. All have strengths and weaknesses: model-based approaches enable the interface to be directly generated, based on an updated view, very easily although additional functionality and machinery are required for this generation. Layered approaches are very practical and in some ways describe the system as it is developed. However, the separation of concerns is not as high as model-based approaches, and so there is always the possibility of functionality drifted between the different layers. Finally, service-oriented architectures support a separation of concerns by a high degree of loose coupling, to such a degree that these architectures are mainly enacted over entire networks. However, their orchestration is complicated, and if the system developer does not have full control over those services, then their persistence and accuracy cannot be guaranteed.

8.4.4 Microservices (and Devops)

“The term “Microservice Architecture” has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.” – Martin Fowler⁷

This is extracted from the best article on Microservices I've come across (even though written in 2014 at <https://martinfowler.com/articles/microservices.html>⁸)

In short, the **microservice architectural** style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

Microservices architectures are currently my preferred way of implementing UX based software. This is mainly because the ability for a piece of software to be created independently of other components and the communication cross facilitated at a later date enables us to separate our concerns not just in the context of development but also in the context of running the system.

As we know, maintenance represents around 70 to 80% of system development effort. And so, building a large monolith means that there are many unintended consequences associated with incorrect maintenance; updates which might very well take down a live system. In the past, this was not a big problem (!) because a release of new software would happen periodically on some sort of hard media and releases would occur infrequently. In this case, we could expect that extensive testing and debugging have

⁷<https://martinfowler.com/articles/microservices.html>

⁸<https://martinfowler.com/articles/microservices.html>

been undertaken because the cost of a new release would be substantial. However, in the current climate of centralised systems, which are run over networks connecting to client application endpoints, then the separation of server and client means that systems would be normally expected to be always on. Further, agile's focus on rapid iteration and deployment means that updates might occur very frequently, indeed up to every two weeks. And so systems which can be independently updated but are cooperating on a single purpose have a huge benefit for the robustness of the user experience.

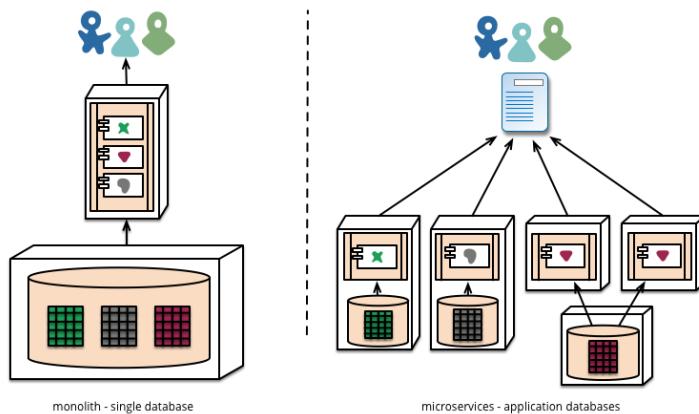


Figure: Microservice Architectures. Microservice Architectures. —Image Credit: Martin Fowler.

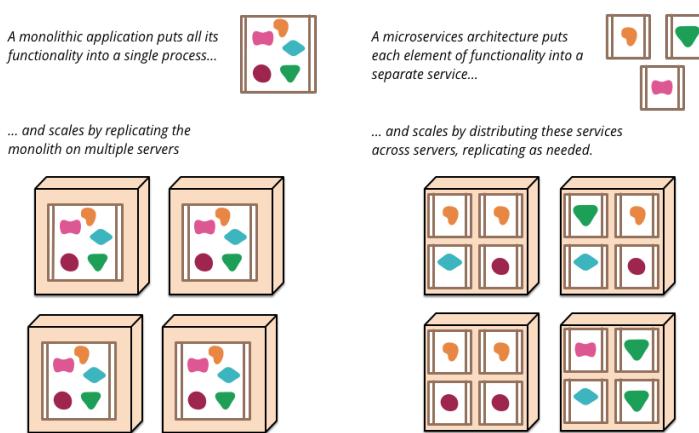


Figure: Micro-Mono. Differences in Microservices and Monoliths. —Image Credit: Martin Fowler.

So in reality what is a micro service? And how are they implemented? This is really a question for software engineering textbooks. However, in the context of user experience and the separation of concerns, we can think of a microservice as being a small component of a larger application but the management of data and interfaces are all controlled by that particular component for its specific job. Communication with other components typically occurs via a bus mechanism such that systems may not even be co-present on the same servers and different languages can be used as long as they all are able to communicate on the same bus. This means systems are very much more flexible in the way that they are developed and in the way that they are maintained.

There are, of course, downsides to microservices architectures, mainly in that they are slower to communicate, and if they are on different servers at distance this communication lag will be increased. However, with modern network and hardware technologies, these lags can appear almost imperceptible to a user at a client endpoint. Microservices

architectures also lead us on to consider development operations or DevOps:

"DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from the Agile methodology." – Wikipedia

DevOps acknowledges the increased need for maintenance of systems and also the agile nature of current development operations such that changes to running systems can be implemented with compartmentalised exposure to bugs and the removal of the possibility of a complete systems failure. Further, DevOps enables immediate feedback on user needs and requirements so supporting rapid agile development cycles. DevOps is beyond the scope of UX but putting users at the centre of all the systems, including those that are both running and being maintained is obviously a method of development, maintenance, and operations something that we should be supportive of.

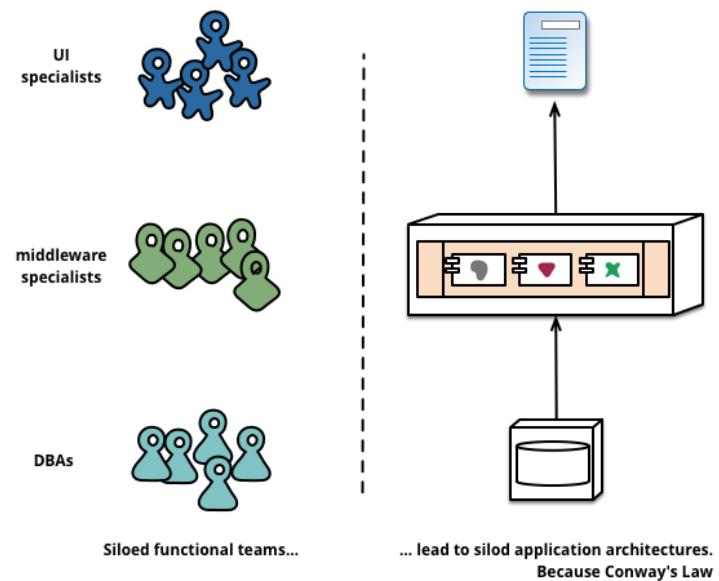


Figure: Monolith Human/System Organisation. Monolith Human/System Organisation. –Image Credit: Martin Fowler.

8.5 Interface Frameworks and the GUI

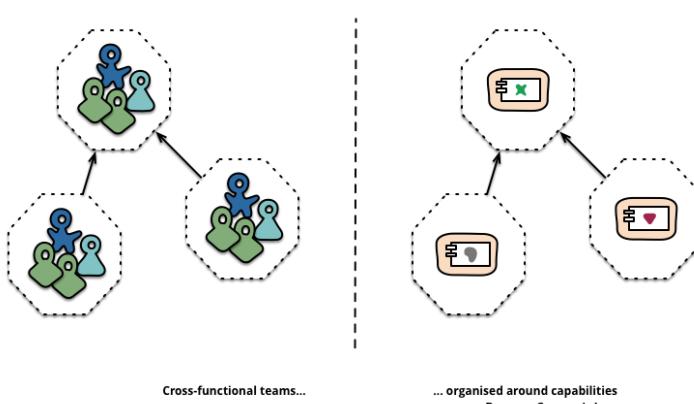


Figure: Microservice Human/System Organisation. Microservice. –Image Credit: Martin Fowler.

Graphical User Interfaces (GUI) are now the standard for user interfaces and will, therefore, be the most familiar to the user. There are other interfaces, such as the command-line interface, which also exist to enable user interaction with the system, however, these are often used by engineering subsections of the user community. The GUI is synonymous with the point and click operating modality of most interfaces.

today and use various real-world naming conventions to orientate users to different functionalities within the application domain. These GUI systems differ from earlier systems in that they make extensive use of the mouse to select the area of attention. Previously systems often worked on a menu based approach by which options are listed, and selections can occur via sequential keyboard operations. Point-and-click meant that different applications or modalities could be displayed at the same time, and the users' attention would switch between these modalities based on their intent. Although managing the input and output of these GUIs tended to be complicated, non-standardised, and buggy. However, systems were developed to address these issues both at the operating system level and at the level of the application development.

8.5.1 Window Manager

Windows managers derived from the WIMP concept that stands for window, icon, menu, and pointing device. These four aspects define the style of interaction that the user can expect within the windowing environment. The use of the pointing device in combination with the screen layout is intended to reduce the cognitive load. Thus enabling the user to remember both the actions which are available while at the same time reducing the learning overhead required to understand those actions and tasks. In this case, the windows manager is the generic name used to describe the operating system specific components which control the placement, appearance, and look and feel of the graphical user interface. The windows manager uses a set of real-world analogies that dovetail into the users' cognitive understanding of what is being presented and tasks that will be required. In most cases the surface of the GUI is called the desktop, applications can be selected by clicking iconic representations or by following an iconic menuing system that allows easy selection of multiple applications at a systems level. This means that the user is not responsible for remembering all of the different applications on the system or how to execute them, but rather lists are presented such that a user can just double-click. Once an application has been initiated it runs in a window. The window analogy is used to reinforce the conceptual format of a view into a working application. Many windows can be open and running at any one time, and each window has a specific menu. Therefore, the commands that are required for each system do not need to be remembered. This list of commands is often found as part of the menu local to the application or in the now commonly known toolbar; that gives graphical representations of the functionality of the application. Few window managers are designed with a clear distinction between the windowing system and the window manager. Every graphical operating system that uses a windows metaphor has some form of window management, however in practice the elements of this functionality vary greatly. Operations usually associated with managers are those which allow the user to open, close, minimise, maximise, move, resize, and keep track of open windows. These facilities exist to provide a degree of consistency between application interfaces with menuing commands such as file, edit and help, etc. These are placed at the same location and following the same conventions across applications and platforms. Finally, most managers also come with additional facilities to help find and control windows and the applications which run on top of them, providing functionality such as docks; application launchers; task bars; and icons.

8.6 Windows Toolkits

Windows Toolkits were created to enable application interfaces to be programmed more swiftly by providing a component-based version of the common interface elements used inside the window manager. These toolkits provide off-the-shelf functionality which enables GUI elements to be speedily assembled and controlled, and also provide for keyboard and pointing interactions. Toolkits also assist development activity because they capture certain principles of interface design, including look and feel, for specific systems and remove the necessity for the developer to reinvent-the-wheel with every interface created. Components within these toolkits are often called widgets. This term is used to denote aspects of the toolkit such as menus, text boxes, scroll bars, spinners, etc. And so a Windows Toolkit can also be called a Widget Toolkit or in some cases a Gadget Toolkit. Another aspect defined within window toolkits is the ability to abstract away from the platform. And enable applications to be created for a specific toolkit (QT for instance) such that native code can be generated to enact that interface on many separate platforms. This cuts down the redevelopment requirements, which are often very heavy concerning interface development, necessary when porting a system from one operating system or platform to another. In many cases, the software engineer will develop code using cross-platform languages and so the application will often be platform neutral; only becoming native once it has been compiled on a specific operating system. However, the interface components often have to be re-implemented using the platforms proprietary Window Manager and toolkit. In this case, the software engineer can write for one multi-platform windowing toolkit that will compile for many platforms. There are problems with this approach, however, in some cases look and feel is not exactly matched to the underlying system and, therefore, the application is cognitively dissimilar to that expected by the user. As an UX engineer, you should only use abstract windowing toolkits if you can guarantee that the resulting native code will always conform to the look and feel of the underlying operating system or platform; as opposed to the specific toolkit itself. If this is not the case, you will end up sacrificing usability for ease of development. Is is an uneconomic model in the long run because the amount of user time will always be far greater than the amount of development time.

8.6.1 Skins

Finally, one of the most basic ways of changing the feel of an interface is to re-skin it. This means that the functionality and the structure of the interface remain unchanged, however, the visual theme and in some cases small aspects of the structure can be manipulated. These minor manipulations enabling users to change and applications look on-the-fly to suit their style of work or preference. Also, the ability to re-skin interfaces means that the application interface itself must be loosely coupled to enable the dynamic changes to occur. Skins may be also associated with themes. And the look and feel of the GUI, in general, can be changed such that these changes percolate through the entire interface including applications built with the native Window Toolkit. An application that is capable of having a skin applied is referred to as being skin-able, and the process of writing or applying such a skin is known as skinning. Some skins make the application more aesthetically pleasing while others can rearrange elements of the interface making the application easier to use. Allowing the interface to be re-

skinned is more work for the developer that allows a certain level of personalization for the user, specifically the power user. Indeed in some regard, designing a system to be scalable means that changes in the interface, and personalization of the interface by the user, is supported and encouraged. This means that testing and evaluation with users is likely to show higher levels of satisfaction because the application can change to accommodate individual preferences.

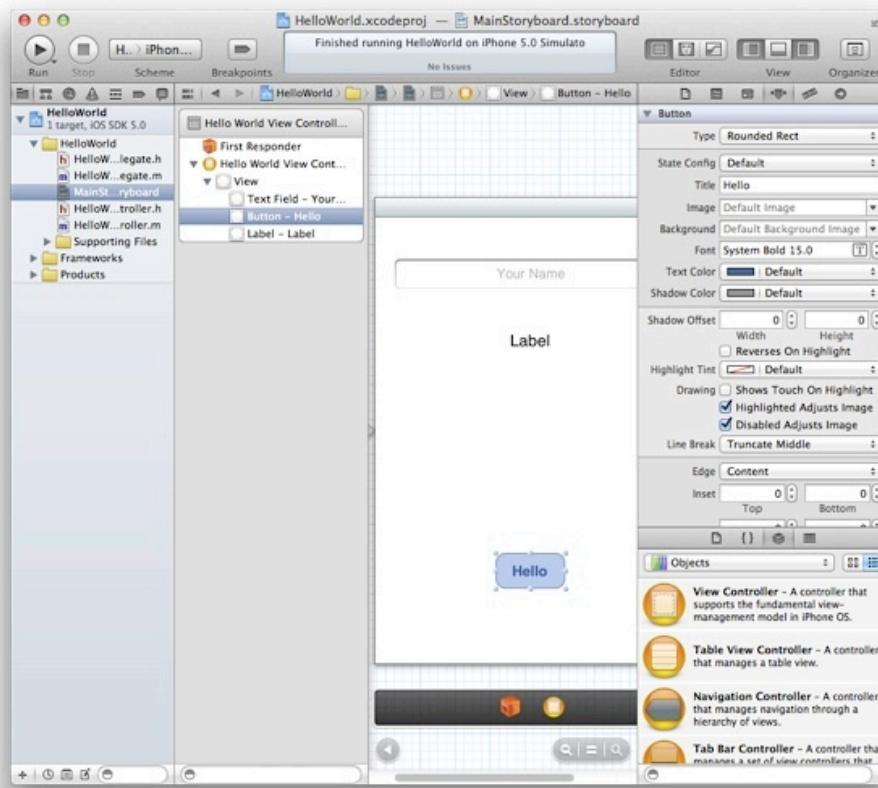


Figure: iOS Development. iOS Development –Image Credit: Apple.

In reality, the interface design has previously been so difficult and non-standard that now the use of windowing toolkits, skins, and Windows managers have become all pervasive. Some argue that this has meant that the research and creativity of interface design have been affected such that different ways of interacting are not now pursued. In this case, you may be wondering why interaction and interface design are even a feature of this chapter. However, we can see that innovations do occur, especially when new devices force changes in the mode of user operations. For example the iPhone (see [Figure: iOS Development](#)) and the Android operating system have both changed interaction modalities such that gesture-based interfaces are now becoming more common. Concerning the standard desktop application, interaction design is focused on the placement of user interface components and the associated functionality of those components. So while the interface does not change directly, interface design is really about matching the needs of the underlying program logic with the best way to interact with that logic from a user perspective. Many interface components do the same job, but in different ways, it is, therefore, necessary to choose the most appropriate method and placement.

8.6.2 UXD and Visual Design

Let's circle back to [UXD and Visual Design](#). In short, we say that UX'ers of the technical/engineering type (you) should not get involved in visual design (although auditory design for conversational interfaces you are as well placed as anyone else). Follow the systems design language, use the development interface design guidelines, and were possible use pre-created toolkits to make sure you match look and Feel. Consistency is very important at the interface level, and you should have a really good reason to break that look and feel.

8.7 Summary

This chapter has mainly dealt with enabling the UX engineer to understand the different kinds of development concerns and methodologies that may be applied to a user-centric development within the context of the software engineering process. Having the effect of making an understanding of the timescale, needs, and concerns of the software engineers more self-evident. And should enable the UX'er to become a better team player with the other disciplines that make up the development team.

As Eric S Raymond suggests in 'The cathedral and the bazaar' [\[Raymond, 2001\]](#) - and elaborates on in 'Homesteading the Noosphere':

"Anyone who watches the busy, tremendously productive world of Internet open-source software for a while is bound to notice an interesting contradiction between what open-source hackers say they believe and the way they actually behave - between the official ideology of the open-source culture and its actual practice. Cultures are adaptive machines. The open-source culture is a response to an identifiable set of drives and pressures. As usual, the culture's adaptation to its circumstances manifests both as a conscious ideology and as implicit, unconscious or semi-conscious knowledge. And, as is not uncommon, the unconscious adaptations are partly at odds with the conscious ideology." [\[Raymond, 1998\]](#)

This is a call to flexibility and user focus, if we simply replace 'open-source' with 'UX' we get a pretty good ethos for UX development.

8.7.1 Optional Further Reading

- [A. Cockburn.] Agile software development. Addison-Wesley, Boston, 2002.
- [A. Davies A] and J. Mueller. Developing Medical Apps and mHealth Interventions. Springer: Cham, Switzerland; 2020.
- [R. C. Martin.] The clean coder: a code of conduct for professional programmers. Prentice Hall, Upper Saddle River, NJ, 2011.
- [A. Oram] and G. Wilson. Beautiful code. O'Reilly, Beijing, 1st. ed edition, 2007.
- [E. S. Raymond.] The cathedral and the bazaar: musings on Linux and Open Source by an accidental revolutionary. O'Reilly, Beijing, rev. ed edition, 2001.
- [I. Sommerville.] Software engineering. Pearson, Boston, 9th ed edition, 2011.
- [The Agile Alliance,] Website <http://www.agilealliance.org/>⁹.

⁹<http://www.agilealliance.org/>

8.7.2 International Standards

- [ISO/TR 9241-100:2011.] Ergonomics of human-system interaction — part 100: Introduction to standards related to software ergonomics. TC/SC: TC 159/SC 4 ICS 13.180; 35.180, International Organization for Standardization (ISO), Geneva, Switzerland, 2011.



Self Assessment Questions

Try these without reference to the text:

1. What are the relationships between the development methodologies discussed, which is best for UX and why?
2. Given need to rapidly create an interface prototype, which agile method would you use and why?
3. What's wrong with Cowboy Coding?
4. What is the Separation of Concerns and why is it important?
5. Windows Toolkits are often specific to the Windows Manager, how can you more efficiently design for multiple Windows Managers?

9. Prototyping and Rapid Application Development

“If a picture is worth 1000 words, a prototype is worth 1000 meetings.”

– David & Tom Kelley, Founder, and Partner of renowned Design and Innovation Consultancy IDEO

In this chapter we’re going to be looking at Rapid Application Development (RAD) and Prototyping. By prototyping we simply mean a mock up that can (often) be run. And when we say run we mean this could be a wizard-of-oz kind of run, in that sense, whereby there’s a person behind the screen making it work. Alternatively, it could be actually running to a certain level of accuracy which could be implemented in part and parts that might not be implemented. So it just depends, but run means that things can be looked at by users. And you can work through the steps of how things might work.



Figure: Lo-Fidelity Website. In this image of a lo-fidelity prototype we can see the idea that you can have a just an A4 page really that's coloured in to make it look a bit more like the web, or an application, and then you've got a set of tasks that you need to do on that paper mock-up. There are various kinds of coloured tags, if you look on the left hand side, that you'll then replace/overlay bits of the fake screen (paper on the right) with so you can see what actually would happen once you've 'pressed buttons' or 'made selections'. –Image Credit: Adobe.

In this case, the initial investment is low in terms of time and coding, because it's really a prototype, it's not the real thing and so hasn't taken the development effort required to make everything work. Its job is to just allow the user to get closer to what the system

could look like and how it could work without making a high initial investment, and as such it is cheaper in the initial development but also allows a user to make rapid changes over time.

9.1 Prototyping

UX prototyping refers to the process of creating interactive and tangible representations of user experiences. It involves building low-fidelity or high-fidelity prototypes that simulate the functionality, flow, and visual design of a digital product or service. UX prototyping plays a crucial role in the user-centred design process, allowing designers and stakeholders to gather feedback, test concepts, and iterate on designs before the actual development phase.

Key aspects of UX prototyping focus on:

User-Centric Approach: UX prototyping focuses on understanding and addressing user needs and expectations. By creating prototypes, designers can gather user feedback early in the design process and make informed decisions based on user insights.

Interactive Simulations: Prototypes are designed to simulate user interactions and workflows, providing a realistic representation of how the final product will behave. This allows designers to test and refine the user experience before investing in development.

Low-Fidelity Prototypes: Low-fidelity prototypes are quick and simple representations of design concepts, often created using paper sketches or digital wireframes. They are useful for exploring and validating initial ideas, gathering feedback, and making early design decisions.

High-Fidelity Prototypes: High-fidelity prototypes are more detailed and visually polished, closely resembling the final product. They can be interactive, allowing users to navigate through screens, interact with elements, and experience the product's functionality. High-fidelity prototypes are valuable for user testing, stakeholder presentations, and demonstrating the user experience to clients.

Iterative Design Process: UX prototyping facilitates an iterative design process, enabling designers to make improvements based on user feedback. By testing and refining prototypes, designers can uncover usability issues, identify areas for improvement, and iterate on the design until it meets user and business goals.

Collaboration and Communication: Prototypes serve as a visual and interactive communication tool, enabling designers to effectively communicate their ideas to stakeholders.



Figure: Stencils. So these templates show you what a mobile phone looks like. And you can have different ones for tablets and different ones for standard Windows systems with the different icons that are current at the moment and just to make it still a bit freeform but a little bit more. It looks a bit more serious if you like it looks a bit more polished because you're using these stencils. You can also have pads. —Image Credit: <https://www.uistencils.com>.

holders, developers, and other team members. Prototypes bridge the gap between design and development teams, aligning everyone on the project's vision and goals.

Prototyping Tools: Various prototyping tools are available that aid in creating interactive prototypes, such as Sketch, Adobe XD, Figma, InVision, and Axure RP. These tools offer features like drag-and-drop interfaces, interactions, animations, and the ability to simulate user flows.

Overall, UX prototyping is a powerful technique that empowers designers to iterate, test, and refine their designs based on user feedback. By incorporating prototyping into the design process, teams can create more user-centred and effective digital products and services. We will be focusing on the fidelity spectrum which runs from lo-fidelity at one end, and hi-fidelity at the other.

9.2 The Fidelity Spectrum

What you'll find is that there are two extremes of prototypes; lo fidelity prototyping and hi fidelity prototyping. And as we move from lo fidelity to hi fidelity there is really a spectrum, and a diversity of change such that lo-fi facilitates big changes which become increasingly small as you move through to hi-fi. This is because there's more work and more cost involved in the hi-fi. Also, as we've seen the closer you get to hi fidelity prototypes, the less likely it is a user will want to make changes because they'll feel you've already put a lot of work into it. So really make sure that the users have been involved on the lo-fidelity prototypes, just so that you can understand any final changes or issues that need to be fixed from a user perspective, realising you are not going to get these to the level of large changes the closer you get to hi-fi prototyping.

Along the spectrum you can increasingly refine the lo-fi prototypes moving from hand drawn prototypes to templates that are still lo-fi but look much more professional, giving a more exact representation of the artefact. We can see that the lo-fidelity prototypes above are mainly freeform and are created so they look handwritten prototypes, none-professionally drawn drawings. This is a positive thing as people don't get scared of making changes. But once you want to make it a little more formal you need to factor in the point that people will be more resistant to change.



Figure: Lo-Fidelity Application. Likewise, for the next one, it is really a mockup of an iOS or Android mobile device. And this one is actually more of a three dimensional low fidelity prototype, because here, it's a little square box. And you see the little numbers at the top are pull tabs in green. So they say 4, 3, 2, and 6 when you look at them a bit more clearly. So these tabs allow you to pull the sheet up so you can see what's actually happening when each step progresses; pull that tab up and something is revealed underneath, push it down and something is covered. So you can see it's got a little bit more of a refinement than just a pure lo fidelity prototype. Remember, that prototyping is on a spectrum as opposed to just lo and hi! —Image Credit: Adobe.



Figure: Templates. So here there are templates that look like a standard web page or mobile app. These come in books so that they can be used as a starting point, and then you can just use that to design your system dropping down from those kind of initial starting points. It gives you things that you're allowed or expect such as dates and times of when this was created, etc, you can also have smaller ones that look like post it notes to be placed on a whiteboard. And then you can show those in more detail on the whiteboard. You can then use the the symbols from the stencils on those templates. —Image Credit: <https://www.uistencils.com>.

Initially, it's still a person just drawing with a little linkage here and there. You shouldn't undervalue this, the value of prototypes, especially of lo-fidelity prototypes is there ability to be changed by anyone (professionals and users alike). And so designers in large companies tend to use lo-fidelity prototyping. One example is the Nintendo Miiverse, one of the original Nintendo designs which utilises a three dimensional box that you hold and pull and push the cards from the top of it. Indeed, designer Kazuyuki Motoyama explains that the only way to actually know what a Miiverse would feel like was to hold it. That's when he built this prototype out of cardboard.

9.2.1 Nintendo Miiverse



Figure: Miiverse. Miiverse Prototype on a Wii U prototype –Image Credit: Nintendo.

Miiverse was actually modelled on Wii U prototype (which was intended to be one conjoined system) three dimensional box that you hold and pull and push the cards from the top of it. So you can see exactly what will happen and when a button is pressed. What's more, oftentimes, you're moving to a certain set of tasks so that you don't get things wrong. So it will tell you what to do. For instance, if you press this button, this will happen. And then you reveal that by removing (or adding) the card. So you are if you like running it yourself, if you like you can get these kinds of index card templates, whereby you have the mobile app on them, and then you can spread them on your desk so that you can show people how things are going to happen, move these templates around, which is quite a nice way of doing it. Once you turn it over, it can tell you the kind of tasks and the things that are expected to be accomplished. And in some cases, it's got a reverse. So you know, if you click a button you clicked over, and then it's got the button clicked on the other side. So you can have different forms of this. But it's all to get this kind of level and feeling of what this thing is going to be like.

9.3 Prototypes in Software Engineering

Being a software engineer, there's always the desire in a lot of ways to just get moving, to get moving on the actual interface, don't bother about design, just get moving on it, get something out there. That's a useful way of doing it when you're trying to get an idea about what it is that you're trying to create. When you need to know something about the systems that you're trying to create and what they're trying to do. And if you don't

know that yet, then it can be problematic without an initial interface model.



Figure: Wii U. Wii U Production Version —Image Credit: Nintendo.

out a real clear idea necessarily. But that's because if you don't have an idea about what's required, or what you're going to build, because it's so novel, then you've got to find some way of doing it.

As we're seeing, increasing development increases fidelity. Once we have moved along from lo-fidelity then we get into common graphics or flow chartering apps, such as sketch, Illustrator and OmniGraffle. And these things can actually also generate some kind of metadata to allow you to better create the screens etc. And then we get to the high fidelity prototype, which is really just to all intents and purposes, a visual designer for a screen but without any data. And then you're able to create the interface and screens and add dummy plug in data in the back of it.

We're going to talk about the separation of concerns soon enough, however we can see how this might work in prototyping as the interface is separated from the data on the interface (and this data is the result of programme logic). When we think about separation of concerns, even in high-fidelity prototypes, it allows us to create the interface for different targets, might be mobile, might be desktop, different kinds of targets, we create the prototype, we create the interface for that target, but then we need the actual programming logic. And so we can then build the programme logic at the back. And this is where it really is useful to have some kind of RESTful interface decoupling of the programme logic from the interface itself. Or aspects like micro services, whereby you can develop parts independently of the other parts of the system. So you can build it up in that kind of separated way.

Now, when we talk about cowboy coding, this is what we mean in a prototyping context. Agile presumes that you've got some idea about what it is you're building, if you have no idea about what it is you're building, then cowboy coding kicks in. We just start making lots of lo-fidelity prototypes out there trying to get people to use them and understand which is the best one, which isn't with-



Figure: UX Cards. UX Cards are a good jump between freehand and more formal where the formal style is still amenable to fast low entropy changes by users and none experts. These cards have slightly more fidelity, if you look, they're not so sketchy looking anymore. And then much more like widgets that you might see nicely printed. And this is like a slide deck, which you could create different sizes of screens. And you can have some way you've got the actual fidelity of say, an image there. But on the back of that, you can flip it over. And it actually gives you that this is an image. And it's just a blank image. So some of it looks like a wireframe. And some of it looks like it's got a high fidelity. —Image Credit: <https://uxkits.com/>



Figure: Mid Fidelity? Professional Looking Generation. And you can see here that really this is about creating them, putting them on a big page showing the linkage of what the flow will be, what the different control flows will be. So that's something you should really also consider. —Image Credit: <https://uxkits.com/>.



Figure: Card Index. Card Index —Image Credit: <https://www.uistencils.com>.

quickly within in the space of a few days, because you can make the changes very quickly.

People are excluded from hi-fidelity prototypes, because you need more knowledge of how to use tools, whereas lo-fidelity, you just need to know how to use paper and pen. It's pretty ubiquitous currently. Lo-fidelity prototypes allows you to think with your hands and not be too bothered about what things might look like, and what decisions you're making, because it's easily changeable by rubbing out with a pencil with an eraser or scrubbing over a pencil and drawing the lines and that kind of thing. So it's a lot easier to do. It allows you to to make lots of validation steps of what's required with users very

Table: UX Prototypes: Low Fidelity vs. High Fidelity. Clickable or static? Axure or paper? No matter which prototyping tools you use, the same tips apply to preparing a user interface prototype for the most effective user research. Credit: <https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/>

	HIGH-FIDELITY PROTOTYPE	LOW-FIDELITY PROTOTYPE
Clickable links and menus	Yes: Many or all are clickable.	No: Targets do not work.
Automatic response to user's actions	Yes: Links in the prototype are made to work via a prototyping tool (e.g., InVision, PowerPoint). Visuals	No: Screens are presented to the user in real time by a person playing "the computer."
Realistic visual hierarchy, priority of screen elements, and screen size	Yes: Graphics, spacing, and layout look like a live system would look (even if the prototype is presented on paper).	No: Only some or none of the visual attributes of the final live system are captured (e.g., a black-and-white sketch or wireframe, schematic representation of images and graphics, single sheet of paper for several screenfuls of information). Spacing and element prioritization may or may not be preserved.
Content	Content and Navigation Hierarchy	Content and Navigation Hierarchy
	Yes: The prototype includes all the content that would appear in the final design (e.g., full articles, product-description text and images).	No: The prototype includes only a summary of the content or a stand-in for product images.

And it allows for ‘user centred design’ whereby the user is actually more part of the part of the process. Because with user centred design, we want the users to be involved in the process, not excluded, and most users won’t have the skills to use SketchUp or OmniGraffle, or even the hi-fidelity generation engines. Further, they’re not as free in their designs if they have to have somebody else do that for them. So even though there might be two people together designing the system, users who wants the designer to make a change in a hi-fidelity prototype is going to be much more reticent because they think it will be more work.

9.4 Users, Commissioners, Engineers!

Prototypes are used with users, with commissioners, and with engineers and typically in that order from lo to hi fidelity. So what would normally happen is that you’d have users who would look at these lo-fidelity prototypes that have been drawn freehand (say) just written on post it notes, so you can move things around quickly, you might then firm that up into something that’s mid range. So it’s more along the lines of an actual computer generated hierarchy using SketchUp, etc, which is something that a lot of designers use in developing designs, and so you can see what’s going on at this stage.

Now, remember, everything I'm talking about when it comes to prototypes is visual; I'm not talking about any prototypes for conversational interfaces, or zero UI, or auditory design, because most users don't think about these aspects in the early stages of the design. So as you're a UX'er, you're going to have to think about how to do for non-visual prototyping because there's no best practice out there.

So commissioners will expect the system to look more professional and they will be expecting to make minimal to no changes, however, it still might not be the final design. But with the engineers, you're probably going to create something that's the final thing, you're probably likely to let the commissioners see that as well at one of the development meetings, but mainly then, with the engineers, you can make the changes and generate out something for the engineers to start work on.

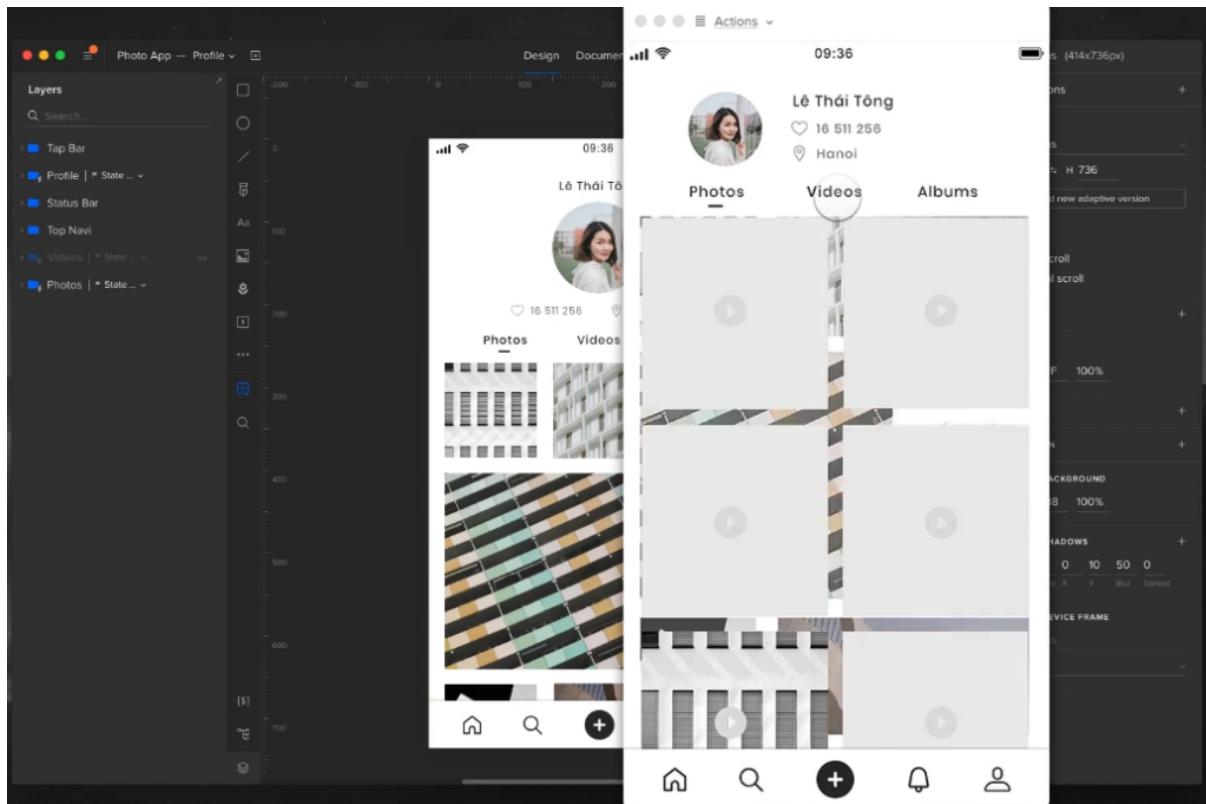


Figure: Hi Fidelity. Building a Pseudo Running System --Image Credit: <https://www.uxpin.com/>.

As a UXer, you're probably going to start work on the low fidelity prototypes and this is typically because you're going to be working more with the users, once it gets to the visual designers, they're going to be using mid range SketchUp like applications. So they'll do the visual design there. Now translating that visual design to the hi, very hi-fidelity prototypes ready to generate out for the engineers might be your job again, because really, you're translating the visual design into something that can then be created as the basis for code. Or it might be an engineer, or it might be a specialist visual designer.

9.5 Rapid Application Development

Rapid Application Development is really the boundary of these hi-fidelity prototype tools. And now, hi-fidelity prototyping tools have taken over the space of rapid application development. RAD was a 1991 era idea or at least it was expressed in that idea whereby we needed to create applications quickly without any sort of background machinery such that users could see what was going to be there and is heavily reliant on prototyping. And so this is kind of a method of doing the prototyping. In this case, RAD is a software development methodology that prioritizes speed and flexibility in delivering high-quality applications. It emphasizes rapid prototyping and iterative development cycles to quickly build and deploy software solutions. RAD aims to accelerate the development process, reduce time to market, and improve customer satisfaction by involving end-users and stakeholders throughout the development lifecycle. Key characteristics and principles of Rapid Application Development are

1. **Iterative Approach:** RAD follows an iterative development model, where software is developed in small increments or prototypes. Each iteration focuses on specific features or functionalities, allowing for quick feedback and continuous improvement. This is a lot like the newer concept of Agile - which is now tied to UX.
2. **User-Centric Focus:** RAD places a strong emphasis on user involvement. Users and stakeholders are actively engaged throughout the development process to gather requirements, validate prototypes, and provide feedback. This ensures that the final product meets user expectations and addresses their needs effectively.
3. **Prototyping:** RAD heavily relies on prototyping techniques to rapidly build and refine software solutions. Prototypes are created early in the development process to demonstrate the application's core functionality, interface, and user experience. User feedback is incorporated into subsequent iterations, leading to an evolving and user-centric design. This is the critically important part of RAD for UX and we might even extend this to imply that an initial software framework can be generated from a hi-fidelity prototype.
4. **Collaboration and Teamwork:** RAD promotes close collaboration among developers, users, and stakeholders. Cross-functional teams work together to define requirements, design prototypes, and deliver working software in short development cycles. Effective communication and teamwork are crucial to the success of RAD projects.
5. **Reusable Components:** RAD encourages the reuse of existing software components and modules. By leveraging pre-built components, RAD speeds up development and reduces the effort required to build certain features. This approach enhances productivity and allows developers to focus on unique requirements and functionality.
6. **Parallel Development:** RAD enables parallel development activities by dividing the project into smaller modules or components. Different teams or developers can work on these modules simultaneously, ensuring faster development and integration of various software components.
7. **Flexibility and Adaptability:** RAD emphasizes flexibility and adaptability to changing requirements. The iterative nature of RAD allows for adjustments and refinements based on user feedback and evolving business needs. This enables the software to evolve and align with changing market dynamics.

8. **Time and Cost Efficiency:** By employing rapid prototyping, iterative development, and close collaboration, RAD can significantly reduce development time and costs. The focus on early user involvement and continuous feedback minimizes rework and ensures that the final product meets user expectations.

It's important to note that RAD is not suitable for all types of projects. It is best suited for projects with well-defined scope, clear business objectives, and actively involved users and stakeholders. RAD is particularly effective in situations where time to market is critical or when requirements are subject to change.

Overall, Rapid Application Development offers a dynamic and flexible approach to software development, enabling faster delivery of high-quality applications while maintaining a user-centric focus. If you're working in RAD (or at least a close variant), then you also need to make sure that the system you're using will generate an application that can run and that can be populated fake data, but that can might not have any real programme logic. For the interface, it will display content which is understandable by a user and it might also then have links to databases in the background and it can become increasingly complex as development and modelling proceeds. RAD is not just about user experience it's also about any system development via higher level languages and so you can also generate, for instance, a database application via flowcharts via state transition diagrams via UML. So any system development will utilise RAD because it's a faster transition from the design phase to the application phase.

9.5.1 RAD & Prototype Comparison

RAD and Prototyping are both software development approaches, but they differ in their objectives and the stages of the development process they focus on. However, there is clear linkage in that RAD heavily utilises prototyping and prototyping at the very hi-fidelity often include the ability to generate a working RAP prototype for user involvement and as a kickstart to the initial development effort.

9.5.1.1 Objectives

- Rapid Application Development (RAD): RAD aims to expedite the development process by emphasizing iterative development and quick delivery of functional software. Its focus is on rapid iteration and user feedback to ensure that the final product meets user requirements.
- Hi-Fidelity Prototyping: Hi-Fidelity Prototyping focuses on creating highly detailed and interactive prototypes that closely resemble the final product. The goal is to test and validate the design and user experience before investing significant resources in development.
- Lo-Fidelity Prototyping: Lo-Fidelity Prototyping focuses on creating low-detail, basic prototypes that represent the core functionality of the final product. The goal is to quickly explore design concepts, test ideas, and gather early feedback before investing significant resources in development.

9.5.1.2 Development Process

- Rapid Application Development (RAD): RAD typically follows a cyclic or iterative process, where each iteration involves requirements gathering, prototyping,

development, testing, and deployment. The emphasis is on reducing the time between iterations to quickly deliver a working product.

- Hi/Lo-Prototyping: Hi/Lo-Fidelity Prototyping is usually an early stage in the development process. It involves creating detailed prototypes that simulate the functionality and user interactions of the final product. These prototypes are used to gather feedback and validate design decisions before moving to development.

9.5.1.3 Level of Detail

- Rapid Application Development (RAD): RAD focuses on creating functional software quickly, often with a minimalistic user interface and core features. The emphasis is on delivering working software within short timeframes, allowing users to provide feedback and shape the development process.
- Hi-Fidelity Prototyping: Hi-Fidelity Prototypes are highly detailed and aim to closely resemble the final product in terms of user interface, interactions, and visual design. They often include realistic data and simulate real-world scenarios to provide a comprehensive user experience for testing and validation.
- Lo-Fidelity Prototyping: Lo-Fidelity Prototypes are intentionally low in detail, often using simple sketches, wireframes, or basic interactive mockups. They are not intended to replicate the final product's design or functionality but rather to capture the basic structure and flow of the application.

9.5.1.4 User Feedback

- Rapid Application Development (RAD): RAD encourages frequent user involvement throughout the development process. Users have the opportunity to provide feedback on the working software during each iteration, allowing for continuous improvement and alignment with their requirements.
- Hi-Fidelity Prototyping: Hi-Fidelity Prototypes are primarily used to gather user feedback and validate design decisions. Users can interact with the prototype, provide input, and offer suggestions for improvement before the development phase begins.
- Lo-Fidelity Prototyping: Lo-Fidelity Prototypes are often used to gather early feedback from stakeholders and users. Since they are low in detail, the focus is on validating and refining the overall concept, functionality, and user experience rather than specific design elements.

9.5.1.5 Time and Resource Investment

- Rapid Application Development (RAD): RAD aims to deliver working software quickly, reducing time-to-market and allowing for faster user feedback. It requires a high level of collaboration, skilled development teams, and well-defined requirements to ensure efficient and rapid development.
- Hi-Fidelity Prototyping: Hi-Fidelity Prototyping requires significant upfront time and effort to create detailed and realistic prototypes. However, it can potentially save time and resources in the long run by identifying design flaws and usability issues early, before investing in full-scale development.

- Lo-Fidelity Prototyping: Lo-Fidelity Prototyping is relatively quick and inexpensive compared to other prototyping methods. It allows for rapid exploration of ideas and concepts without investing significant time and resources. However, it may require additional effort to translate the low-fidelity prototype into a high-fidelity design during the development phase.

In summary, RAD focuses on quickly delivering functional software with iterative user feedback, while Fidelity Prototyping emphasizes creating detailed and interactive prototypes for early validation of design decisions. The choice between the two approaches depends on the project requirements, timeline, and the level of detail and user involvement needed at each stage of development.

9.6 Summary

Though low-fidelity prototyping has existed for centuries, it has recently become popular with the spread of agile design methodologies, inspired by several movements including design thinking advocates for ‘thinking with your hands’ as a way to build empathetic solutions. Lean startup relies on early validation and the development of a minimum viable product to iterate on. User-centered design calls for a collaborative design process where users deliver continual feedback based on their reactions to a product’s prototype.

Prototypes are used by application/system users, commissioners, and engineers as part of the initial Design Phase, such that once requirements are collected and you want to convey information to Visual Designers or Software Engineers. As a UX'er you'll probably mostly work with Low-Fidelity prototypes. Initial low investment of time which makes it very flexible and Iteratively evolves to be more like an app with a higher investment of time and therefore less flexibility. very hi-fidelity prototypes can often be ‘run’, and/or generate, a working framework, and it is at this point that RAD processes can be utilised. RAD demands frequent user interfacing, and is heavily reliant on prototyping.

Finally, remember, ‘A process that is not critically dependent on prototyping is not a UX process.’ — Julian Caraulani, Senior UX Designer at Samsung.

9.6.1 Optional Further Reading

- [Ben Coleman and Dan Goodwin (2017)] — Designing UX: Prototyping. SitePoint. 978-0994347084.
- [Reinhard Sefelin, et al (2003)] — “Paper prototyping - what is it good for?: a comparison of paper- and computer-based low-fidelity prototyping”. CHI ‘03 Extended Abstracts on Human Factors in Computing Systems. ACM: 778–779. doi:10.1145/765891.765986. ISBN 978-1581136371. S2CID 275647
- [Nick Babich (2020-09-25)] — The Magic of Paper Prototyping. Medium. Retrieved 2021-12-13.
- [Steve McConnell (1996)] — Rapid Development: Taming Wild Software Schedules, Microsoft Press Books, ISBN 978-1-55615-900-8

- [James M. Kerr & Richard Hunter (1993)] — Inside RAD: How to Build a Fully Functional System in 90 Days or Less. McGraw-Hill. ISBN 0-07-034223-7.
- [James Martin. (1991)] — Rapid Application Development. Macmillan. ISBN 0-02-376775-8.



Self Assessment Questions

Try these without reference to the text:

1. What is Rapid Application Development?
2. What is Prototyping?
3. Why does fidelity matter?
4. What are the Qualities of Low (Lo) Fidelity Prototypes?
5. What are the Qualities of High (Hi) Fidelity Prototypes?

10. Principles of Effective Experience (Accessibility)

While pedalling a Boris Bike I start to shuffle in my pocket. Where is that damn iPhone? Finally, I find it. The next challenge is to get it out my pocket whilst still navigating the Humvee of the bike world. Eventually, I give up. I pull over and get my iPhone out. I look up the nearest docking station with spaces and get back on my way.

– LondonCyclist

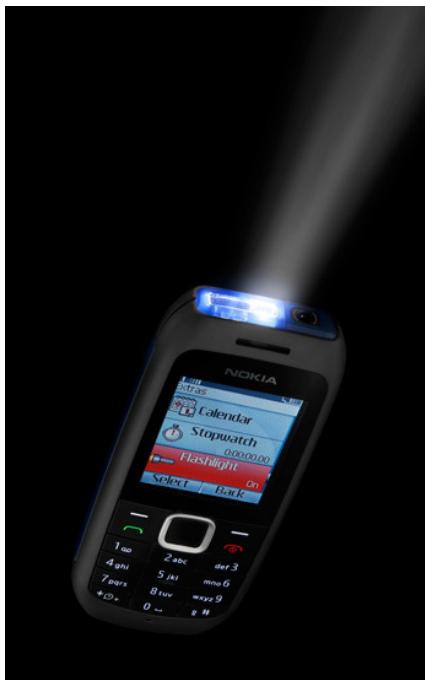


Figure: Nokia Torch. Nokia 1616 now including a Torch. –Image Credit: Nokia.

This is an example of ‘[situational impairment](#)’ a barrier to access, the inability to produce or accomplishing the intended effect; in short non-effectual use.

10.1 Effective¹, effectual², accessible³

To my way of thinking these three terms mean the same thing, and, in reality, were going to be talking about accessibility. However, you should be thinking of accessibility in the more general terms of effective or effectual use. that because the concept of accessibility is much broader than the narrow confines of disability it is often associated with [\[9241-129:2010, 2010\]](#).

This notion of the situationally-induced impairment, by which they do not mean an actual impairment of the person directly, but indirectly by the computational device or the environment in which it must be used. They point out that non-disabled individuals can be affected by both the environment in which they are working and the activities in which they are engaged, resulting in situationally-induced impairments. For example, an individual’s typing performance may decrease in a cold environment in which one’s finger does not bend easily due to extended exposure at low temperature. Or the similarities between physical usability issues on both small devices, such as a Mobile Telephone, Personal Digital Assistant, or any other hand-held device with a small keyboard and display, and

¹The removal of all technical barriers to effective interaction – this is my definition, but there are plenty more in [the Appendix](#).

accessible interaction scenarios. This means that by making your interfaces accessible, you can transfer accessibility solutions into the mobile and situational context.

This support for situational impairments, through flexibility, adaptability, and the ability to personalise the user experience, in other words transformable UIs, is important for access in developing regions too. Web use in developing regions is currently characterised by constrained operating modalities. Slow speed, low computational power, reduced bandwidth, compact keyboards, small screens, and limited power, all compound the problem of access and inclusion. In addition, interaction is sometimes without conventional written language and illiteracy is also a barrier to information and services. However, the benefits of technology are so great that the peoples of these regions often adopt resourceful methods of interaction and access sometimes repurposing resources so that they are put to a different use than for which they were intended. The most obvious example was noted when Nokia sent a team to Africa and realised that people were using mobile phones, able to connect to a carrier or not, as a torch (see [Figure: Nokia Torch](#)). There is also the well-known use of Pringle tubes to direct a single point of wifi around a village community (see [Figure: Pringle Tube Antenna](#)); as well as other novel work which is less high-tech [[Zobel, 2013](#)]

So how can we develop effective user experiences? Well openness comes to our rescue – as do Sony Ericsson⁴, and Little Fluffy Toys Ltd⁵ – with Sony's LiveView and Little Fluffy Toys' Cycle Hire LiveView Gadget (see [Figure: LiveView Cycle Gadget](#)) “Using Bluetooth, the Live View gadget connects to your Android phone (2.0 or later). You can then do cool stuff such as check out the latest tweets, SMS messages, Facebook messages, take pictures remotely or, find the nearest Boris Bike docking station.” Problem solved! By taking account of the inability for the cyclist to access data on a conventional device using a conventional computer system, the engineers have created an accessible product; because the key to accessibility and effectual use is a focus on the user, and their personal needs. But to address these needs, needs which you may not understand or have experience with, you must think (like the Sony UX'ers) about extensibility and adaptability, ergo personalisation and openness of both the data, and the APIs to access it. By making sure that the system is open, and that aspects of the interface and human facing system components can be modified, adapted, or interacted with via an API, the developer is making an explicit statement which allows for extensibility. This



Figure: Pringle Tube Antenna. Pringle Tube Antenna. —Image Credit: Gregory Rehm.

⁴<http://www.sonyericsson.com>.

⁵<http://www.littlefluffytoys.com>.

means that all aspects of the application can be modified and changed once further information comes to light or in the event of processes in the organisational systems of the commissioner changing.

Accessibility aims to help people with disabilities to perceive, understand, navigate, and interact with the computer system or interface. There are millions of people who have disabilities that affect their use of technology⁶. Currently most systems have accessibility barriers that make it difficult or impossible for many people with disabilities to use their interfaces [Kirkpatrick et al., 2006] Accessibility depends on several different components of development and interaction working together, including software, developers, and users. The International Standards Organisation (ISO) 9241-171:2008 [9241-171:2008, 2010] which covers issues associated with designing accessible software for people with the widest range of physical, sensory and cognitive abilities, including those who are temporarily disabled, and elderly people. There are also other organisations that have produced guidelines, but, the ISO guidelines are more complete and cover the key points of all the others. There are however, no homogeneous set of guidelines that developers can easily follow. Disabled people typically use assistive technologies, hardware and software designed to facilitate the use of computers by people with disabilities, to access interfaces in alternative forms such as audio or Braille.



Figure: LiveView Cycle Gadget.
LiveView Cycle Gadget. —Image Credit:
<http://www.londoncyclist.co.uk>.

Pre-eminent when discussing accessibility is visual impairment and profound blindness, however, it should be remembered that visual disability is just one aspect of accessibility and technology. Other disabilities also exist which require changes to technology to enable a greater degree of accessibility. Before considering these, though, it is important to consider just what accessibility really means and why it is necessary to think about 'access' when building interfaces. Many disabled users consider technology to be a primary source for information, employment and entertainment. Indeed, from questioning and contact with many disabled users we have

discovered that the importance of technology cannot be under-estimated.

"For me computer systems are everything. They're my hi-fi, my source of income, my supermarket, my telephone. They're my way in."

This quote, taken from a blind user, sums up the sentiments we experience when talking with many disabled users and drives home the importance of accessibility in the context of independent living. Indeed, by making technology accessible, we help people live more productive lives.

So how does this work in practice? Disabled users, use intermediate technologies to provide access to technology and computer functionality. In general this technology exists to provide access to any generic software on the computer, however this is not al-

⁶Indeed, we could think of all users on a continuum of personal difference and requirements (which psychology alludes to in that disciplines study of individual difference and Differential Psychology) include cognitive differences (disputed by some but known as cognitive style).

ways the case for specialist systems. These 'Assistive Technologies' form a bridge from the user to the computer, much like a keyboard and monitor, the keyboard provides input, the monitor, output. In the case of say physical disability the input device, the keyboard, is not accessible to the user and so a software keyboard is substituted. This keyboard can then be tailored depending on the physical characteristics of the user. A mouse or joystick may be used to place the cursor, or a simple switch is used to interact with a software keyboard exhibiting a 'scanning' behaviour. However, some technology tends to be seen as a special case, this means that in some cases assistive technology exists as a solution just to the unique problems found when interacting with specialist systems as opposed to more general system accessibility [Harper and Yesilada, 2008]. Most assistive technology focuses on output, and so assistive technology for users with input problems, such as profound physical disability, tends to be focused on the computer and software in general and does not normally impact on specialist accessibility domains, such as Web accessibility. This output accessibility is developed independently along two fronts, both as part of interface design along with assistive technology on the user's computer. However, due to the design effort needed to build interfaces, the focus is primarily on the accessibility of the interaction or the tailoring of the interactive components to a specific audience. Indeed, this latter case tends to be the focus of accessibility for cognitive disability and hearing impairment.

It is my firm belief that internationalisation, also known as I18n, and culture can be barriers to the accessibility of software and so could be addressed under the border category of accessibility. However in most contexts internationalisation along with localisation are mainly seen as technical issues requiring multilingual translation of language components. Indeed, in this regard culture, which can also be a barrier to access, is often not addressed at all. For instance, in the mid-90s some users of the World Wide Web did not want to select the hyperlinks. This was because the links were blue, and blue was seen as an inauspicious colour. So then, internationalisation, in the broader sense, is the design and development of a product, application or interfaces that enables easy localization for target audiences that vary in culture, region, or language. In effect this means enabling the use of Unicode, or ensuring the proper handling of legacy character encodings where appropriate, taking care over the concatenation of strings, avoiding dependence in code of user-interface string values, or adding support for vertical text or other non-Latin typographic features. This also means enabling cultural conventions such as supporting: variance in date and time formats; supporting local calendars; number formats and numeral systems; sorting and presentation of lists; or the customary handling of personal names and forms of address.

In summary, there is no 'catch-all' solution to providing accessibility to computer-based resources for disabled users. Each disability requires a different set of base access technologies and the flexibility for those technologies to be personalised by the end-user. To date, the main focus has been on visual impairment. However, there are moves to address accessibility for hearing and cognitive impairments. Effective interaction, with the computer system or interface, requires that firstly the system can be accessed, secondly, that once accessed the system can be used effectively, and thirdly, that once used effectively the experience of usage is pleasurable and familiar. Keeping in mind, these three overriding priorities will enable you to design better interfaces and interactive environments for the user.

10.2 Barriers to Effectual Use

Often, when we start to build interfaces or design systems we can find it helpful to place ourselves at the centre of the process. Designing for ourselves enables us to think more easily about the problems at hand, and the possible cognitive or perceptual aspects that the interface will be required to fulfil [Chisholm, 2008]. While this is perfectly acceptable as a way of creating an initial prototype. Indeed, this is becoming increasingly known as 'Autobiographical Design' [Neustaedter and Sengers, 2012, Neustaedter and Sengers, 2012a] – it is not acceptable as the design progresses. This is because there are many different kinds of people and these people have many different types of sensory or physical requirements; indeed, this is where [scenarios and personas come in handy](#). Remember, it is your job to understand these requirements and keep them mind in all aspects of your interface or system designs.

10.2.1 Visual Impairment

Blindness, visual disability, visual impairment, or low vision means that the user will not be able to make adequate use of visual resources. In most cases blindness does not mean that the user can see absolutely nothing, but that aspects of their vision do not work correctly and cannot be corrected with lenses. The World Health Organisation (WHO) defines low vision as visual acuity⁷ of less than 6/18 (20/60), but equal to or better than 3/60 (20/400), or corresponding visual field loss to less than 20 degrees, in the better eye with best possible correction. Blindness is defined as visual acuity of less than 3/60 (20/400), or corresponding visual field loss to less than 10 degrees, in the better eye with best possible correction. This means that a legally blind individual would have to stand around three metres from an object to see it. According to the most recent estimates of the global burden of visual impairment, more than 161 million are blind. This means that if we are to support users with low vision or blindness we need to make sure that two things can happen when thinking about the interface. The first is that the screen can be magnified and that components which are necessary for interaction can be located in a physically similar location so that, when zoomed, they are usable together. Secondly, the interface should conform to general accessibility guidelines and principles meaning that each visual object has a corresponding textual description, and that the textual description can be accessed.

10.2.2 Cognitive Impairment

Cognitive impairment, also associated with learning difficulties, is one of the most pernicious barriers to human interaction with computers. This is mainly because aspects of cognitive disability not yet well understood and the spectrum of cognitive disability is so widespread that simply following a set of design criteria cannot assist in creating interfaces suitable for users with cognitive and learning impairments. It is therefore difficult to present any accurate figures relating to cognitive impairment as

⁷Visual acuity (VA) is acuteness or clearness of vision, more technically, a measure of the spatial resolution of the visual processing system. In the expression, 20/40 vision, the 20 is the distance in feet between the subject and the chart. The 40 means that the subject can read the chart as well as a normal person can read a chart that is 40 feet away. This is calculated by finding the smallest optotype they can identify and calculate the distance at which it has a visual angle of 5 arcminutes. A vision of 20/20 is considered nominal performance for human distance vision. A vision of 20/40 is considered half as good as nominal performance. A vision of 20/10 is considered twice as good as nominal performance – Wikipedia.

they change so rapidly and the definition is in some cases fluid. In general, users with cognitive impairments may have difficulties with executive function, memory, attention, visual and spatial perception, language, and emotions. Indeed it is more likely, that to a greater or lesser extent, minor aspects across most of these categories may be present. However, studies conducted in the USA, suggest that around 56% of people with cognitive impairment do enter the labour market and of those a fair percentage are employed in the use of computation devices. Therefore, the UX'er must also keep in mind the needs of this type of user when building systems interfaces; paying particular attention to learnability. Indeed, one of the key things to remember when designing interfaces and systems is to try and use as simple a language as possible, and where appropriate, annotate the interface with consistent graphical symbols.

10.2.3 Hearing Impairment

Hearing impairment is a broad term used to describe the loss of hearing in one or both ears. Hearing impairment refers to complete or partial loss of the ability to hear, with the level of impairment being either mild, moderate, severe or profound. Deafness, on the other hand, refers to the complete loss of ability to hear from one or both ears. Of the two types of hearing impairment, Conductive and Sensorineural. Conductive occurs when there is a problem conducting sound waves anywhere along the route through the outer ear, tympanic membrane (eardrum), or middle ear (ossicles). Where Sensorineural is usually permanent, and is caused by excessive noise or ageing, and requires rehabilitation, such as with a hearing aid. According to WHO estimates, 278 million people worldwide have moderate to profound hearing loss in both ears, with many living in low to middle-income countries. Also, the number of people worldwide with all levels of hearing impairment is rising. This is mainly due to a growing global population and longer life expectancies within those populations. In this case there are a number of aspects to consider when thinking about the user interface. Using sound alone to convey status information is inappropriate, indeed, relying solely on audio to convey any information should be avoided. Where possible captioning and text should be used. However, the UX'er should also remember that written or spoken language is not the native language of hearing-impaired users. In this case, sign language has become the accepted form of communication within this community, and indeed sign language is also a cultural requirement for many hearing-impaired or deaf users. Communication then, within the deaf community can be thought of as a combination of internationalisation and culture.

10.2.4 Physical Impairment

Physical Impairment affects a person's ability to move, and dexterity impairments are those that affect the use of hands and arms. Depending on the physical impairment, there are many challenges that affect users' interaction such as pointing to a target or clicking on a target. There are two kinds of impairments that affect interactivity: (a) musculoskeletal disorders that arise from loss, injury or disease in the muscle or skeletal system such as losing all or part of a hand or arm; and (b) movement disorders that arise from a damage to the nervous system or neuromuscular system such as Parkinson's disease which cause slowness of movement. Physical impairment is widespread mainly because many factors can bring on its onset. Indeed, one of the

largest surveys conducted in the United Kingdom estimated that 14.3% of the adult population had some mobility impairment that includes locomotion, reaching and stretching and dexterity. You should realise that people with motor impairments use a variety of creative solutions for controlling technology. These include solutions such as alternative keyboards and pointing devices, voice input, keyboard-based pointing methods and pointing based typing methods. Indeed, it can be seen that some of these solutions are also useful for users without a physical impairment, but who may be hindered by the size of their computational device or the environment in which it must be used.

10.2.5 Situational Impairment

Situational Impairment is not actually an impairment of the person directly, but indirectly by the computational device or the environment in which it must be used. Situation impairment is a recent phenomenon, indeed, it was first defined by Sears and Young in 2003. They point out that non-disabled individuals can be affected by both the environment in which they are working and the activities in which they are engaged, resulting in situationally-induced impairments. For example, an individual's typing performance may decrease in a cold environment in which one's finger does not bend easily due to extended exposure at low temperature. Anecdotal evidence also suggests that there are strong similarities between physical usability issues in both small-devices, such as a Mobile Telephone, Personal Digital Assistant, or any other hand-held device with a small keyboard and display, and accessible interaction scenarios. Both small-devices and accessible interfaces share the need to support various input techniques; they both benefit from flexible authored and accessible interfaces, and text entry and navigation in both scenarios are slow and error-prone. As a UX specialist, we can assume that different forms of disability have similarities with common situational impairments present in mobile situations. Many factors may, therefore, affect how users interact with mobile interfaces and these must be taken into account when building interfaces for these kinds of interactive scenarios.

10.2.6 Combinatorial Impairment

Population demographics indicate that our populations are ageing across the board. Indeed, the world's older population is expected to exceed one billion by 2020. Evidence suggests that approximately 50% of the older population suffer from impairments, such as hearing loss, with one in five people over the age of 65 being disabled. As the population ages the financial requirement to work more years is increased, but age-related disability becomes a bar to employment. At present, only 15% of the 65+ age-group use computers, but as the population ages this number will significantly increase. An ageing, but computer literate, population indicates a large market for computer-based services especially when mobility is a problem for the user. In many developed countries, the growth of the knowledge economy and a move away from manual work should improve the prospects of older Web users being able to find employment, providing technology, and specifically computational resources, is accessible to them. The aspects of impairment that define ageing is those of a combinatorial nature. In effect, ageing users have problems found in one or more of the groups listed in this section, but often, these impairments are less severe but more widespread.

Hummm... In the paragraph above I imply ageing is a barrier and combinatorial disabilities are its cause. But am I right to make such sweeping generalisations, swept along by a prevailing western research mindset⁸ [Lunn and Harper, 2011]?

Well, my mind drifted back to a paper by Peter G. Fairweather from IBM T.J. Watson called 'How older and younger adults differ in their approach to problem-solving on a complex website'.

Peter challenges the view that "Older adults differ from younger ones in the ways they experience the World Wide Web. For example, they tend to move from page to page more slowly, take more time to complete tasks, make more repeated visits to pages, and take more time to select link targets than their younger counterparts. These differences are consistent with the physical and cognitive declines associated with ageing (sic)."

The picture that emerges has older adults doing the same sorts of things with websites as younger adults, although less efficiently, less accurately and more slowly."* He presents new findings that show that "to accomplish their purposes, older adults may systematically undertake different activities and use different parts of websites than younger adults. We examined how a group of adults 18 to 73 years of age moved through a complex website seeking to solve a specific problem.

We found that the users exhibited strong age-related tendencies to follow particular paths and visit particular zones while in pursuit of a common goal. We also assessed how experience with the web may mediate these tendencies."* [Fairweather, 2008]

The interesting part is that Peter finds that younger and older people do things differently with older users taking a less risky route, but also that experience, not age, is the defining factor of regarding how to solve the assigned problem using the web. Younger inexperienced users make more mistakes than older experienced users and take longer. So what does this say about our conception of older users having problems found in other disability groups – ageing as disability, ageing as impairment?

I'm beginning to think that ageing, in general, isn't a barrier to access although combinatorial disability – at whatever age it occurs – may well be. But it is true that low-level combinatorial impairments affect older users more, proportionally than others. However, in all cases, a lack of experience and knowledge are much more significant barriers to access, so education is key. It seems we have got into a habit of using ageing as a proxy term for combinatorial disability, its inaccurate and we should stop it.

10.2.7 Illiteracy

Education is the key to Illiteracy. The United Nations Educational, Scientific and Cultural Organisation (UNESCO) defines literacy as 'The ability to identify, understand, interpret, create, communicate, compute and use printed and written materials associated with varying contexts. Literacy involves a continuum of learning to enable an

⁸We conclude the same in some recent work we have just completed.

individual to achieve his or her goals, to develop his or her knowledge and potential, and to participate fully in the wider society.' Literacy comprises a number of sub-skills, including phonological awareness, decoding, fluency, comprehension, and vocabulary. Mastering each of these sub-skills is necessary for learners to become proficient readers. However, literacy rates can vary widely from country to country or region to region. This often coincides with the region's wealth or urbanisation, though many factors play a role, such as social customs and education. As a UX specialist you must be aware that it is possible your software may be used in circumstances where illiteracy is prevalent. Indeed, studies suggest that around 20% of the world's population are illiterate and so there is a high chance that your software will be used by someone who will have some difficulty in understanding the labels, prompts, textual components, or content of the interface. In this case you should try to anticipate this aspect by adding images and sound, and making the interface as flexible as possible so that it can be more easily adapted to the users needs.

10.2.8 Developing Regions

Computer use in developing regions is currently characterised by constrained operating modalities. Slow speed, low computational power, reduced bandwidth, compact keyboards, small screens, and limited power, all compound the problem of access and inclusion. Also, interaction is sometimes without conventional written language and illiteracy is also a barrier to information and services. However, the benefits of computer technology are so great that the peoples of these regions often adopt resourceful methods of interaction and access sometimes repurposing technical equipment so that it is put to a different use than for which it was intended and sharing access often on a community-wide basis. While computational use in developing regions may not seem directly related to user experience, the interface considerations and its adaptability are important within this context and do indeed speak to the wider human factors domain. In this case, it is useful to keep in mind the possible problems that may be encountered in constrained operating modalities and the possible requirements for interface adaptation.

10.2.9 Exclusion

The opportunities created by digital technologies are not enjoyed by the whole of society, indeed, there is a strong correlation between digital exclusion and social exclusion. There are significant and untapped opportunities to use technology better on behalf of citizens, communities, and digitally disenfranchised groups. However to achieve inclusion, systems must be created seeing the human factor as a part of an integrated solution from the outset, not as an adjunct but also not as a focus. In addition, the multiplicity and ubiquity of devices and their interfaces are key to successful inclusion, and systems must be tailored to what users require and will use; as opposed to what organisations and government require and use.

10.3 Technical Accessibility Issues

As we have already discussed, pre-eminent when discussing accessibility is visual impairment and profound blindness. However, it should be remembered that visual disability is just one aspect of software accessibility. Other disabilities also exist which require changes to the interface to enable a greater degree of accessibility. Since the 1980s, visually impaired people have used computers via screen-readers. During the 1980s, such applications relied on the character representation of the contents of the screen to produce spoken feedback. In the past few years, however, the widespread trend towards the development and use of the GUIs has caused several problems for profoundly blind users. The introduction of GUIs has made it virtually impossible for blind people to use much of the industry's most popular software and has limited their chances for advancement in the now graphically oriented world. GUIs have left some screen-readers with no means to access graphical information because they do not use a physical display buffer, but instead, employ a pixel-based display buffering system. A significant amount of research and development has been carried out to overcome this problem and provide speech-access to the GUI. Companies are now enhancing screen-readers to work by intercepting low-level graphics commands and constructing a text database that models the display. This database is called an Off-Screen Model (OSM), and is the conceptual basis for GUI screen-readers currently in development.

An OSM is a database reconstruction of both visible and invisible components. A database must manage information resources, provide utilities for managing these resources, and supply utilities to access the database. The resources the OSM must manage are text, off-screen bit maps, icons, and cursors. Text is obviously the largest resource the OSM must maintain. It is arranged in the database relative to its position on the display or to the bit map on which it is drawn. This situation gives the model an appearance like that of the conventional display buffer. Each character must have certain information associated with it: foreground and background colour; font family and typeface name; point size; and font style, such as bold, italic, strike-out, underscore, and width. Merging text by baseline (i.e. position on the display) gives the model a 'physical-display-buffer' like appearance with which current screen-readers are accustomed to working. Therefore, to merge associated text in the database, the model combines text in such a way that characters in a particular window have the same baseline. Each string of text in the OSM has an associated bounding rectangle used for model placement, a window handle, and a handle to indicate whether the text is associated with a particular bit map or the display. GUI text is not always placed directly onto the screen. Text can be drawn into memory in bit-map format and then transferred to the screen, or can be clipped from a section of the display and saved in memory for placement back onto the screen later.

In windowing systems, more than one application can be displayed on the screen. However, keyboard input is directed to only one active window at a time, which may contain many child windows; the applications cursor is then placed in at least one of these child windows. The OSM must keep track of a cursor's window identification (i.e. handle) so that when a window becomes active the screen-readers can determine if it has a cursor and vocalise it. Additionally, the OSM must keep track of the cursor's screen position, dimensions, the associated text string (i.e. speakable cursor text), and character string position. If the cursor is a blinking insertion bar, its character position

is that of the associated character in the OSM string. In this case, the cursor's text is the entire string. A wide rectangular cursor is called a selector since it is used to isolate screen text to identify an action. Examples of selector cursors are those used for spreadsheets or drop-down menus. The text that is enclosed within the borders of the selector is the text that the 'Screen Reader' would speak.

Modifying current screen-readers to accommodate the new GUI software is no easy task. Unlike DOS, leading GUI software operates in multitasking environments where applications are running concurrently. The screen-reader performs a juggling act as each application gets the user's input. However, many frameworks and application bridges have been created to help the operating system accomplish this 'juggling'.

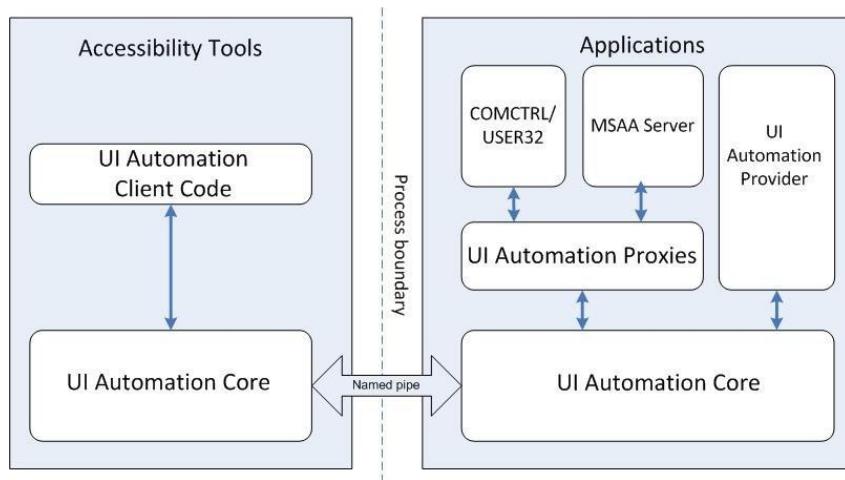


Figure: Interaction Schematics for Microsoft's 'UI Automation'. Interaction Schematics for Microsoft's 'UI Automation'. —Image Credit: Wikimedia.

10.3.1 MSAA and UIA

Microsoft Active Accessibility (MSAA) is a developer technology that improves the way programs and the operating system work with accessibility aids. MSAA can help software developers make their programs more compatible with accessibility aids, and accessibility aid developers can make more reliable and robust aids. MSAA is an underlying technology that is invisible to users and provides a standard way for accessibility aids to obtain information regarding user interface elements. This standardisation helps software developers and assistive technology developers alike to ensure their products are compatible mainly on the Windows platform. This standard also gives software developers more flexibility in designing their application interfaces by understanding the needs of assistive technologies. In this way, UX specialists can innovate more freely without sacrificing compatibility and have the confidence that the underlying application logic and interface routines will integrate into the accessibility frameworks used by assistive technology developers. Microsoft UI Automation (UIA) (see [Figure: Interaction Schematics for Microsoft's 'UI Automation'](#)) is similar to MSAA in that it provides a means for exposing and collecting information about user interface elements and controls to support user interface accessibility and software test automation. However, UIA is a newer technology that provides a much richer object model than MSAA and is compatible with new Microsoft Technologies such as the '.NET' Framework.

10.3.2 IAccessible2

IAccessible2 is a new accessibility API, which complements Microsoft's earlier work on MSAA which fills critical accessibility API gaps in MSAA by combining it with the UNIX Accessibility ToolKit (ATK). A detailed comparison of MSAA to ATK revealed some missing interfaces, events, roles and states within the MSAA framework and so IAccessible2 was created out of a desire to fill these gaps. Therefore, IAccessible2 is an engineered accessibility interface allowing application developers to use their existing MSAA compliant code while also providing assistive technologies access to rich document applications. IAccessible2's additional functionality includes support for rich text, tables, spreadsheets, Web 2.0 applications, and other large mainstream applications. In this case the Linux Foundation's Open Accessibility Workgroup summarises IAccessible2 this as "*complementing and augmenting MSAA by responding to new requirements; it was designed to enable cost effective multi-platform development; it supports Web 2.0 (AJAX, DHTML); it is a proven solution; its effectiveness has been demonstrated on both sides of the interface, by large applications like the IBM Workplace Productivity Editors and Firefox and by the leading AT vendors; and finally, it is a royalty free and open standard, free for use by all, open for review and improvement by all.*" In this case, it seems more appropriate that software developers think seriously about using IAccessible2 as a means of providing accessibility at the interface level.

10.3.3 Interface Bridges

In an attempt to solve the accessibility problems of assistive technology and platform independent software, Sun Microsystems (Sun – now Oracle) introduced the Java Accessibility API, which was designed as a replacement for the OSM. With a Java application that fully supports the Java Accessibility API, no OSM model is necessary because the aim of the API is to provide all of the information normally contained in an OSM. For existing assistive technologies available on host systems (e.g. Microsoft Windows, OSX, Linux) to provide access to Java applications, they need some way to communicate with the Java Accessibility support in those applications. The Java Accessibility Bridge supports that communication. The Access Bridge (see [Figure: JVM, Java Accessibility API and Access Bridge](#)) makes it possible for an OS based assistive technology like JAWS to interact with the Java Accessibility API. The Java Accessibility API is implemented in the Swing user interface components. The Access Bridge is a class that contains 'native methods', indeed, part of the code for the class is actually supplied by a dynamically linked library (DLL) on a Windows host system. The assistive technology running on the host communicates with the native portion of the bridge class. This bridge class in turn communicates with the Java Virtual Machine, and from there to the Java Accessibility utility support and on to the Java Accessibility API of the individual user interface objects of the Java application it is providing access to.

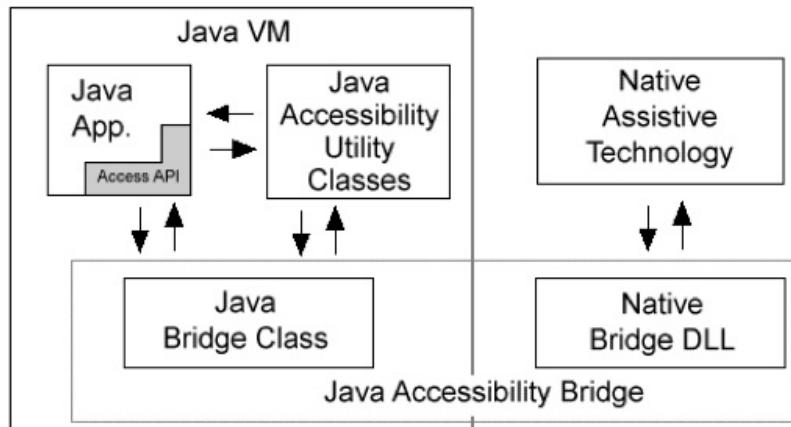


Figure: JVM, Java Accessibility API and Access Bridge. How the JVM, Java Accessibility API and Access Bridge work together with the host system to provide the relevant information to an assistive technology.. –Image Credit: SUN.

The Java Accessibility API defines a contract between individual user-interface components that make up a Java application and an assistive technology that is providing access to that Java application, through the Accessible interface. This interface contains a method to get the AccessibleContext class, which contains the common core set of accessibility information that every user-interface object must provide. If a Java application fully supports the Java Accessibility API, then it should be compatible with assistive technologies such as screen-readers. For example, in order for ‘JAWS’ to provide access to Java applications running on Microsoft Windows, it would make calls to the Java Accessibility Bridge for Windows. When the user launches a Java application, the bridge would inform JAWS of this fact. Then JAWS would query the bridge and the bridge would in turn forward those queries on to the Java Accessibility Utilities that were loaded into the Java Virtual Machine. When those answers came back to the bridge, the bridge would forward them on to JAWS.

10.3.4 Platform Independent Software

We can see that the problem of platform independent software is easily surmountable by using technologies like the accessibility API and Bridge, but faults persist. The nub of the problem of platform independent developments is the fact that an additional level of indirection is present when compared with native developments. With a native development information regarding text, titles, window, and component status are implicitly accessible by the native OS and so in many cases this information does not need to be explicitly provided. This is not the case with platform independent software. In this case, the bridge component provides an intermediary step by implementing an OS native end along with a conduit to the Java native end. However, the application still cannot correctly resolve accessibility information unless it is made explicit. In summary, the Java Virtual Machine (JVM), Java Accessibility API, and the Java Access Bridge work together with the host system to provide the relevant information to an assistive technology. Because ‘implicit’ hooks are not available, a failure at the programming level to make information explicit can invalidate the entire pipeline and break the accessibility of the application. As we are seeing an increase in platform independent languages, such as ‘Ruby’, ‘Tcl’, ‘Perl6’, and the like, the software engineer

and UX'er must be aware of the additional programming and testing required to make the interface accessible.

10.4 Potted Principles of Effectual User Experience

Conforming slavishly to a set of principles or success/conformance criteria – with our brain switched off – is never going to be a good idea. Indeed, it is such a bad idea that [heuristic evaluation](#) was created, to catch the errors introduced while trying to conform in the design and build phase. This said principles were useful if applied from a high level, and success criteria at a low level, or at least as cues you should be thinking about as you go through the design and development process. These are useful and will eventually reduce the number of negative user experiences encountered within the testing and validation phase.

There are many different principles and conformance criteria ranging from those described in the common texts to those described in the international standards [\[9241-20:2008, 2008\]](#). Indeed, standards documents seem to be very in-depth and have many checkpoints and bulleted lists that you must individually assess and validate, common texts seem to specify only high-level principles without the detail covered in the standards documents. In this case there seemed to be no middle path. However, it is my opinion that the high-level principles can be discussed and accompanied with a series of questions the UX'er can keep in mind as the design and development proceeds. In this way, a constant mental questioning-framework is built in the UX'ers psyche which enables flexible critique based on the principles.

In my opinion that the most important considerations for effective and accessible interactions are those of openness, perceivability, operability, understandability, and flexibility. Keeping these five principles in mind – along with the questions accompanying each principle – will enable you to better: conform your designs and developments to acknowledged good practice, maintain your questioning nature, and be able to retrospectively apply these principles when it comes to heuristic evaluation and software audits.

However, you must remember that experience is everything. The way to truly understand these principles, and the questions contained within them, is to apply them in real life, develop them in real life, and test them in real life. Remember, memorising these principles does not make you an expert, applying them many times does.

10.4.1 Facilitate Openness

By making sure that the system is open, and that aspects of the interface and human facing system components can be modified, adapted, or interacted with via an application programming interface (API), the developer is making an explicit statement which allows for extensibility. This means that all aspects of the application can be modified and changed once further information comes to light or in the event of processes in the organisational systems of the commissioner changing (for example see [Figure: Durateq ATV](#)).

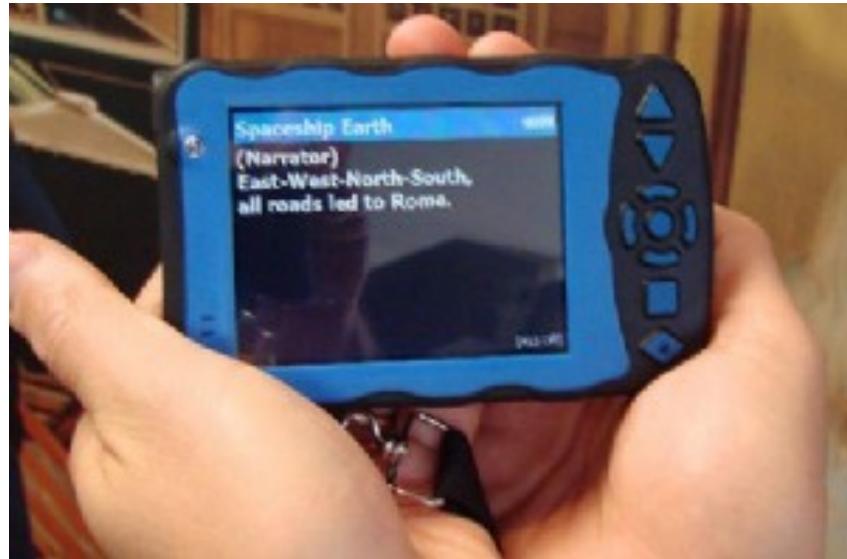


Figure: Durateq ATV. The Durateq ATV is customised for assistive technology applications, and supports Closed Captioning, Assistive Listening, and Descriptive Narration. —Image Credit: Durateq.

Open software means that bugs can be more easily fixed by an army of interested parties who may also want to use the software for their own purposes. Additions to the system can also back-flow to the originators work such that an enhanced system can be created without any additional effort on the originators part. Obviously there are problems with systems which the commissioner requires to be closed or which uses proprietary or bespoke data formats and interface specifications. However by leaving the system open to an extension, the developer can ensure a fast turnaround on any recommissioned work.

Questions to think about as you design your prototype:

1. If you have provided succinct, meaningful name and label for each interface component – including alternatives for icons, and the like – are these names displayed and available to third-party technology?
2. Is your software open such that its information is available to third part systems including conventional assistive technologies?
3. Does your software allow input and manipulation by these third party systems?
4. If your systems are closed then, does it allow all input and output to pass through to third-party software?

10.4.2 Facilitate Perceivability

Critically, the user interface and human facing aspects of the development should be perceivable by many different types of user (for example see [Figure: A Braille Display](#)). It is obvious that the general software engineer cannot account for all aspects of perception and personalisation that may be required by an individual user. However, by ensuring that all information regarding interfaces components, the presence of alternative descriptions for specific widgets or components, and that this text has some semantic meaning, the developer can make sure that tools to translate between a standard interface and one which may require modification can occur much more easily.



Figure: A Braille Display. A Braille Display. A refreshable Braille display is an electro-mechanical device for displaying Braille characters, usually using raising dots through holes in a flat surface. Blind computer users, who cannot use a normal computer monitor, use it to read text output. Speech synthesisers are also commonly used for the same task, and a blind user may switch between the two systems or use both at the same time depending on circumstances. —Image Credit: DBSV / Max Lautenschläger.

Questions to think about as you design your prototype:

1. Does your system convey information by colour alone (such as a button turning red) but without an alternate textual description?
2. Is there a succinct, meaningful name and label for each interface component – including alternatives for icons, and the like?
3. Can the user personalise the interface settings, including different styles and sizes of typography, fore and background colours, and default interface look and feel?
4. How easy is it to find and interact interface controls and the shortcuts to them?
5. Have you provided keyboard focus and text cursors, and will your system restore its previous state when regaining focus?

10.4.3 Facilitate Operability

As with perceivability, operability is also very important. As a UX'er you should realise there are many different kinds of user requirements when it comes to interfacing with the software. However, most systems which enhance or override standard operability systems use the keyboard buffer as a universal method of input. In this case you should make sure that all aspects of the functionality of the software, including navigation and control of the components, can be controlled via a series of keystrokes. For instance I am currently dictating this section, which means that I must have the Mac OS X accessibility extensions switched on, then the speech software can translate my speech into keystrokes and places those keystrokes within the keyboard buffer. To the application it seems as though someone is typing using the keyboard and therefore all interaction is handled accordingly. By supporting operability, device manufacturers can be assured that your software is operable regardless of the input modality.

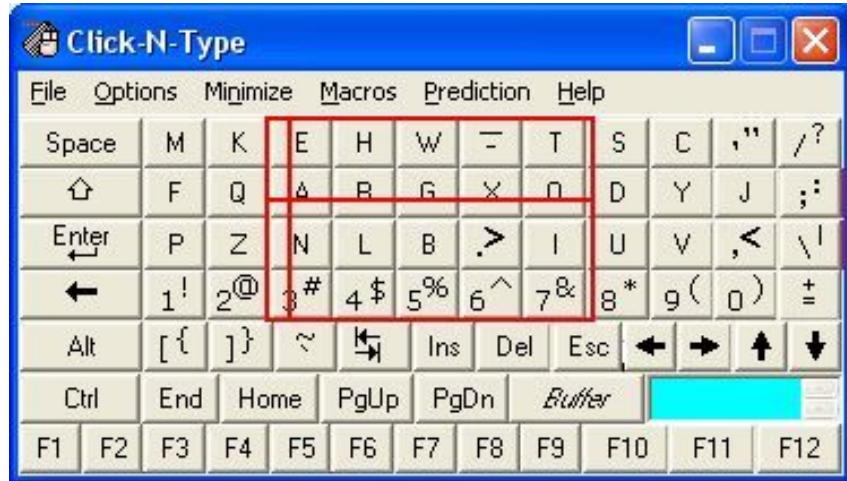


Figure: Click-N-Type Scanning Keyboard. A Scanning Keyboard; normally a cursor moves across the top of each column (referred to as column scanning). To select a character the user presses the switch when the cursor is over the column containing that entry. At which point the cursor starts down the menu column highlighting each character in that column as it moves. When the desired character is highlighted, the user presses the switch to select that character. —Image Credit: Click-N-Type.

Questions to think about as you design your prototype:

1. Can the user control all aspects of an audio - visual presentation (such as a video)?
2. Can the user personalise the interaction settings, and change the length of any timed events?
3. Are all functions/interface components/interactions able to be controlled via the keyboard?
4. Are all functions/interface components/interactions able to be controlled via a mouse-like pointer controller?
5. Are all multiple chorded key presses available in a sticky-key sequential form?
6. Can the keyboard be remapped?
7. Have you provided keyboard focus and text cursors, and will your system restore its previous state when regaining focus?

10.4.4 Facilitate Understandability

Systems need to be understandable. This is especially important when it comes to the interface and the language used within that interface, including error messages, exception handling, and the system help. Using language that is simple to understand and supporting foreknowledge as to the outcomes of various choices the user may be asked to make; will assist in how usable the system becomes and performance increases. In this case, the systems developers should use jargon-free language that can be easily understood by the layperson and that has been tested with people who do not have a direct understanding of the computational aspects of the system. However, there are exceptions to this rule. Consider a bespoke piece of software created for an organisation that has a set of specific institutional abbreviations along with a specific knowledge of what these abbreviations, or linguistic shorthand, means. In this case, it may be appropriate to use terminology that is familiar to organisational users of the system but would be unfamiliar to a wider audience. In reality, interfaces are understandable if computational and developmental jargon is not used and if there

is a clear statement as to the outcome is of any actions or decisions that the user is required to make. Further, the concept of ‘affordances’ [Gibson, 1977] – the quality of an object, or an environment, which allows an individual to perform an action (a knob affords twisting, a cord affords pulling) – can also be useful for implicitly/tacitly conveying information.

Questions to think about as you design your prototype:

1. Is there a succinct, meaningful name and label for each interface component – including alternatives for icons, and the like?
2. Are all interface control and interaction features well documented?
3. Do all error or alert messages give consistent information, appear in a consistent place, and facilitate movement to the site of those errors?
4. Do you follow the OS keyboard conventions and best practice and style guides?
5. Have you tied into the system wide spell checker?



Figure: Guardian's Adaptive Website. The Guardian's Adaptive Website, allows delivery to different form factor devices. —Image Credit: Guardian's Adaptive Website.

10.4.5 Facilitate Flexibility

We cannot always plan and understand all possible actions and eventualities that will be required by an interface. Users are highly personal and the interactive requirements are different between users and also change over time for individuals. Requirements for the system may also change and requirements for the human facing aspects of the functionality may likewise change as the system becomes more familiar to the user. In this case, it is good practice to try and ensure flexibility within the interface design such that it is possible to highly tailor the operations of the human facing aspects of the system (for example see [Figure: Guardian's Adaptive Website](#)). This flexibility, however, leads to increased complexity within the design and likewise an increased pressure is placed on time and cost. If flexibility cannot be easily achieved in a cost-effective or-time effective manner, then it may be more appropriate for a series of software hooks

to be left in the underlying design is such that changes can be made and the system can be modified if required at a later date.



Figure: Qt Development. Nokia's Flexible Qt allows you to write advanced applications and UIs once and deploy them across desktop and embedded operating systems without rewriting the source code saving time and development cost. —Image Credit: Nokia.

One key aspect of flexibility is the ability to adapt your system on the fly, based on a device description, or to build in the ability to conform to different devices. Remember we talked about '[Model View Controller Architectures](#)' way back, well this separation of concerns enables real-world frameworks, such as Nokia's Qt – in the development cycle, the separation of concerns facilitates flexibility of the interface and, therefore, enables adaption to a target device (for example see [Figure: Qt Development](#)).

Questions to think about as you design your prototype:

1. Does your system allow users to conform the colour encoding to their preferences?
2. Can the user personalise the interaction settings, and change the length of any timed events?
3. Can the user personalise the interface settings, including different styles and sizes of typography, fore and background colours, and default interface look and feel?
4. Can the keyboard be remapped?
5. Do you allow keyboard shortcuts and accelerator keys to be personalised to the user preferences?

Remember... these are just the first five of our total principles - for the [next batch you can skip ahead](#). But just because you can, maybe you shouldn't... take your time to read the rest!

Now I'm presuming you aren't building an operating system from scratch but are getting one out of the box. In this case, lots of stuff is already done for you (for example

see [Figure: OSX Accessibility Settings](#), so I've not listed it in the questions attached to each principle – that are all still valid anyhow. Now if you are creating one from scratch, maybe for a bespoke kiosk, auditory driver feedback system in a car, for an embedded system - you must also think about input and output of the OS via the hardware and devices attached to it. How will you set key-repeat intervals for people with tremors, how will you notify hearing impaired users if there is only auditory feedback? There are many more questions to be thought about, and so you should make sure your system is compatible with a modern accessibility bridge, provide ports for Bespoke I/O hardware, and take an in-depth look at ISO 9241-171:2008. What's more, you'll find that in some accessibility guidelines, [robustness which I cover later](#) – is often included; however I think of this as either part of the ‘utility’ / engineering of the system or as part of its efficient use⁹.

10.5 Summary

Ask any well-informed developer to rate how important accessibility is, and I'd expect most to rank it high. Ask them why, and I'd expect most to talk about disabled users, altruism, and the law. An enlightened few may talk about the business case, or expand on the many technical advantages of making their software accessible.

It may surprise many designers and developers to know that when they are developing accessible code they are also, more than likely, developing aesthetically pleasing code too. But this shouldn't be the most basic motivator for accessibility. Indeed, developers should also understand that accessible means mobile.

Designing and building effective systems aren't just about accessibility, but by being accessible, we can ensure effectual usage. Indeed, accessibility is now not just about making systems accessible to disabled users, but based on the belief that we are all handicapped in some way by the environment or the devices we use. We can see this in our earlier discussions on situational impairment, and in some cases, people have begun to refer to temporal impairment or temporary impairment that is our inability to use effectively our computational systems for short amounts of time-based on some characteristic of ourselves or the environment or the condition of both.

Understanding that interfaces must be transformable also supports users who are conventionally excluded. Currently, the opportunities created by computer technolo-

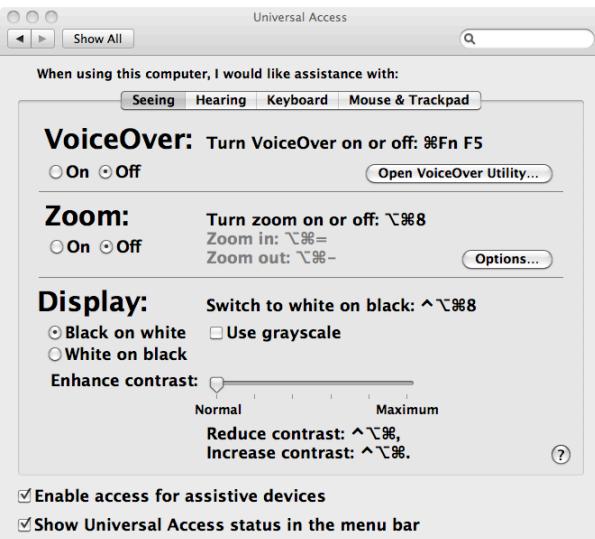


Figure: OSX Accessibility Settings. Mac OS X Accessibility Settings are built right into the Operating System. Notice that keystrokes need not originate at the keyboard – and that shortcuts exist for all commands. —Image Credit: Apple.

⁹After all, a system my still be used even if it fails every so often.

gies are not enjoyed by the whole of society. Indeed, there is a strong correlation between technological exclusion and social exclusion. There are significant and untapped opportunities to use technology better and on behalf of citizens, communities, and digitally disenfranchised groups. However to achieve inclusion, systems must be created seeing the user experience, not as an adjunct, but as a part of an integrated solution from the outset. We know that the multiplicity and ubiquity of devices and their interfaces are key to successful inclusion, households may very well have a games console or digital television, but no general purpose computer system. Being able to deliver content to any device, and support the users needs as opposed to the developers is key to making good interfaces that will be used, and which matter to real people.

Effective use is not just about disability, if anything it is more about a flexibility of mind at every level of the construction process from commissioning, through design and build, and on to evaluation. Effectual use accentuates good design and adaptability which helps future-proof your builds against changes in technologies, standards, and extensions to the design.

I cannot impress on you enough my belief that by understanding disabled user's interaction we can gain an understanding of all users operating in constrained modalities where the user is handicapped by both environment and technology. [UX testing and audit](#) including users with disabilities is a natural preface to wider human facing testing focused on effective use.

10.5.1 Optional Further Reading / Resources

- [M. Burke] How I use technology as a blind person! - Molly Burke (CC). YouTube. (2015, December 18). Retrieved August 10, 2022, from <https://www.youtube.com/watch...>¹⁰
- [W. Chisholm.] Universal design for Web applications. O'Reilly Media Inc., Sebastopol, CA, 2008.
- [A. Kirkpatrick] R. Rutter, C. Heilmann, J. Thatcher, and C. Waddell. Web Accessibility: Web Standards and Regulatory Compliance. Friends of ED, July 2006.
- [S. Harper and Y. Yesilada] Web Accessibility: A Foundation for Research, Volume 1 of Human-Computer Interaction Series. Springer, London, 1st edition, September 2008.
- [Y. Yesilada and S. Harper] Web Accessibility: A Foundation for Research, volume 2 of Human-Computer Interaction Series. Springer, London, 2nd edition, 2019.

10.5.2 International Standards

- [ISO 9241-171:2008.] Ergonomics of human-system interaction – part 171: Guidance on software accessibility. TC/SC: TC 159/SC 4 ICS: 13.180 / Stage: 90.20 (2011-07-15), International Organisation for Standardisation (ISO), Geneva, Switzerland, 2010.
- [ISO 9241-20:2008.] Ergonomics of human-system interaction – part 20: Accessibility guidelines for information/communication technology (ict) equipment and service. TC/SC: TC 159/SC 4 ICS: 35.180; 13.180 / Stage: 90.60 (2011-06-17), International Organisation for Standardisation (ISO), Geneva, Switzerland, 2008.

¹⁰<https://www.youtube.com/watch?v=TiP7aantnvE>

- [ISO 9241-129:2010.] Ergonomics of human-system interaction – part 129: Guidance on software individualisation. TC/SC: TC 159/SC 4 ICS: 35.180; 13.180 / Stage: 60.60 (2010-11-08), International Organisation for Standardisation (ISO), Geneva, Switzerland, 2010.
- [WHA 54.21] International Classification of Functioning, Disability and Health (ICF) - <http://www.who.int/classifications/icf/en/> .
- [UN ESCAP] UN ESCAP Training Manual on Disability Statistics - 2.3 ICF terminology and definitions of disability http://www.unescap.org/stat/disability/manual/Chapter2-Disability-Statistics.asp#2_3 .



Self Assessment Questions

Try these without reference to the text:

1. Give an example of why Accessibility is for everyone.
2. What is your view regarding ‘Combinatorial Impairment’?
3. Pick an interface bridge and describe it.
4. What is the relationship between effective and accessible design?
5. What are the five main principles of effective design?

11. Principles of Efficient Experience (Usability)

Universal usability is more a function of keeping all of the people and all of the situations in mind and trying to create a product which is as flexible as commercially practical, so that it can accommodate the different users and situations.

– Gregg Vanderheiden



Figure: Nest Learning Thermostat. Thermostat Showing Heating. –Image Credit: Nest Labs.

mobile telephones and personal digital assistants, is on the increase.

Let us take a case in point, the ‘Nest Learning Thermostat’ (see [Figure: Nest Learning Thermostat](#)) learns about the user and their home to balance comfort and conservation, without the hassle of programming or constant re-adjustments. Nest programs itself based on the temperatures the user sets, thereby learning the user’s personal schedule – in a week – and starts automatically turning down heating or cooling – to save energy – when that user is away³. Nest keeps refining its schedule over time and takes input by user rotation of the outer ring to adjust the temperature. The display turns blue when cooling and red when heating, push down opens the very simple menu to enable temperature changes deletion and other fine control (see [Figure: Nest Thermostat Control](#)).

¹Productive of effects; effective; adequately operative. The cause that makes effects be what they are (esp. of a system or machine) achieving maximum productivity with minimum wasted effort or expense — OED.

²The fact or quality of being usable (That can be used; that can be readily put to practical use) — OED. The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use — ISO.»

³There are other more aesthetic considerations – [which we will look at later](#) – in that the brushed stainless steel dial frames the display while the ring’s curved, neutral silver finish creates a chameleon effect that grounds Nest within its environment by picking up the colour of the wall upon which it’s mounted. This combination of sleek design elements and premium materials makes Nest a thermostat a user can feel proud to display in your home.

As we can see, this interface is easy to understand and operate, it uses a familiar modality – in that it works similar to a turnable knob – and removes what has become a complicated interaction⁴ into a very simple one.

(see Figure: Modern Thermostat)

This occurs because the complexity of the program functionality has been removed from the end user and into the system itself. This, obviously, makes the software more difficult to develop, but it removes the complexity from the user and places the onus on the software engineer and UX'er. This is important in a world where information and the requirement for its manipulation is increasing at an alarming rate, requiring the user to respond to this information and placing a premium on easy interactivity. Often, however, these informational resources are centralised and, therefore, need to be tailored to the target audience at the time that they are delivered. Complexity, centralisation, and the need to enhance the performance of the user means that the usability of the system, or interface, has become increasingly important.



Figure: Nest Thermostat Control. Thermostat Showing Fine Heating Control. — Image Credit: Nest Labs.



Figure: Modern Thermostat. Thermostat Modern Thermostat with an 'Impenetrable' Interface. — Image Credit: Wikipedia.

than it was before the interactive component was either altered or created when the interface design must be better as an enhancement has occurred. Common performance measures include: the time required by the user to complete a task; the time spent navigating the interface; the number of incorrect choices or errors created; the number of jobs completed, either correctly or incorrectly; the number of observations of user frustration; and finally the frequency of interface components or behaviour that is never used.

The **user centred design paradigm** was created to assist in the construction of these usable interfaces. In this regard, part of the design process is driven by users who are

⁴...not least because the requirement to interact with the thermostat is sporadic; therefore reducing the familiarity that would become a learned behaviour to a bespoke interface.

The human facing aspects of the system are based on an understanding of the interactive needs of the user and the way their psychology and cognition affect their interaction. This, in general, manifests itself as the ability to perform tasks and actions required by the interface in ways that are also easy for the user to comprehend and execute. Indeed, this is one of the primary driving forces of the graphical user interface and the mouse and keyboard. Often the usability of the system is measured by the performance, the time to execute a task, of a user enacting jobs over the system. As we shall see the rationale is if the task is completed faster

conscripted on to the design team. In this way, it is hoped that user-friendly systems can be built and the usability of systems increased. Unfortunately, it is often difficult to solicit a representative cross-section of the possible user group, and so usability for all is often missed. This has led to the popularity of universal usability especially within the information society. In this case, universal usability refers to the design of information and communications products and services that are usable for every citizen; [discussed to some extent](#) and [expanded upon later](#).

In reality, there is little consensus regarding the relationship of UX, ergonomics, or human factors to usability. Indeed, some think of usability

“... as the software specialisation of the larger topic of ergonomics. Others view these topics as tangential, with ergonomics focusing on physiological matters (e.g., turning a door handle) and usability focusing on psychological matters (e.g., recognising that a door can be opened by turning its handle).”

However, many experts have written separate, but overlapping, frameworks for aspects of usability which should be taken into account when designing and building systems interfaces ([we'll discuss these further](#)). However, before we start on these comparisons let's look at work that gave rise to some of the earliest principles by which we should design interfaces.

11.1 The Xerox 'Star'

The Xerox 'Star' was a commercial version of the prototypical Xerox Alto – if one thousand fully working systems, used internally at 'PARC' day-in-day-out over seven years, can be said to be prototypical⁵. While the system itself is an interesting development in computer science, the interface and the mode of user interaction is truly visionary. Star, adopted novel technologies such as the mouse (the Alto had both a mouse and a portrait monitor), operationalised developments such as the graphical user interface, and created novel technologies such as the desktop (see [Figure: Xerox Star Desktop](#)). However, as UX'ers, we are more interested in the design methodology they used and the principles derived from the team's experiences designing and developing the interface and interactivity of the implementations. These are some of the first usability principles to be mentioned in common computer literature – dating back to the early 1970s – and are still applicable today. First of all, let us look at the design methodology adopted by the Xerox Star team – it needs no further discussion from me, and so I reproduce it verbatim:

“One of the most troublesome and least understood aspects of interactive systems is the user interface. In the design of user interfaces, we are concerned with several issues: the provision of languages by which users can express their commands to the computer; the design of display representations that show the state of the system to the user; and other more abstract issues that affect the user's understanding of the system's behaviour. Many of these issues are highly subjective and are therefore often addressed in an ad hoc

⁵PARC definitely 'Eat Their Own Dog Food'!

fashion. We believe, however, that more rigorous approaches to user interface design can be developed..."

"These design methodologies are all unsatisfactory for the same basic reason: they all omit an essential step that must precede the design of any successful user interface, namely task analysis. By this, we mean the analysis of the task performed by the user, or users, before introducing the proposed computer system. Task analysis involves establishing who the users are, what their goals are in performing the task, what information they use in performing it, what information they generate, and what methods they employ. The descriptions of input and output information should include an analysis of the various objects, or individual types of information entity, employed by the user..."

"The purpose of task analysis is to simplify the remaining stages in user interface design. The current task description, with its breakdown of the information objects and methods presently employed, offers a starting point for the definition of a corresponding set of objects and methods to be provided by the computer system. The idea behind this phase of design is to build up a new task environment for the user, in which he can work to accomplish the same goals as before, surrounded now by a different set of objects, and employing new methods."*

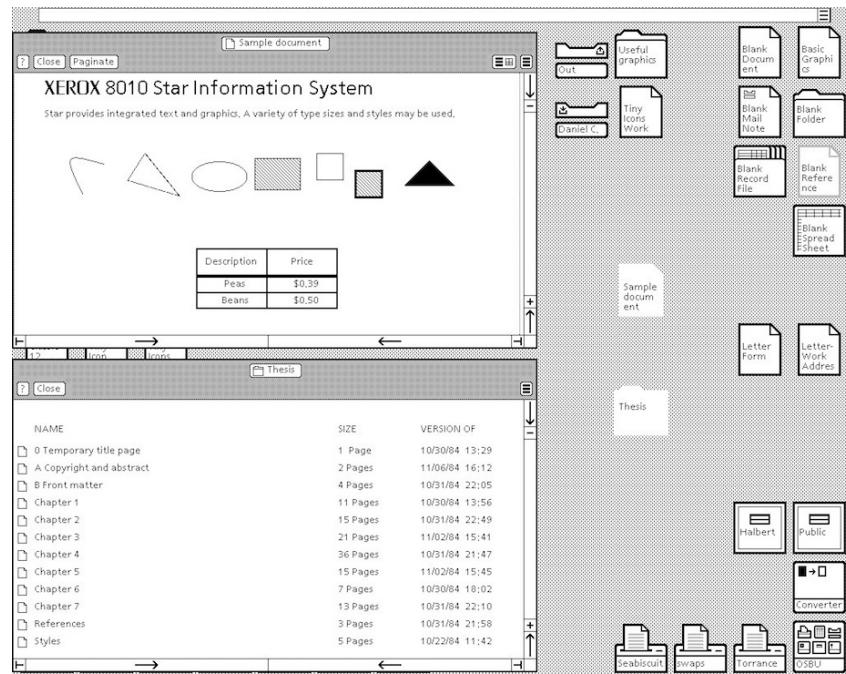


Figure: Xerox Star Desktop. Xerox Star Desktop; for a system designed 40 years ago, how different is this from our modern desktops? —Image Credit: <http://toastytech.com>.

As the prototyping proceeded the team evolved some principles: familiar user's conceptual model; seeing and pointing versus remembering and typing; what you see is what you get; universal commands; consistency; simplicity; modeless interaction; and

user tailor-ability. We have already seen that the Nest Thermostat exhibits ‘familiar user’s conceptual model’ in the use of the turning and pressing knob, both ‘consistency’ and ‘simplicity’ of operation in the movement of complexity from the user-domain, and ‘tailor-ability’ in the learning algorithms of the underlying programme logic. We’ll further expand upon the core Star principles next with the exception of ‘seeing and pointing...’, ‘what you see is...’, and ‘modeless interaction’ as these three can be taken for granted by the modern UX'er – their integration into the operating system being complete and ‘invisible’:

1. ‘Familiar Conceptual Models’ – Star equates familiar conceptual models specifically with the idea of the desktop and the items that you might find on it. It exemplifies e-mail and suggests that because e-mail is similar to postal mail than icons that describe e-mail in the form of in-baskets and out-baskets – letter icons being added to each to represent the incoming and outgoing mail. Also, Star tailors these to the users familiar conceptual models by sending both text and files. This was a very new departure, previous files were sent using FTP – or some other transfer protocol – but this separation is not the case in the real world. So in reality, Star takes familiar real world concepts and maps them to the virtual world.
2. ‘Universal Commands’ – Star also proposes the idea of universal commands. Previously each application defined its set of commands, and shortcuts to those commands, and expected the user to remember the specific operations without reference to those already present on the system. Star does away with all this by defining a set of universal commands such as ‘move’, ‘copy’, ‘delete’ and the like; having the same action and being controlled by the same shortcuts, as every other program within the operating system. This may seem obvious and normal to us now, but it wasn’t in 1980. Now, universal commands are present in the standard operating system but there is sometimes a tendency to change these commands in the application, or define a new set of commands when an application is going to be written once had distributed to different platforms (different platforms having different universal commands). In this case, you should think very hard about the rationale of breaking universality; if you are writing for many platforms, think about defining a set of commands to be used at runtime based on the kind of platform the application is being run upon.
3. ‘Consistency’ – Consistency, or pragmatic consistency as the Star Team suggest is best, is another important principle for usability. By making sure that consistent actions occur across the programme functionality we can assist the user in learning the correct outcomes for the same action over different applications, reinforce the users understanding of what these outcomes will be, match a user’s expectations with real-world actions, and dovetail into the familiarity of actions in the user’s conceptual model. Consistency should also be considered specifically concerning the dominance of real-world physical metaphors when compared to virtual ones, and the pragmatics of maintaining consistency even when that might go against the users understanding of what should happen. In should, you need to think very carefully about how to maintain consistency within your applications.
4. ‘Simplicity’ – Simplicity is incredibly important, but it’s not a clear-cut principle; we can make a simple interface, but it may not fill the desired outcomes required by the user. What is simple to one user might be verbose and tedious to another user, especially when the first user might be a novice while the second, an expert. We can overcome some of these problems of simplicity by enabling expert access

to system complexity. Similar to the Nest Thermostat, we can choose to use the technique of progressive disclosure. Or we can separate out novice and expert interfaces such that shortcuts and accelerated behaviours are accessible by experts while slower more progressively disclosed interactions that support learnability and familiarity are available for the novice user.

5. ‘User Tailor-ability’ – Usability is all about customisation, personalisation, and system adaptation; this means flexibility. We have briefly discussed the principle of flexibility already. The Star system is reasonably straightforward in its customizability, and individualisation (as it is known in the ISO standard), but this level of adaptability was unheard of at the time and pointed the way to a deeper acknowledgement that all individuals are different.

It is my opinion that these five principles are so important and timeless that their formulation and practical application as part of the Xerox Star user interface was without a doubt revolutionary. Without this interface, there would be no Apple Mac, or Microsoft Windows – well at least not as we know them. However not everything is down to the Star team, some of the knowledge (particularly regarding human cognition and behaviour) even pre-dates its development, and now exist as user models to be applied before development; in an attempt to uncover the usability of an as yet unknown system.

11.2 Universal Design and Design for All!

“The Star system is reasonably straightforward in its customizability, and individualisation (as it is known in the ISO standard), but this level of adaptability was unheard of at the time and pointed the way to a deeper acknowledgement that all individuals are different.”

This is what Star implies in its discussion of ‘User Tailor-ability’... ‘*individuals are different*’. The concept of universal design, or design for all, was created to address this very idea of individual differences; and, in reality, means universal usability, the design aspect being applied to signify that this universality must be thought of from the design stage through to inception. Universal design can mean many things to many people. Some discuss it in terms of the society at large, by making reference to socio-economics, ethics, and issues of general discrimination. Others see design-for-all as a technological issue and a problem to be solved. Still others link design-for-all to a way of thought that should encompass everyone. In the context of computing and software development, many suggest that technology must focus on designing products so that they are usable by the widest range of people. In reality, every person is a unique individual, and so this view may not be sustainable or achievable because, to create universal usability by designing for all, involves making generalisations about users, and it is these exact generalisations that were the impetus for Universality in the first place.

However, ‘Universality’ suggests to most UXers and engineers that the solutions they come up with must best fit most of the population most of the time. Many organisations follow the viewpoint that universal usability means design-for-all; the argument often following thus:

"A focus on designing products so that they are usable by the widest range of people operating in the widest range of situations as is commercially practical". Paradoxically, they also come up with a pointer to a workable solution: "As one might imagine, there are no universally usable products. There simply is too great a range of human abilities and too great a range of situations or limitations that an individual may find themselves in".

But unfortunately, do not take it: "Thus, universal usability is more a function of keeping all of the people and all of the situations in mind and trying to create a product which is as flexible as commercially practical, so that it can accommodate the different users and situations." [Vanderheiden, 2000]

While Universal Usability seems reasonable on the first inspection, it may not be sup-portable in practice. By trying to address all user needs in one design, the technologist is apt to address none. Making software usable is not just about a utilitarian view of software use, it is also about the personal choice of the user ([we have already touched on this](#)). Too often designs are implemented based on knowledge and aesthetics of the designers and engineers, but not on those held by the user. Consider the example of a spreadsheet application, in which a visually impaired user may have difficulty in choosing cells and accessing spreadsheet functions because the user interface (in this case a GUI display) does not support their needs. Another example concerns how to notify a deaf person working in a communal office that the application they are using has sound (which may be set too high).

When mainstream computer systems were initially developed, the overriding aim was focused on the creation of a computing resource, and not on the interaction of humans with that resource. Engineers and Systems Architects thought that they had overcome so many problems in the creation of computing resources that users would be thankful just for the computational ability; *and they were*. However, as systems became more complex and machines spread to the desktop to be operated by non-technical users, interaction with these machines became more of a concern. Some solutions, collectively known as User Interface Management Systems, were suggested. These mainly focused on interface generation for large bespoke embedded systems (aircraft avionics and the like) which met with little success when moved to off-the-peg software development.

I think that system flexibility is the only way to overcome the constraints placed on users by the need to service the perceived interaction requirements of all users. Design-for-all is only needed if that design is trying to fulfil all the gaps in technology provision created by an inappropriate user interface. We need a way to make the user interface bespoke to the individual user, and I think that universal access to software applications does not truly exist because the user interface and the system application logic are con-joined. Further, a stable and usable interface specification between these parts does not exist, and this means that separation cannot occur. Interaction design [Heuristics](#) that support a separation between the user interface and the code that implements the functionality of the application are ably demonstrated in the Mozilla User Interface (XUL) implementation. XUL allows a different 'Look and Feel' to be used for different applications and operating systems (also see '[Skins](#)'). By this separation, universal access to applications can be accommodated because the interface can adapt to the task without the need to change any part of the other functionality. We can see that this kind of design thinking supports the utilitarian user requirement, user activity,

and users personal choice. It provides a halfway-house ‘coarse’ grained interface and a method of fine tuning to a specific user. Even though the rendering is still visual, Mozilla points the way to component separation, and, therefore, interface adaptability.

In his article ‘Bridging the Digital Divide with Universal Usability’ [[Shneiderman, 2001](#)], Ben Shneiderman asks the question ‘Can you design a text-only interface that conveys the contents and experience of an animated Flash presentation?’ It is an interesting problem, but one that cannot be solved by designing one all-encompassing solution. Indeed, there can only be a solution if the information in question provides the opportunity for universal access. Once the opportunity is provided, interfaces can be developed to access that information in the most appropriate way to the user and not to the information itself. For instance, if an audio file is created without the opportunity of both a graphical or textual expression, then a driver interacting by audio only could not access that information source.

My point is, you should think of Universal Design, Design for All, or Universal Usability less concerning creating a physical design to be captured in programme logic, but more about providing the opportunity for universality. The way to provide this opportunity is to support ‘openness’ and ‘consistency’ of the data and API, and tailor-ability of the interface. Now it would be wrong to suggest there is no reason to provide an interface beyond an open API; we have to interface with the system at some point, but adding this separation of interface and programme logic will help you focus on supporting the user’s experience. Further, understanding the user, and the usability of your system will also help you build a flexible, but the coarse-grained interface, while understanding which aspects most require the ability to be personalised.

11.3 Usability Models

Models of the user have existed for many years. Some of the first were developed by Miller in an attempt to apply information theory to the human. Information theory is a branch of applied mathematics and electrical engineering involving the quantification of information. Historically, information theory was developed, by Shannon, to find fundamental limits on signal processing operations such as compressing data and on reliably storing and communicating data. Miller extended this model into the psychology domain by defining a model of the bandwidth that people could cope with, calling it a channel. This led to the historic, and still quoted, 1955 work ‘The Magical Number Seven, Plus or Minus Two - Some Limits on Our Capacity for Processing Information’ – [as we have already seen](#)). Extending this idea of modelling the user, Card et al.’s famous 1983 work ‘The Psychology of Human-Computer Interaction’ [[Card et al., 1983](#)] proposed the Human Processor Model which is also still referred to today. User models are in continued development with many active researchers trying to add knowledge and principles to the field.

You’re likely to see these models variously described as User Models, Cognitive Models, or User Simulators; I’ve listed them here as Usability Models, because in reality that is just what they are. They see the users as a very one-dimensional entity – analogizing them to a processor or processing entities. These models say nothing of the non-task related traits such as effective use, emotion, or pleasure – but see the best outcome only as measurable and efficient. Keep this in mind when you are applying them because

they are not means complete, or rich enough to do a human, full justice. However, they can be very useful for predicting usability.

In reality you're unlikely to come across these kinds of user models in your UX work, they are used more widely in research and at the very edges of development. This said you may come across them in some unlikely cases, and it may also be useful for you to have a brief understanding of the key points of their application, just in case...

11.3.1 The Human Processor Model

The Human Processor Model is a cognitive modelling method used to calculate how long it takes to perform a certain task. This method uses experimental times to calculate cognitive and motor processing time. The value of the human processor model is that it allows a system designer to predict the performance concerning the time it takes a person to complete a task without performing experiments directly. In reality, this means that empirical work already undertaken is captured as a set of principles that are so general that they can be applied to any usability problem within their scope and domain. This removes the onus on software engineers and UX specialists to run their experiments, instead asserting conformance by applying the user model to their development. The model uses the cognitive, perceptual, and motor processors along with the visual image, working memory, and long-term memory stores. Each processor has a cycle time, and each memory has a decay time. Therefore by following input pathways and combining them with the associated cycle or decay times, the time it takes a user to perform a certain task can be calculated. Card et al define the method for determining processes as the following steps: (1) write out main steps based on: a working prototype, simulation, step by step walk-through of all steps; (2) clearly identify the specific task and method to accomplish that task; (3) for each final step identify sub-levels down to a basic process; (4) convert into pseudo code; (5) list all assumptions; (6) determine time of each operation; (7) determine if operation times should be adjusted; (8) sum up execution times; and finally (9) iterate as needed and check with prototyping.

11.3.2 GOMS

Goals, Operators, Methods, and Selection rules (GOMS) was also proposed in the same book as the Human Processor Model. It reduces a user's interaction with a computer to their primary actions. Using these primary actions as a framework, an interface can be studied. There are several different GOMS variations that allow for different aspects of an interface to be studied and predicted. Goals are what the user intends to accomplish. Operators are actions that are performed to get to the goal. And methods are sequences of operators that accomplish a goal. There can be more than one method available to accomplish a single goal, if this is the case, then selection rules are used to describe when a user would select a certain method over the others. The GOMS method is not necessarily the most accurate of all the usability measurement methods, but it does have certain advantages. An estimate of a particular interaction can be calculated with minimal effort in a short amount of time. With a careful investigation into all of the detailed steps necessary for a user to successfully interact with an interface, the time measurement of how long it will take a user to interact with that interface is a simple calculation. The main drawback of GOMS is that it expects users to follow logical

routines and is not resilient to user unpredictability, and all of the techniques work under the assumption that a user will know what to do at any given point. This aspect of user interaction are commonplace and are often what makes UX so challenging and interesting.

11.3.3 KLM or KLM-GOMS

Keystroke Level Modelling, sometimes referred to as KLM or KLM-GOMS, is an eleven step method that can be used by software engineers seeking ways to estimate the time it takes to complete simple data input tasks using a computer and mouse. By using KLM, individuals often find more efficient or better ways to complete a task simply by analysing the steps required in the process and rearranging or eliminating unneeded steps thus, (1) obtain a working prototype of computer interface or a step by step operational description of a task; (2) identify the goals or the desired outcome of work; (3) for each of these goals, find subgoals or tasks that achieve the main goals; (4) identify methods to main goals and all subgoals; (5) convert description of methods to pseudo-code; (6) state any and all assumptions used in the making of pseudo-code and goals; (7) determine appropriate mental or keystroke operators for each step; (8) assign time values to mental or keystroke operators; (9) add up execution times for operators; (10) adjust total time of task to be sensitive by age of expected, this step was initially implied but is explicit as a later extension; and finally, (11) verify the validity of the results. The method is designed to be much easier than GOMS and especially useful in the evaluation of time specific tasks that require, on average, less than 5 minutes to complete.

11.4 Collated Usability Principles, Guidelines, and Rules

The first thing to notice about usability principles is that there are many of them, and many different ones are proposed by many different usability luminaries⁶. This is different in some regard to those [previously proposed](#) where there is broad agreement among experts and a smaller set of principles at work. However, usability has been a major topic in the understanding of user experience for a much longer period than has accessibility. In this case, it is appropriate to collate the different usability principles along with their proponents to identify the overlap that exist between them.

You will notice in '[Table: Collated Usability Principles](#)' that the left column describes the principle, guideline, or rule (these are sometimes used interchangeably between the different experts)⁷; while on the right side the experts are listed along with a footnote pointing to the text from which the principle is derived. In collating these principles, I have not followed slavishly the nomenclature proposed by each author but have instead placed them in categories that I believe have the same conceptual value, even if the naming of that principle does not follow from its source.

Table: Collated Usability Principles. Usability Principles Collated by Source.

⁶Remember, usability principles sell books!

⁷We'll call them Principles from now on.

Principle	Appears in Source
Closure (Dialog Yields)	Shneiderman ⁸
Consistency / Standards	Xerox-Star ⁹ ; Shneiderman; Norman ¹⁰ ; Nielsen ¹¹ ; ISO 9241-110 ¹² ; Dix, Finally, Abowd & Beale ¹³ ; Raskin ¹⁴ .
Constraints (Exploit)	Norman.
Control & Freedom (Support)	Shneiderman; ISO 9241-110; Nielsen.
Error Handling (Simple)	Shneiderman; Norman; ISO 9241-110; Nielsen.
Familiarity & Metaphor	Xerox-Star; Norman; Dix, Finally, Abowd & Beale; Raskin.
Feedback (Informative)	Shneiderman.
Help & Documentation	Nielsen.
Interrupts (Resumption)	Raskin.
Describing (Self)	ISO 9241-110.
Heuristic Evaluation	Nielsen.
Learnability	Dix, Finally, Abowd & Beale, ISO 9241-110; Sharp, Rogers and Preece ¹⁵ .
Mappings (Real-Virtual)	Norman; Nielsen; Dix, Finally, Abowd & Beale.
Memory Load (Reduce)	Shneiderman; Sharp, Rogers and Preece.
Navigation & Freedom (Support)	Raskin.
Reversal of Actions (Easy)	Shneiderman; Nielsen; Dix, Finally, Abowd & Beale.
Safety	Sharp, Rogers and Preece.
Shortcuts (Provide)	Shneiderman.
Simplicity	Xerox-Star; Norman; Brooks ¹⁶ .
Singularity of Focus (Attention)	Raskin.
Task Suitability & Conformance	Dix, Finally, Abowd & Beale; ISO 9241-110.
Tailor-ability / Flexibility	Xerox-Star; Nielsen; ISO 9241-110; Dix, Finally, Abowd & Beale.
Universal Commands	Xerox-Star; Dix, Finally, Abowd & Beale; Raskin.
Utility	Sharp, Rogers and Preece.
Visibility (Make Things)	Norman; Nielsen.

From a cursory view of '[Table: Collated Usability Principles](#)' we can see that there are many principles that deserve greater investigation. These include learnability, is it

⁸B. Shneiderman and C. Plaisant. Designing the user interface: strategies for effective human-computer interaction. Addison-Wesley, Boston, 5th ed edition, 2010.

⁹D. C. Smith, C. Irby, R. Kimball, B. Verplank, and E. Harslem. Designing the star user interface. BYTE, 7(4):242–282, 1982.

¹⁰D. A. Norman. The design of everyday things. Basic Books, New York, 1st basic paperback edition, 1988.

¹¹J. Nielsen. Usability engineering. Academic Press, Boston, 1993. Nielsen also lists Aesthetic and minimalist design.

¹²ISO/TR 9241-110:2006. Ergonomics of human-system interaction – part 110: Dialogue principles. TC/SC: TC 159/SC 4 ICS 13.180, International Organisation for Standardisation (ISO), Geneva, Switzerland, 2006.

¹³A. Dix, J. Finlay, G. Abowd, and R. Beale. Human-computer interaction. Prentice Hall Europe, London, 2nd ed edition, 1998. Dix, Finally, Abowd & Beale describe '*Learnability*' as: Predictability, Synthesizability, Familiarity, Generalizability, and Consistency. '*Flexibility*' as: Dialog initiative, Multi-threading, Task migratability, Substitutivity, and Customizability. '*Robustness*' as: Observability, Recoverability, Responsiveness, and Task conformance.

¹⁴J. Raskin. The humane interface: new directions for designing interactive systems. Addison Wesley, Reading, Mass., 2000.

¹⁵H. Sharp, Y. Rogers, and J. Preece. Interaction design: beyond human-computer interaction. Wiley, Chichester, 2nd ed edition, 2007. Preece, Rogers and Sharp also list Satisfying, Enjoyable, Fun, Entertaining, Helpful, Motivating, Aesthetic, Supports creativity, Rewarding, and Emotionally fulfilling.

¹⁶F. P. Brooks. The mythical man-month: essays on software engineering. Addison-Wesley Pub. Co., Reading, Mass., anniversary ed edition, 1995.

easy to learn, or work out, how the system operates. Efficiency, how fast can tasks be accomplished. Memorability (memory load), is it easy to remember how the system operates. Errors, how many errors are made and can they easily be recovered. And satisfaction, do users like the overall feel the system. Also, I would add the following: information flow; is feedback fast, appropriate and not too detailed, but detailed enough. Granularity is the interface sectioned enough so that progressive disclosure of the interface can occur. And egocentricity, is their user specific feedback and guidance (but we will get to these later). For now, however, consider '[Table: Collated Usability Principles](#)' in more detail, and I will try to explain my rationale for the grouping and discarding of the principles within it, to form a smaller aggregate set.

So I'm going to discard some of these principles because I think they are very niche because I don't agree with them, because I don't think they are helpful, or because there seems to be little consensus in the community.*

1. First, I'm going to discard Shneiderman's 'Closure (Dialog Yields)', I think this is pretty obvious and is now a well-understood software engineering practice; you wouldn't normally have a dialog that opens another, further in some cases such as a wizard this principle isn't helpful.
2. Next I'll discard Nielsen's 'Heuristic Evaluation', not because there is anything wrong with it, but because [I'm going to cover this in more detail later](#).
3. Next, Preece, Rogers and Sharp's 'Safety' is for the chop, while safety at the interface level may be useful, I think it is a little too niche to be included in a general set of UX principles.
4. Now I think 'Task Suitability & Conformance' – proposed by Dix, Finally, Abowd & Beale, and ISO 9241-110 – can [be referred to earlier](#), it has already been addressed as part of the UCD aspect of the design.
5. Finally, I'm discarding Preece, Rogers and Sharp's 'Utility'. Utility is obviously important - but if you've gone to the trouble of starting the development I'd imagine you expect the software to have some utility, and if the utility is about the interface under construction, well that can only be [assessed after evaluation](#).

Now the 'carnage' is over, let's look at the principles that will stay and/or be amalgamated:

1. 'Consistency / Standards' + 'Universal Commands'. This seems pretty obvious to me - in that universality is directly related to standards to provide *consistency*.
2. 'Constraints (Exploit)' + 'Memory Load (Reduce)' + 'Simplicity'. Again I think this is straightforward, in that the constraint reduces complexity, which reduces the load on short-term memory and assists in interface *simplicity*.
3. 'Control & Freedom (Support)' + 'Shortcuts (Provide)' + 'Tailor-ability / Flexibility'. Supporting user control, and user freedom, all contribute towards *Flexibility* (Tailor-ability), providing shortcuts is one additional way of providing the control we envisage. Now I'm not expanding on this one further in this chapter, but instead, I'm going to [refer back](#).
4. 'Error Handling (Simple)' + 'Feedback (Informative)' + 'Singularity of Focus (Attention)' + 'Visibility (Make Things)' + 'Navigation & Freedom (Support)'. Now we don't talk about it here, but these four principles can all be subsumed to assist in '*Situational Awareness*'. Situational awareness involves a perception of

the environment critical to making decisions in complex and dynamic situations. Indeed, situation awareness “*involves being aware of what is happening in the vicinity to understand how information, events, and one's own actions will impact goals and objectives, both immediately and in the near future*”. I contend that simple error's, informative feedback, visible components and interactions, coupled with a singularity of focus, all go towards creating an environment that enhances situational awareness.

5. ‘Familiarity & Metaphor’ + ‘Mappings (Real-Virtual)’. By ensuring there is a cognitive mapping from real to virtual we can assist in the side goal of interactive *familiarity.
6. ‘Help & Documentation’ + ‘Describing (Self)’. It is my opinion that *self-description* is the principle to which good help and good documentation contribute.
7. ‘Interrupts (Resumption)’ + ‘Reversal of Actions (Easy)’. Reversal and resumption of actions and interactions seem to be related to me. This also feeds into the user control, but these are a little more specific and seem to me to me mainly about *'interaction stability'*.
8. ‘Learnability’; we'll get to this later on.

Before moving on to a discussion of our principles, consider for a moment the usability requirements of the novice and expert user. The usability principles as described in this section not straightforwardly applicable because there is some conflict between the requirements of different users; at a coarse-grained level, these can be classed as the novice and the expert. The thing to remember with all these principles is that you need to make provision for slow, simplistic and constrained novice use; provision for fast, complex and unconstrained expert use; and some learnability path that enables novices to become experts in the shortest amount of time possible. This last requirement is most difficult because understanding if learnability is effective and efficient, can only be done with longitudinal studies; in the real world, it is very unlikely you will have the budget for this. This said it is up to you, the UX usability specialist, to keep these concepts of the novice, expert, and the transition between the two, in your mind at all times.

As is the case for all user experience it can be difficult to validate the effectiveness and efficiency of the usability principles you design into your system. It is often easier to see that a usability principle is not present than to see if a usability principle is present.

11.5 Potted Principles of Efficient User Experience

Interaction design focuses on improving technologies that provide interaction with the user **[9241-110:2006, 2006]**. This is normally led by the UX interaction developers, who are responsible for engineering software that shapes the user interface. Technologies mainly include Windowing Toolkits, Graphical User Interfaces, and System operations, etc. Use of these technologies affects how people interact with the system both as creators and consumers of information. Therefore in any effort to support the user, it is crucial that the features and limitations of these technologies are clearly understood. For the UX'er, current core principles that should run through all aspects of the development can be summarised as follows.

11.5.1 Facilitate Consistency

Consistency is a key factor in usability; and two major ways of attaining consistency is to follow standards, and to develop systems that apply the same command, control, and interface structures universally across the development. While standards and universality are two key elements of consistency, the concept of consistency can exist upon its own, and you should also think at all stages of development 'Am I developing a consistent interface, are the interactions consistent across the platform and development?'

The use of standardised components is a major aspect when designing for the user. In this way, the developer can be assured that look and feel of the software will be harmonious with the best practice of the operating system and dovetail into the guidelines of the underlying environment. This is often difficult if the design is to be accomplished on different platforms because the look and feel changes (for instance there is a significant difference between the operating modality and look and feel of Microsoft Windows and Apple Mac OS X). However, there are some ways to address the problems of standardisation across the interface. The developer can design for the interpreter platform, or by choosing a cross-platform system such as Java or Qt, which adapts to the base operating system environment. In this way, the development cost is removed from the developer to the language or framework engine. Besides, the developer may wish to use techniques such as Model-View-Controller (MVC) to separate the presentation from the programme logic and use a window toolkit to enable a more **harmonious distribution across different platforms**. In either case standardisation to the underlying environment is critical to maintaining the usability of the development, and to reduce the additional cognitive overload on the user in switching between operating modalities.

Besides, there are many pan-system standards which should also be followed when building user-centric software. These guidelines and standards have been created so as to encapsulate knowledge and best practice into a format that can be easily applied by engineers who do not have an in-depth knowledge of human factors or ergonomics. The standards have been designed by both operating system manufacturers and international standards bodies, and by following these standards the developer can be assured that their system will be usable by the most number of people.

Questions to think about as you design your prototype:

1. Am I developing a consistent interface?
2. Are the interactions consistent across the platform and development?
3. Is my command and event structure universal across the development and platform?
4. Am I following standards and best practice?
5. Am I following the platform design guide¹⁷?

11.5.2 Facilitate Familiarity

Familiarity (including via Metaphor) is a key principle for enhancing usability. As we have already seen from the Xerox Star team, and from many usability experts,

¹⁷Such as those for the Mac OSX <http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/Introduction/introduction.html>

dovetailing into a user's learned behaviour is the primary way of enhancing familiarity with a system, which they are as yet unfamiliar with. You can do this in many ways, from mapping real world to virtual world concepts, using common terms and jargon, by making the look and feel of the new system similar to that of an old system – or an old manual way of doing things (such as forms that look similar). However, you decide to implement familiarity or enhance that familiarity within your system, always make sure you think if your concept of familiarity is likely to be the same as your users. Remember the development is not for you, but for your target users.

Questions to think about as you design your prototype:

1. Does your system map real world concepts to the virtual world?
2. Does your system use terms the user is familiar with (including Jargon)?
3. Does the system work in familiar ways, regarding itself and other comparable applications?
4. Do you assuage 'intuition' for familiarity?
5. Does your system use easily understandable (and, therefore, familiar) messages?

11.5.3 Interaction Stability

You won't find the principle of interaction stability in many other texts, but it seems it is a reasonable summary of some of the aggregated principles and guidelines that other usability experts purport. When we are thinking of interaction stability we are trying to make sure that interactions are stable, that they can be resumed if they break, that data will not be lost, that the place someone is at, in an interaction (the stage for the step), can be returned to in the event of a failure. For example, a web form that times-out after one hour and does not save the existing contents of that form does not exhibit interaction stability; however a form that times-out, saves the data, and allows the user to commence from this saved point, does exhibit interaction stability.

One further way of enhancing interaction stability is 'preview'. A lack of preview of upcoming information is one of the major issues to be addressed when building interfaces and considering interaction stability. Consequently, the preview is considered to be a primary component of good interaction stability. This preview can be manually achieved by try operations and awaiting desired or undesired outcomes from the environment. In an interface context, the lack of previews of both the outcomes of specific actions and information relating to navigation of the interface itself suggests that some degree of 'foreknowledge' should be implemented so that a limited preview can be obtained. In this case, a good interface design will remove the onus on the user by providing outcomes and descriptions of what will occur if a specific action is performed, as well as suggesting paths to certain information of functional outcomes.

Questions to think about as you design your prototype:

1. Are you able to resume interrupted actions?
2. Are you able easily reverse an action, incorrectly taken?
3. Are you able to understand your location in the interaction?
4. Does your system recover well from an unexpected event?
5. Do your interactions (including dialogs) exhibit stable non-cyclical behaviour and closure?

11.5.4 Facilitate Learnability

Learnability is a key aspect for interface design. Indeed, this is why large computer systems and operating environments often come with a series of rules and best practices for software development that describe the standard ways of interacting on that specific system. Application menus that normally have 'file', and 'edit', to the left and 'help' to the far right all the way through to common dialogues for picking colours or interacting with files systems, all try to enhance learnability. I have not yet met a user who has read the user manual for any large system. In reality, users prefer to investigate and orientate themselves to the system. Therefore, interface development should assist the user in self-directed learning of the interface components as opposed to relying on the reading of the user manual or help text. It is only by helping make your system easily learnable that you will be able to reduce the amount of support required by the user and, therefore, reduce long-term costs while enhancing user satisfaction.

Questions to think about as you design your prototype:

1. Is your system behaviour predictable?
2. Can users easily transit from novice to expert?
3. Can you understand the system behaviour without recourse to manuals or help systems?
4. How easy is it to learn any bespoke system functionality?
5. Does your system facilitate self-learning and functionality investigation?

11.5.5 Facilitate Robustness

As with all aspects of a computer system, speed of service balanced with the robustness of the system is critical for the user. In most cases, users prefer systems to be more robust because the effect of a faster service speed balanced against a system that crashes frequently. This scenario costs more time in the long run and introduces a lack of trust in the user's perception of the interface and therefore the quality of the end product. Obviously, robustness is a systemwide concern, however, because the interface elements are often developed as one of the last things in the development lifecycle, robustness is sometimes sacrificed for a speedy development that ends on time.

Questions to think about as you design your prototype:

1. Does your system recover well from an unexpected event?
2. Are errors easily dealt with?
3. Are incorrect user actions easily recoverable?
4. Is the user-state saved between sessions in the event of a system failure?
5. How does your system handle abnormal input?

11.5.6 Facilitate Progressive Disclosure

Progressive disclosure is a tricky concept to discuss. It was previously thought that usability was enhanced when the number of selections required of a user was reduced to the bare minimum. This created interfaces that were very complex having as

much functionality as possible on the screen at the same time. In reality, quantitative measures showed that this decreased performance because it took the user longer to make a decision about what to select, as opposed to the actual selections themselves. Progressive disclosure is an antidote to this kind of information and operation overload and suggests that a hierarchy of operations moving through simple selectable steps is faster and easier for the user to understand than counting the selections required to achieve a specific action. Of course, progressive disclosure can lead to problems because there may be multiple levels of disclosure, in which case the functionality can be hidden in the depths of the computer system such that it is never used. The interface developer, therefore, needs to understand these complexities and try to mitigate them by providing progressive disclosure but limiting the hierarchical nesting of the activities that are being disclosed to a minimum.

Questions to think about as you design your prototype:

1. Does your interface look overly complex? If so, simplify.
2. Are there a lot of components displayed at one time? If so, clean it.
3. Is there a multitude of possible actions available to the user? If so, focus on building one action for one interface element.
4. Is there a tight logical hierarchy of actions?
5. Is the user-led along the interactive path?

11.5.7 Facilitate Scalability

Again, scalability is not a principle you will find in most usability texts. However, it is important in the real world. If your system is not able to scale concerning the way it displays data, or the way interactions progress, in a system that is being progressively overloaded, then the efficiency of that system, its usability, is drastically reduced. For example, I once worked on a system that displayed data in interactive bubbles, this worked well with small amounts of data – and also looked supercool. However, as the amount of data was increased the interface was not scalable and so it became unusable, the number of interactive bubbles expanding exponentially – making the visual search very difficult. The solution was to define the point whereby the display became unusable and then adapt the data back into a more traditional tabular format that was far easier to search and could efficiently display much more data than the interactive bubble format – although it didn't look as cool.

Questions to think about as you design your prototype:

1. Does your interface scale to handle larger datasets than envisaged etc.?
2. Does your system handle data and interaction within an acceptable time?
3. Do complex actions scale up regarding data and user requirements?
4. Do your interfaces remain simple when information is being dynamically added?
5. Can new functionality be added to your system without negatively impacting on its current interactions and interfaces?

11.5.8 Facilitate Self-Description

The principle of self-description is one which is pointed to by some usability experts, and is implicit in other usability principles; such as those which espouse good help and good documentation. The difference with self-description is that in the best possible circumstance the user should not need to refer to help or the documentation. This is because it is very unlikely – indeed normally only in extremes – that a user will ever consult the manual. One thing you should consider is that self-description is not necessarily only about explicit textual descriptions. Implicit descriptions based on the visual rendering – the way something looks – dovetailing into familiarity and simplicity, are also important in the self-description principle. By visually suggesting a path the user should take you enhance the self-description of that particular interface and interaction.

Questions to think about as you design your prototype:

1. Is your system well documented?
2. Is help present and informative?
3. Is it possible to understand the program functionality without recourse to the manual?
4. Is it possible to understand the interface, widgets, and interactivity without recourse to the manual?
5. Is it possible to fully understand all dialogs, messages, and status'?

11.5.9 Facilitate Simplicity

Simplicity is a commonly espoused usability principle; indeed, many usability experts list this as a key principle to be followed. However, from work that my team and I have conducted, building in simplicity treads a very fine line, because it impacts on aesthetics, pleasure, and emotion. An interface that is too simplistic will be seen as boring, or only utilitarian, one that is too complex will be perceived as being overly complicated and aesthetically displeasing. The simplicity of the interaction, on the other hand, is a useful trait to develop, and this can be done by hiding complex system functionality from the user (often the novice user), thereby presenting them with a constrained set of choices while still enabling the complexity to be accessed by expert users.

Questions to think about as you design your prototype:

1. Is your system presented simply?
2. Are the interactive elements simple to understand and use?
3. Can you understand the system behaviour without recourse to manuals or help systems?
4. Does your system exploit natural interactive constraints?
5. Is complexity hidden from the novice user?

11.5.10 Facilitate Situational Awareness

Again, situational awareness is not a principle you will commonly come across in many usability texts. However, I think it is a good way of describing certain aspects of a usable design. Let us recap: Situational awareness involves a perception of the environment critical to making decisions in complex and dynamic situations. Indeed, situational awareness “*involves being aware of what is happening in the vicinity, understanding how information, events, and one's own actions will impact goals and objectives, both immediately and in the near future*”. Can increasing our awareness of the situation and then building a model integrating aspects that are common within the usability literature help. Aspects such as the desire for simple errors, informative feedback, visible components and interactions, coupled with a singularity of focus, all go towards creating an environment that enhances situational awareness. But situational awareness is much more than this; let us take a brief look at a model of situational awareness (see [Figure: Endsley's Model of Situation Awareness](#)).

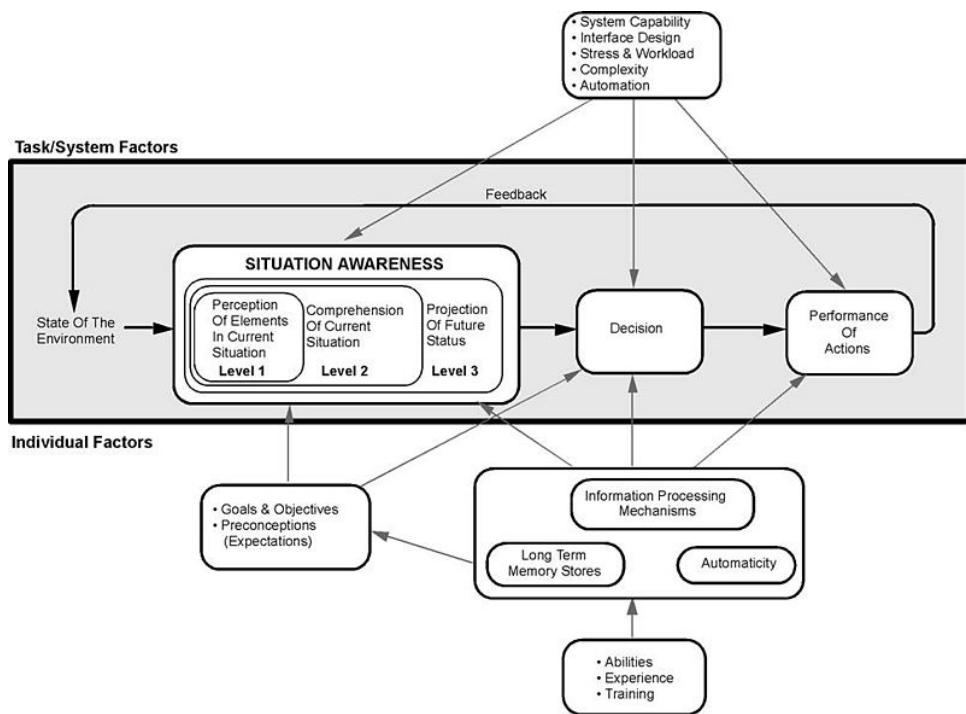


Figure: Endsley's Model of Situation Awareness. Endsley's Model of Situation Awareness –Image Credit: adapted from Endsley, 1995.

Endsley's Model espouses a good perception of the elements in the current situation, a comprehension of that situation, and an understanding of the future status. All of these factors enable you to make a decision as to the interaction; perform these interactions that then alter the state of the environment – in this case the interface or the interaction; which then affects the perception of elements... so on. We can see that this kind of view is directly related to good usability. Understanding the interface and its abilities are all part of the perception of interface elements, a comprehension of the current interactive situation including your location in the interaction. This is known as the user orientation or ‘where-ness’ (detecting cyclic behaviour, direction and distance) and is important in interface design as it enables users to navigate with

some degree of accuracy. The environment, however, must be updated such that cues are provided in an appropriate manner, giving explicit orientation information such that navigational information can be detected. The similarities between real-world movement and that surrounding the interface suggest that the provision of some form of explicit and appropriate orientation method are an advantage when navigating the interface. This would mean that a user can make a choice as to whether they want to be at the current location and if not aid them in deciding how to navigate best to their perceived destination. Finally, situational awareness rests upon our ability to predict our future status (this may be including familiarity as well as an understanding of how the system works). As an example, think of the last time you completed a credit card transaction or order from the likes of 'Amazon'. In this case, you are led through a sequence of events; each step is given in graphical overview at the top of the page, and as each step is accomplished so the colour is changed to highlight a stage has been satisfactorily completed (see [Figure: Amazon Check Out](#)). This is all about enhancing your situation awareness by providing you with a stepwise overview of the current situation and an understanding of what will happen in the future.



Figure: Amazon Check Out. Amazon Check Out – 'Projection of Future Status' –Image Credit: Amazon.

Questions to think about as you design your prototype:

1. Does your system facilitate orientation both within the interface and within the interaction?
2. Is orientation and navigation, around and through the interface (and interaction), easy?
3. Is error handling simple? Is feedback informative?
4. Are all components, needed for the interaction, visible?
5. Do you maintain a single focus of interactive attention, without distractors?

By understanding and following these simple design principles any software developer can begin to create user-centric applications that will be implicitly friendly to users. By applying these principles through requirements elicitation, analysis, user modelling, and interface development, user facing components can be enhanced, and best practice can be implicitly included within the development.

Remember... these are just the next ten of our total principles - for the next batch you can skip ahead). But just because you can, maybe you shouldn't... take your time to read the rest!

11.6 Summary

We can see that designing and building usable and efficient interfaces are by no means a trivial task. Even with the wealth of toolkits and accessibility API's currently available the software engineer and UX specialist should not underestimate the amount of work

and testing required for a comprehensive interactive development. Fortunately, there are a number principles to assist in this process, as well as the basic elements of the graphical user interface which can be assembled into a form that directly supports usability.

Some researchers see the ubiquity of the graphical user interface as being detrimental to the HCI field in general. They claim that focusing only on this kind of interaction means that there has been no novel work undertaken in the HCI domain since the heady days of Parc Xerox. I have some sympathy for this view but in reality, it is not completely correct. With the development of gesture-based interfaces and the upcoming use of haptics, the interactive elements of interface design are alive and well. I would agree, however, that the interactive displays used in these systems are stagnating to some degree. But in the context of development, building and designing interactive components that are tailored to the users need is difficult enough. By leveraging the principles already undertaken and the toolkits that provide well understood and consistent components and metaphors the software engineer can be reasonably assured that the interface design will be very familiar to the user and in this way assist usability in a practical manner.

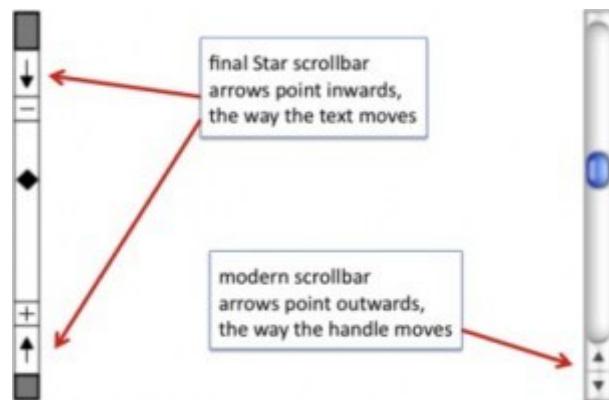


Figure: Scrollbar Miss-Design. Xerox Star and Modern Mac OS X Scrollbars –Image Credit: Byte.

Before we finish, a word of warning – as always from Dix – who suggests we may often ignore our fields previous work and start to reinvent the wheel:

“When one builds the justification of why something should work, the argument will not be watertight in the way that a mathematical argument can be. The data on which we build our justification has been obtained under particular circumstances that may be different from our own, we may be bringing things together in new ways and making uncertain extrapolations or deductions. Some parts of our argument may be strong, and we would be very surprised if actual use showed otherwise, but some parts of the argument may involve more uncertain data, a greater degree of extrapolation or even pure guesswork.”

“These weaker parts of the argument are the ideal candidates for focusing our efforts in an evaluation. Why waste effort on the things we know anyway; instead use those precious empirical resources (our own time and that of our participants) to examine the things we understand least well. This was precisely the approach taken by the designers of the Xerox Star. There were many design decisions, too many to test individually, let

alone in combinations. Only when aspects of the design were problematic, or unclear, did they perform targeted user studies. One example of this was the direction of scroll buttons: should pressing the ‘up’ button make the text go up (moving the page), or the text go down (moving the view)?

If there were only one interpretation it would not be a problem, but because there was not a clear justification this was one of the places where the Star team did empirical evaluation... it is a pity that the wrong answer was used in subsequent Apple Lisa design and carried forward to this day.” (see [Figure: Scrollbar Miss-Design](#)).

Indeed, in conversations with David Smith at CHI98 Dix described how in the first version of the design documents for the Star

“the scrollbar arrows pointed outwards as they do in modern interfaces. However, unsure of the correct orientation, the Star design team performed user studies with both orientations. Whereas the software designers were quite happy with the outwards form, the non-computing users were uniformly confused by this direction of arrows. Hence the inwards pointing arrows were adopted for the final Star design.

Unfortunately when the Star design documents were passed on to the later design teams for the Lisa and Macintosh, the initial, wrong version of the scrollbar designs was used! Hence, we came by our current scrollbar arrow direction by accident, and it is precisely the opposite of what was found to be easy to use”¹⁸.

11.6.1 Optional Further Reading / Resources

- [S. K. Card,] T. P. Moran, and A. Newell. The psychology of human-computer interaction. L. Erlbaum Associates, Hillsdale, N.J., 1983.
- [J. Johnson.] GUI bloopers 2.0: common user interface design don’ts and dos. Elsevier/Morgan Kaufmann Publishers, Amsterdam, updated and rev., edition, 2008.
- [S. Mestel] How bad ballot design can sway the result of an election. The Guardian. (2019, November 19). Retrieved August 10, 2022, from <https://www.theguardian.com/us-news/2019/nov/19/bad-ballot-design-2020-democracy-america>
- [J. Nielsen.] Usability engineering. Academic Press, Boston, 1993.
- [H. Sharp,] Y. Rogers, and J. Preece. Interaction design: beyond human-computer interaction. Wiley, Chichester, 2nd ed edition, 2007.
- [B. Shneiderman] and C. Plaisant. Designing the user interface: strategies for effective human-computer interaction. Addison-Wesley, Boston, 5th ed edition, 2010.

11.6.2 International Standards

- [ISO/TR 9241-100:2011.] Ergonomics of human-system interaction – part 100: Introduction to standards related to software ergonomics. TC/SC: TC 159/SC 4 ICS 13.180; 35.180, International Organisation for Standardisation (ISO), Geneva, Switzerland, 2011.

¹⁸<http://www.guidebookgallery.org/articles/handsacrossthescreen>.

- [ISO/TR 9241-110:2006.] Ergonomics of human-system interaction – part 110: Dialogue principles. TC/SC: TC 159/SC 4 ICS 13.180, International Organisation for Standardisation (ISO), Geneva, Switzerland, 2006.



Self Assessment Questions

Try these without reference to the text:

1. What is the significance of the Xerox Star interface?
2. What are the five main principles proposed by the Xerox Star team?
3. What does GOMS stand for and what does it involve?
4. What are the ten main principles of efficient design?
5. How do these principles differ from Shneiderman's rules?

12. Principles of Affective Experience (Emotion)

If you want a golden rule that will fit everybody, this is it: have nothing in your house that you do not know to be useful, or believe to be beautiful.

– Donald A. Norman

Affective¹ use, or emotional² use/design, both mean the same thing, and, in reality, were going to be using both terms interchangeably. However, you should be thinking of emotional use in more specific terms; a more specialised term of affective use. This is because the concept of emotional design is much broader than we need to address at the interface / interactive level. What we are trying to get to is ‘emotional engagement’ and to begin with, let me ask you a question...

What makes a person display a bumper (or fender) sticker on their car, which espouses their love for Microsoft’s new Metro User Interface (see [Figure: Metro Bumper Sticker](#))?



Figure: Metro Bumper Sticker. Metro Bumper (Fender) Sticker.
—Image Credit: Microsoft.

Indeed, we might ask a further question, ‘what is it that makes the Microsoft Metro User Interface an example of good design?’ (see [Figure: Microsoft’s Metro UI](#), and as applied to a Windows 7 Mobile Phone – [Figure: Metro on a Windows Phone](#)). It is this concept of goodness that is more difficult to explain especially concerning technologists who see goodness as effective and efficient; both readily measurable, and tangible.

Microsoft tells us that: “*The Metro design language was designed specifically to consolidate groups of common tasks to speed up usage. This is accomplished by excluding superfluous graphics and instead relying on the actual content to also function as the main UI. The resulting interfaces favour larger hubs over smaller buttons and often feature laterally scrolling full bleed canvases. Page titles are usually large and consequently, also take advantage of lateral scrolling. Animation plays a large part, with transitions, and user interactions such as presses or swipes recommended always to be acknowledged by some form of natural animation or motion. This is intended to*

¹Of or relating to the affections or emotions, esp. as contrasted with the intellect or rational faculty; emotional. Psychol. and Psychiatry. Of, relating to, or involving feelings, emotion, or mood. AFFECT - Philos. An emotional, unreflective response. Psychol. (and Psychiatry). A feeling or subjective experience accompanying a thought or action or occurring in response to a stimulus; an emotion, a mood. In later use also (usu. as a mass noun): the outward display of emotion or mood, as manifested by facial expression, posture, gestures, tone of voice, etc. — OED.

²Of or relating to the emotions; based on or appealing to the emotions. any strong mental or instinctive feeling, as pleasure, grief, hope, fear, etc., deriving esp. from one's circumstances, mood, or relationship with others — OED.

give the user the impression that the UI is ‘alive’ and responsive, with ‘an added sense of depth.’”

When I first saw the Metro interface I immediately loved it; it was clean, clear, stylish and with just enough elegance for me to perceive that using it would be a pleasant interactive experience. But this isn't the case of everyone:

“If the Windows environment comes to revolve around a kiddish interface with big colourful buttons that are limited to only presenting content to the user, I can imagine the whole Vista mess repeating itself all over with a huge lack of essential developer support and mocked in Apple advertisement’s” or “Man oh man, it is days like this when I miss the simplicity of DOS.” —Various Forums

And here lies the problem with designing for collective emotional experience, it is certainly not readily applicable to all users. Indeed, when we address easily quantifiable aspects such as usability or accessibility we get very consistent results, with affective computing, this is not necessarily the case. We see far more individualistic results, far more outliers, and far more variance within the data that we gather. In this case, you might be wondering why affective use is so important if it isn't quantifiable. There are three reasons, firstly, attractive things work better, their attractiveness produces positive emotions that cause mental processes to be more creative and more tolerant of minor difficulties or faults; secondly, emotional attachment is natural to us as humans, and I would venture, initially much more powerful than an efficient experience (the often quoted ‘function follows form’ is still applicable because the people who subscribe to this view, see function as an aesthetic quality); finally, just because something is not easily quantifiable doesn't mean to say that we cannot try to understand it, or that it is not of any use or significance to us in the UX domain.



Figure: Microsoft's Metro UI. Microsoft's Metro UI. Metro is an internal code name for a typography-based design language created by Microsoft, originally for use in Windows Phone 7. —Image Credit: Microsoft.

12.1 Visceral, Behavioural, and Reflective

In his book, ‘Emotional Design’ [Norman, 2004], Donald A. Norman proposes three levels (or dimensions) when designing with emotional response in mind, these are:

- Visceral: is equated with appearance, and ties into how we have evolved within our environment, which drives our perceptions of aesthetic and pleasure at an internal and unconscious level;
- Behavioural: relates to the pleasure and effectiveness of use, [we have seen this previously](#), however, Norman contends that there is pleasure in efficiency (we’re back to form following function again); and
- Reflective: is more complicated and coupled with self-image, personal satisfaction, and memories, it is about the message, culture, and the *meaning* of a product.

To us the most important is visceral and the reflective; we might think of this as determining if the interface looks ‘cool’ and if we have an overall positive impression. Indeed, reflective design is all about determining this overall impression of the product – in our case the interface; thinking back to the interface, reflecting upon its total appeal including the experience of using it. Many factors come into play and the weaknesses in one aspect can be outweighed by the strengths in another. Minor difficulties might be overlooked in the overall assessment, or blown out of all proportion. The overall impact of a product comes with reflection, retrospective memory, and reassessment.



Figure: Metro on a Windows Phone. Microsoft’s Metro UI on a Windows Phone. –Image Credit: Microsoft.

Norman touches on many aspects of design, and indeed, he is not focused specifically on software engineering, software artefacts, or the user interface. However, some of his thoughts and concepts are universally applicable and this is one reason ‘emotional design’, more than his other books, has influenced the user experience domain.

Norman also has something to say about design by committee; [which we have come across before](#)). This can be summarised as ‘appropriate use’, in that Norman is a proponent of user centred design by which he believes behavioural aspects of the product can be tested with many users – iteratively refined over many users and

evaluation. However, he also agrees with Henry Lieberman regarding the design of a visceral or reflective experience by committee:

"The brilliant conceptual artist Vitaly Komar and Alex Melamid conducted surveys asking people questions like, what's your favourite colour? Then they produced exhibitions of perfectly 'user centred art.' The results were profoundly disturbing.

The works were completely lacking in innovation or finesse of craftsmanship, disliked even by the very same survey respondents. Good art is not an optimal point in a multi-dimensional space; that was, of course, their point. Perfectly 'user-centred' design would be disturbing as well precisely because it would like that artistry."

Norman espouses user centred design for behavioural aspects only and feels that a single cohesive individual vision is required for emotional design based on visceral and reflective outcomes. This ties in with Lieberman in that removing behavioural from the truly emotional aspects of the design means that the entire design is not 'perfectly' user centred.

In this chapter more than any other, I would not advise a user-centred approach to the design of the affective user experience. This should be an individual vision, as the UX'er, it is your job to select the best person to fulfil that vision based on your clients brief; and this is best done individually, without a committee.

Finally, Norman seems to come to a conclusion that we are all designers, from the personalisations we make to our houses, to the cars we drive chosen as representations of a personality, even down to the way we organise our desks – these are all design issues. Norman logically concludes that personalisation and customisation are key aspects of the emotional design process, mainly because we are all predisposed to make these personalisations ourselves anyway. This is how we become more emotionally attached to the item; by modification. Only in this way can we move from emotional design and affective use to the much more important *emotional value*.

Norman suggests that it is impossible to build a mass produce item that fits every individual precisely. But that there are five ways to deal with this problem: the first is just to live with a personalised item; the second is to customise, the third is to customise mass production items, the fourth is to design our products, and the fifth is to modify purchased products. Either way, it seems that Norman suggests the value of emotional design can be captured in its ability to tailor itself to you viscerally, behaviourally and reflectively.

Norman opens and closes his book with a quote from William Morris: *"If you want a golden rule that will fit everybody, this is it: have nothing in your house that you do not know to be useful, or believe to be beautiful."*

We've looked at what it means to be useful, now let us look at what it means to be beautiful.

12.1.1 Beauty

The visual appearance of interactive systems tends to convey more information than we imagine [Fishwick, 2006]. Contact points that users first make to perceive such a

system affect the rest of the interaction process. Studies show that distinct layout, smart captions and interesting pictures can transform a wall of dry text into a presentation that users will approach enthusiastically. In this way, two components of visual beauty, visual complexity and visual aesthetics, are so closely related that it is necessary to discuss both.

12.1.2 Visual Complexity

Complexity can be defined as ‘the degree of difficulty in providing a verbal description of an image’ (how easy is it to describe [Figure: Windows Phone 7.5?](#)). Textures with repetitive and uniform oriented patterns are less complex than disorganised ones. A visual pattern is also described as complex if its parts are difficult to identify and separate from each other. Perception of image complexity depends on the extent to which objects are grouped, the quantity of the parts an observer perceives in the scene, familiarity with the scene and existing knowledge of objects inside the scene. Visual complexity is mainly represented by the perceptual dimensions of quantity of objects, clutter, openness, symmetry, organisation, and variety of colour. Studies also report a positive relation between visual complexity with the computer-generated bitmap size that is measured by its compressed file size as well as with the emotional and psychophysiological effect of the image. Besides, Tuch et al. demonstrate that the visual complexity of websites has multiple effects on human cognition and emotion, including experienced pleasure and arousal, facial expression, autonomic nervous system activation, task performance, and memory.

Further, visual perception is affected by cognition, content and form. Human cognition affects how a user retrieves and uses information in an interface. Content, interface components, and the amount of information that is available affect complexity since they can cause information overload. The form of the interaction on the user interface, navigation, and structure is a further factor that affects complexity. Indeed, studies try to identify interface design metrics that predict whether an interface is highly rated concerning complexity. These studies relate design guidelines with complexity explaining that the way an interface is presented depends on the way the interface itself is designed and what elements (metrics) are used. Others propose cognition, content and form as the three major factors that impact the complexity of an interface.

Concerning the Web, Ivory et al. perform a quantitative analysis of Web page attributes (e.g. number of fonts, images, and words) using a large collection of sites. These sites were judged based on content, structure and navigation, visual design, functionality, interactivity, and overall experience and were selected from categories such as financial, educational, community, health, service and living. They found that ‘Web pages and sites differ from each other on many dimensions, such as layout quality,



Figure: Windows Phone 7.5. Windows Phone 7.5 with Metro and Active Tiles. —Image Credit: Microsoft.

screen coverage and information quality'. In the same study, metrics concerning page composition, such as word and link count, page formatting, such as emphasised text and text positioning along with overall page characteristics, such as page size, help to predict (with 63% accuracy) whether a page can be ordered similar to that of human judges. Using the first study and the same characteristics of the page-level elements, they developed profiles of the Web pages based on the overall ratings that distinguished between good and bad pages about design and usability.

These studies were part of a project that tried to develop techniques to investigate empirically all aspects of Web site design but did not attempt to define and investigate visual complexity. However, I would suggest that the visual complexity of an interface depends on the presentation of the interface's components and by the density and diversity of the components that are presented, and that these aspects also overlap with visual aesthetics.

12.1.3 Visual Aesthetics and Design

Aesthetics is commonly defined as the appreciation of the beautiful or pleasing but the terminology is still controversial [Lavie and Tractinsky, 2004]. Visual aesthetics is 'the science of how things are known via the senses' which refers to user perception and cognition. Current findings suggest that good visual aesthetics enhances positive feelings toward applications. Several studies have established that aesthetic opinions about interfaces are formed quite quickly, and do not wane immediately; this suggests that aesthetic considerations should be seen as an important aspect of any application life cycle.



Figure: Jacobsen's Framework for Psychology of Aesthetics. Jacobsen's Framework for Psychology of Aesthetics. —Image Credit: Jacobsen.

Since several factors influence aesthetic preference, Jacobsen argues for a multi-disciplinary approach to the study of aesthetic preference in experimental psychology. He proposes a seven-fold framework that incorporates these different perspectives to un-

derstand how aesthetic opinions are processed. These different facets may also interrelate (see [Figure: Jacobsen's Framework for Psychology of Aesthetics](#)), and include:

- Diachronia: Aesthetic preferences may change with time;
- Ipsichronia: Social/cultural processes may shape a person's aesthetic opinions;
- Mind: An individual's mental model of the visual stimulus or emotions could influence aesthetic judgements;
- Body: Brain activities could affect aesthetic evaluation processes;
- Content: The stimulus being evaluated could influence aesthetic processing;
- Person: The evaluator's background may play a role in aesthetic preference; and
- Situation: The surrounding circumstances (this includes time and place) could influence aesthetic choices.

Empirical studies on visual aesthetics have followed one of two directions. There are those studies that have concentrated on replicating experiments to validate visual aesthetic theories. In such studies, interfaces are used as visual stimuli in place of works of art and geometrical shapes which served as test beds in the early days of experimental aesthetics. One popular aesthetic theory that has been investigated is Berlyne's arousal theory. Berlyne's theory holds that people love to experience aesthetic pleasure at moderate levels. There are also studies that have focused more on investigating the relationship between visual aesthetics and different aspects of user experience. Some of the aspects of user experience investigated so far include usability, credibility, desirability, complexity and accessibility.

The use of visual aesthetics can alter the perceptions of the viewer by manipulating visual components such as colours, text style and size, images and animations. In this way, the user is unknowingly or unconsciously involved with the message of the software artefact. The visual appeal of an interface can be assessed within 50 milliseconds in which time users make their impression about the software. Studies have established that a positive relationship exists between visual aesthetics and usability when beautiful interfaces were perceived to be easier to use (as we have discussed in the introduction). Users' judgement about software credibility is also shown to be formed within the first few seconds where an interface with high aesthetic treatment was judged as having higher credibility. The consistency of these aesthetic impressions were evaluated and it was established that these impressions did not wane immediately. That is, an interface that was perceived to be visually pleasing at first exposure, continued to remain attractive to the user for longer durations of exposure.

Visual clarity and richness were shown to be two of the most important aesthetic dimensions that strongly influence software users. Visual clarity refers to 'classical' aesthetic qualities pointing to clean, clear, and symmetric designs. Visual richness refers to 'expressive' aesthetics such as creativity, originality of the interface's aspects and designers' finesse and ability to 'break design conventions'. Karvonen also reports a case where 'design quality' emerged as one of the six factors that enhanced 'on-line trust', suggesting that users would more readily transact with beautiful software.

12.2 Narrative Art and Visual Aesthetics

The purpose of art and design is to alter the perceptions of the viewer by manipulating visual components. The main way to accomplish this is to create a narrative that the work refers to, or a narrative that is created by moving in sequence through the visual components of the work. Narrative art, art that tells a story, can be found as far back as the Palaeolithic period. Palaeolithic peoples were nomadic hunters and gatherers who lived by sheltering in caves, using fire, and fashioning stone tools. By the Upper Palaeolithic, there is evidence of communal hunting, constructed shelters, and belief systems centring on magic and the supernatural. Palaeolithic Art, produced from about 32,000 to 11,000 years ago, falls into two main categories: Portable Pieces (small figurines or decorated components carved out of bone, stone, or modelled in clay), and Cave Art; reaching their peak in the Magdalenian culture of Cro-Magnon man. Of the cave art currently discovered a majority seems to have narrative features such as the examples at Chauvet, Altamira, Pech Merle, and Lascaux. These narratives depict spiritual and ritualistic activities, hunting and mapping (see [Figure: Hunters with Elephants](#)), and animals of veneration. In these cases, the artist is conveying a narrative to the observer, which may be centred on aspects such as good hunting grounds, what animals the observer can expect to find, the kind of predators that are in the area, and the spiritual nature, and offerings required, for good luck. The sequence of the visual components which form the overall narrative is as important, or more important, than the aesthetic alone.

Most art, especially Western art is narrative in nature, depicting stories from religion, myth and legend, history and literature; with audiences being assumed to be familiar with the stories in question. In modern art, however, formalist ideas have resulted in narrative being frowned upon. However, the visual narrative is still present in the coded references to political or social issues, or to events in the artist's life. Such works are effectively modern allegories, and require information about the artist to be fully understood. This can be clearly seen in [Figure: St. Simon's Church](#); in which the artist takes the observer through a sequence of visual components which raise questions as the story is 'built'. In this case, the observer is first directed to the church (central, large, imposing) then by the placement / framing of the small slivers of buildings to the left and right, down the street to the right and out into the landscape. The observer then focuses on the smaller detail including the people in the front of the picture moving back (again down the street to the right). Indeed, the placement of the people, lamp-post, and street orientation draws a diagonal line from bottom left to mid right along which the eye follows, and by which the visual narrative is conveyed; this also means that the figure at far-mid-left is 'hidden'.

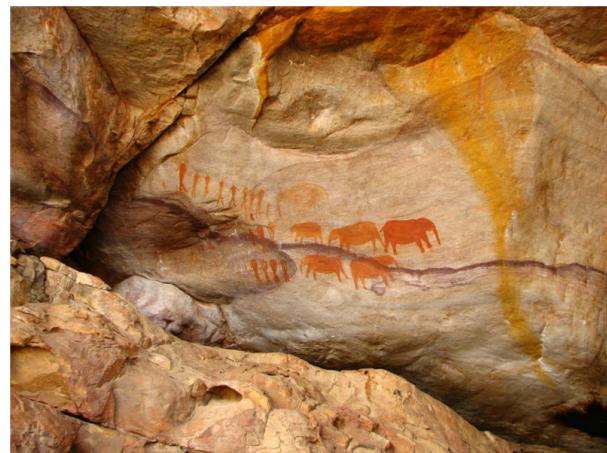


Figure: Hunters with Elephants. Hunters with Elephants – Cederberg Mountains, South Africa – Image Credit: Wikipedia.

The principles of the visual narrative are all captured and used by the graphic, and specifically the UX designer. Trained in the visual medium, and the core principles contained within this medium, designers use colour, text style and size, images and animations to alter the perceptions of the viewer taking them on a narrative journey through an interface. In this way the user is unknowingly or unconsciously involved with the message of the software artefact, ergo the narrative, via serialisation of the visual components. In the visual aesthetics survey of Hoffman and Krauss, the authors point to studies that try to determine user's perceptions of the aesthetic qualities of interfaces. These studies showed that visual clarity (clean, clear, and symmetric designs) and visual richness (creativity and originality) are two of the most important aesthetic dimensions. Both are important to users in unconsciously understanding the visual narrative.

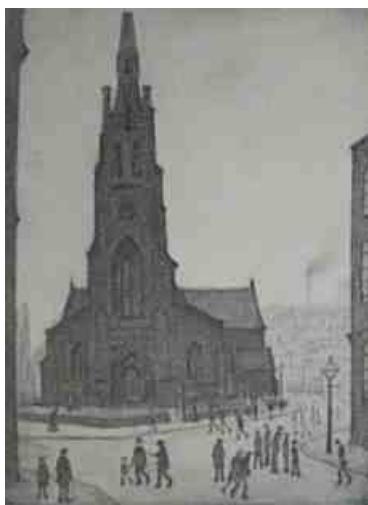


Figure: St. Simon's Church. St. Simon's Church. –
Image Credit: Lowry Collection.

We can see that the affective component of the user experience is to some extent determined by the arrangement of elements or details. It is the 'creative art of executing aesthetic or functional design and it is clear that both interface and interaction design can either facilitate or impede a user through the available functionality.

12.3 Visual Attention

As we have seen, interface design is determined by the arrangement of elements or details on the canvas that can either facilitate or impede a user through the available functionality. This is because the amount of information

on the canvas that reaches our eyes is far greater than our brain can process, and it is for this reason that visual attention is key to good design. Selective visual attention is a complex action composed of conscious and subconscious processes in the brain that are used to find and focus on relevant information quickly and efficiently. There are two general visual attention processes, *bottom-up* and *top-down*, which determine where humans next locate their attention.

Bottom-up models of visual attention suggest that low-level salient features, such as contrast, size, shape, colour, and brightness correlate well with visual interest. For example, a red apple (a source of nutrition) is more visually salient, and, therefore, attractive, than the green leaves surrounding it. Top-down models, on the other hand, explain visual search driven by semantics, or knowledge about the environment: when asked to describe the emotion of a person in a picture, for instance, people will automatically look to the person's face.

Both forms of visual attention inevitably play a part in helping people to orientate, navigate and understand interfaces. Bottom-up processing allows people to detect

quickly items such as bold text and images, which help to explain how the content is organised. It also helps people to group the information into ‘sections’, such as blocks of text, headings, and menus. Top-down processing enables people to interpret the information using prior knowledge and heuristics.

It can be seen from eye-tracking studies that users focus attention sequentially on different parts of the page, and that computational models have been successfully employed in computer graphics to segment images into regions that the user is most likely to focus upon. These models are based upon a knowledge of human visual behaviour and an understanding of the image in question. Indeed, studies exist which record user’s eye movements during specific browsing tasks, to find out those features on a Web-page that were visited, in which order and where ‘gaze hotspots’ were found. In these data we can see an association between the interface components and eye-gaze, but not one as simple as ‘the user looked at the most visually obvious interface features’. Indeed, sometimes a headline next to an attention grabbing feature, such as an image, was fixated upon. However, we can deduce that, for instance, the information in some text may not itself draw a user’s attention but ideally has some feature nearby which do. This idea is supported by other studies that try to create interface design metrics that predict whether a site is visually complex or not. These studies relate to interface design with complexity explaining that the way an interface is perceived depends on the way the interface itself is designed and what components are used.

By our understanding of visual attention, we can see that there are levels of granularity associated with visual design. This granularity allows us to understand a large visual rendering by segmentation into smaller more manageable pieces; components or components. Our attention moves between these components based on how our visual attention relates to them, and it is this knowledge of visual attention, captured as design best practice, which allows designers to build interfaces that users find interesting and easy to access. The serial narrative the designer builds for the observer is implicitly created by the visual appearance (and, therefore, attraction) of each visual component (dovetailing into our understanding of attention).

12.4 Collated Affective Concepts and Touch-points

As with other collated material, the first thing to notice about affective concepts is that there are many of them and many different ones are proposed in many different sources. However there are differences from usability principles, in that these are concepts and not principles - mainly because most work still sits in the psychology literature and not the nascent user experience domain. Indeed, this is why I’m calling them concepts (at this point) and not principles - it would be difficult to build testable principles from the concepts listed in ‘Table: Collated Affective Concepts’ without some additional processing. And herein lies the issue, for the most part, the concepts coming up (even when I turn them into principles) are not likely to be directly testable – in fact, it is accepted within the domain that we probably won’t be able to check that a principle is present, but we may be able to check that the expected affect is absent.

As with ‘Table: Collated Usability Principles’, you will notice in ‘Table: Collated Affective Concepts’ that the left column describes the concept (these are sometimes used

interchangeably between the different authors)³; while on the right side the authors are listed along with a footnote pointing to the text from which the concept is derived. In collating these concepts, I have not followed slavishly the nomenclature proposed by each author but have instead placed them in categories that I believe have the same conceptual value even if the naming of that concept does not follow that in the source.

Table: Collated Affective Concepts. Affective Concepts Collated by Source.

Concepts	Appears in Source
Aesthetic	Sharp, Rogers and Preece ⁴ ; Nielsen ⁵ ;
Emotionally Fulfilling	Norman ⁶ . Sharp, Rogers and Preece; Khaslavsky & Shedroff ⁷ .
Enjoyable	Sharp, Rogers and Preece.
Entertaining	Sharp, Rogers and Preece.
Enticing	Khaslavsky & Shedroff.
Facilitate Touch-points	Reeves & Nass ⁸ ; Fogg ⁹ ; Jordan ¹⁰ .
Flow (Enhance)	Csikszentmihalyi ¹¹ .
Form Relationship	Khaslavsky & Shedroff.
Fun	Sharp, Rogers and Preece; Norman.
Helpful	Sharp, Rogers and Preece.
Memory (Evocation)	Norman.
Minimalist Design	Nielsen.
Motivating	Sharp, Rogers and Preece.
Personalisation & Customisation	Norman.
Pleasing	Jordan; Norman.
Rewarding	Sharp, Rogers and Preece.
Satisfying	Sharp, Rogers and Preece; Norman.
Self-Image	Norman.
Supports Creativity	Sharp, Rogers and Preece.

You'll also notice that I mention 'Touch-points' in [Table: Collated Affective Concepts](#), and by these I mean various human traits you should be aware of when designing for emotional experience. These considerations mainly focus on the kinds of pleasure or dynamics we need to induce in the human for successful emotional engagement. Jordan [[Jordan, 2003](#)] (and to some extent Norman [[Norman, 1988](#)] – who also provides a nice interpretation) see these as:

- Physio-Pleasure: Anything perceptible by the senses - and includes movement – for UX the main points here are visual appearance and auditory output - maybe haptics at the very edge of development;
- Socio-Pleasure: Good interaction with other – again for UX this can be translated into the software domain, we are more engaged if we can good clear and worthwhile communication;

³We'll call them Principles later on.

⁴H. Sharp, Y. Rogers, and J. Preece. Interaction design: beyond human-computer interaction. Wiley, Chichester, 2nd ed edition, 2007.

⁵J. Nielsen. Usability engineering. Academic Press, Boston, 1993.

⁶D. A. Norman. Emotional design: why we love (or hate) everyday things. Basic Books, New York, 2004.

⁷J. Khaslavsky and N. Shedroff. Understanding the seductive experience. Commun. ACM, 42:45–49, May 1999.

⁸B. Reeves and C. I. Nass. The media equation: how people treat computers, television, and new media like real people and places. CSLI Publications, Stanford, Calif., 1996.

⁹B. J. Fogg. Persuasive technology: using computers to change what we think and do. Morgan Kaufmann Publishers, Amsterdam, 2003. Touch-points include – Physical, physiological, language, social dynamics, and social roles.

¹⁰P. W. Jordan. Designing pleasurable products: an introduction to the new human factors. Taylor & Francis e-Library, London, 2003. Touch-points include – physio pleasure, socio-pleasure, psycho pleasure, Ideo pleasure.

¹¹M. Csikszentmihalyi. Flow: the psychology of optimal experience. Harper & Row, New York, 1st ed edition, 1990.

- Psycho-Pleasure: Reactions to the system, how it interacts with the user and how the user feels about those interactions; and
- Ideo-Pleasure: How does the user see the software and their interaction with it. How do they perceive the interface and how does that feed into their self-image and positive memories.

Further, Khaslavsky and Shedroff [**Khaslavsky and Shedroff, 1999**] tell us that: “*The seductive power of the design of certain material and virtual objects can transcend issues of price and performance the buyers and users alike. Too many an engineer's dismay, the appearance of a product can sometimes make or break the product's market reaction. What they have in common is the ability to create an emotional bond with their audience...*” and come up with:

- Enticement: The interface should be enticing before any interaction even commences – there is a food related quote that we ‘eat with our eyes before our mouth’ – this is the same but for the user experience – we perceive a quality interactive experience before we embark on it;
- Relationship: We envisage a relationship, both before it occurs, and then while it is forming; and
- Fulfilment: Are our expectations fulfilled, if not then we have a problem at the UX level as an unfulfilled implied promise can often create more negative memories than the same bad experience but without the enticement.

Finally, Fogg [**Fogg, 2003**] elaborates on Reeves and Nass [**Reeves and Nass, 1996**], by suggesting five primary social Touch-points that people use to infer sociability with a device with which they are interacting:

- Physical: Again, perceptible by the senses - and again includes movement;
- Physiological: Personal feelings, emotions, and thoughts;
- Language: Spoken and written language;
- Social Dynamics: Cooperation, turn taking, collaborative activity; and
- Social Roles: The type of role, our authority, the interaction is occurring with – gives context to the interaction.

We can see that there are plenty of overlaps here, which you should be aware of. Now, as previously, I’m going to discard some of these concepts and touch-points because I think they are very niche because I don’t agree with them, because I don’t think they are helpful, or because there seems to be little consensus.

- First I’m combining ‘Enjoyable’ and ‘Entertaining’ and addressing those later.
- Now I think that ‘Forming Relationships’ is a great idea but I think to move this into a tangle principle, as is just too ambiguous.
- ‘Fun’, well we all like to have fun but I think this belongs in the next chapter.
- I think we have already covered ‘Helpful’.
- Again, ‘Memory (Evocation)’ has been covered.
- Next I’m removing ‘Motivating’ as this is a part of ‘gamification’ (**Don’t Panic – we’ll look at this later**).
- ‘Personalisation’ and ‘Customisation’ has been extensively covered.

- ‘Self-Image’ is a deep concept and asking the software to enhance self-image is just a little too much.
- Finally, ‘Supports Creativity’ is not generally applicable - not all software should do this and it is not appropriate in most situations.

Now the ‘carnage’ is over, let’s look at the concepts and touch-points that will stay and be amalgamated:

- ‘Pleasing’ + ‘Enticing’. Ensure that the system is pleasing and enticing, not just concerning the usability of the experience but concerning how the user anticipates and enjoys the system from an emotional standpoint.
- ‘Flow’ [[Csikszentmihalyi, 1990](#)] + ‘Facilitate Touch-points’. The flow and touch-points seem related in that touch-points in some way define the direction of the flow. In this case, we should plan to enhance the interactive flow to increase positive emotional responses.
- ‘Aesthetic’ + ‘Minimalist Design’. Both are related with minimalism being the servant of aesthetics. In this case, we need to make sure the design is aesthetically pleasing with a reduction in complexity by adopting a minimalist design.
- ‘Satisfying’ + ‘Rewarding’ + ‘Emotionally Fulfilling’. These three concepts all seem to be related to me. This means the interface, and its interactivity, should be satisfying and fulfilling - you should find your interactive experience to be rewarding at the end and after the interaction has finished.

Before we move on to turn these concepts and touch-points into principles (with a whole heap of caveats) lets first discuss ‘affective computing’, and then briefly discuss ‘trust’.

12.4.1 A Note on Affective Computing

In the future, you may come across the term ‘affective computing’, and begin to ask yourself what is its relationship to the user’s affective experience. ‘Affective computing’ is a term originally coined by Rosalind Picard at MIT in the mid-1990s and is used to describe a set of computing technologies that centre around. The detection of emotion, the ability for computers to express emotion, and in some cases the ability of computers to have emotions (in some form). You can see that this kind of emotional computation is very different from that of the emotional user experience we are studying. However, a part that is interesting to us is that centred around the detection of emotions.

As we shall see, this is because one of the only ways in which we can confirm emotional responses is by detecting them in the user. Simply, we can validate that our ‘effective’ principles are present and correctly captured in software because our users should be able to use the system. We can test our efficient principles are captured in software because we can measure task completion times. And we can test that our affective principles are captured in software if we can detect emotion. You’ll notice that I say ‘if’, as the detection of emotion is by no means trivial or absolute, and has still not moved fully into development or the practitioner domain. However, [we will investigate](#) two possibly useful emotion detection methods, being galvanic skin response (GSR), and facial expression recognition. It is unlikely that in real-world situations you will

have access to GSR or facial expression recognition software. However, it is useful to understand these now because they may become more prevalent in the future.

In reality, it is far more likely that we will be able to understand if the user has had a positive emotional experience by looking for clues as to how they relate their experiences in a social setting (by using systems that allow social feedback such as 'Star' and 'like' ratings). However, in all cases, you should realise that you will not be able to test the presence of each principle individually. As an emotional response to a system is a combinatorial construct it will be difficult to say whether a minimal aesthetic design is more useful, or gives a more emotional response in your system, than say, our satisfying and rewarding concept. What you will be able to discover is if there is little, or negative, emotional response that will suggest that somewhere, one or more of these principles has not been correctly captured.

12.4.2 A Note on Trust

Trust is one additional aspect to consider when engaged in emotional design. You should realise that aesthetics defines how an interface looks and how it emotionally dovetails into a user's concept of trust. This is very important for most user interfaces, especially web-based ones. In the world that seems to be becoming more untrustworthy and remote (about social connection), trustworthiness is a key concept and that concept can be reinforced by applying emotional design techniques to enhance the trust a user feels for a system.

Many empirical studies found that users' perceived information obtained from a Website with good visual aesthetics to be more credible than that obtained from a Website with poor visual aesthetics, even if the two Websites in question had the same content. This was observed upon short exposure to the Websites. Consequently, Robins and Holmes argue that visual aesthetics is seen as an important aspect of interaction design as it plays more significant roles than a mere 'decorative' one.

Remember, trust is mainly formed as an emotion based on many imperceptible and intangible cues. Of course, trust can also be misplaced and indeed 'social engineering' is a technique used by unsavoury characters to explore to manipulate our trust for their gain.

12.5 Potted Principles of Affective User Experience

Understanding the affective user experience, and building systems that capture it is complicated, error-prone, and prone to misinterpretation. Emotion is all about subjectivity, and this subjectivity occurs temporally, culturally, and psychologically. Indeed, if we look back at responses from Law 2009 (see [Table: Twenty-three Statements About UX](#)) we see statements such as:

1. Fleeting and more stable aspects of a person's internal state (e.g.. needs. motivations) affect a person's experience of something;
2. UX occurs in, and is dependent on the context in which the artefact is experienced;
3. Prior exposure to an artefact shapes subsequent UX;

4. We cannot design UX, but we can design for UX;
5. UX is highly dynamic – it constantly changes while interacting with a product;
6. UX can change even after a person has stopped interacting with the artefact; and
7. UX is based on how a person perceives the characteristics of an artefact, but not on the characteristics per se.

...all get a high-level of agreement from UX professionals. Further, if we look back at '[Figure: Time Spans of User Experience](#)', we will notice that the user experience extends both before and after the actual usage, this allows a user to anticipate and reflect upon their user experience, and this is also significantly important in forming their desire to use that system, or return to using the system.

So when considering these upcoming principles you must remember: that their presence cannot be individually or directly tested; that you are far more likely to be able to understand if they are not present, than if they are present; they contain a number of subjective words (such as Good) which make them difficult to quantify; and the emotional experiences to which they relate are likely to be subjective, temporal, cultural, and based in the user's own psychology.

One final thing to remember, the emotional responses recorded from users varies based on the task and how emotionally engaged in that task a user is. Performance increases with physiological or mental arousal, but only up to a point, when levels of arousal become too high, performance decreases (see [Figure: Yerkes-Dodson Curve](#)). So when I use the terms 'good', 'reduce', 'maximise', etc., what I mean is, dead centre of the 'Hebbian version' of the 'Yerkes-Dodson'.

12.5.1 Facilitate Quality

Quality is always going to be a difficult principle to apply, or, at least, test for. As with most principles in this section, understanding whether quality is present, and how to include it within your interface design will mainly come with experience. It's difficult to define quality, but often we know it when we see it.

I think that most people understand quality from their own perspective, they

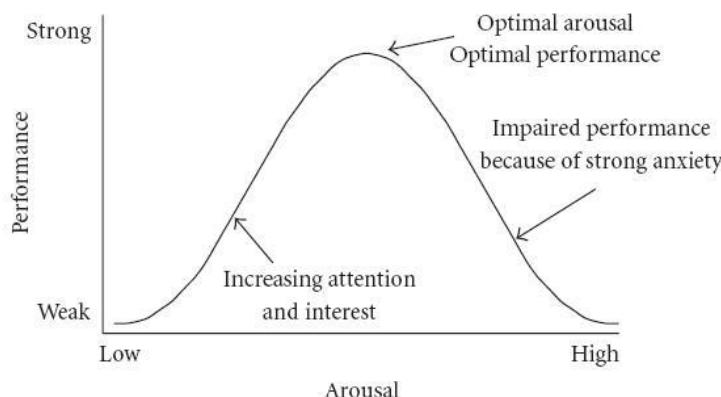


Figure: Yerkes-Dodson Curve. Hebbian version of the Yerkes-Dodson Law (this version leaves out that hyper-arousal does not adversely impact simple tasks). —Image Credit: Wikipedia.

understand if they are rushing work or that their work is not performed to the kinds of standards they might apply in other situations. The intangibility of the concept of quality means that the only way of knowing if the quality is being applied is your feeling and perception as a trained UX specialist. If you feel you do not have the ability to assess the quality of the application of the various emotional principles in this section,

then you should make sure your choices of the graphic designer, web designer, or product designer is good. Remember, you will not be on your own, but part of a team, you will mainly be interested in the overall architecture of the UX project but other team members who deal with graphics and design should also be present. Errors only normally occur in quality because it is intangible, and so its importance in creating emotional engagement is often overlooked.

Questions to think about as you design your prototype:

1. Do you feel that the interface exhibits current best practice?
2. Is the interface design fit-for-purpose for each stakeholder?
3. Did the best people for each job work on the interface and its interactions?
4. Is the underlaying code cleanly built?
5. Has quality been maintained at every level?

12.5.2 Facilitate Aesthetics

We have [already discussed](#), what it means, and the views of different experts around the subject. As we are already aware, there are no easy ways to understand aesthetics directly, or if the principle is captured as part of the design. Of course, it is testable in the long run because you can ask users survey questions to understand if they feel that the design is aesthetically pleasing or beautiful. Remember, an attractive interface has knock-on usability and efficiency benefits. This is because when we judge something to be aesthetically pleasing we are basing this judgement on our biological and cognitive evolution. Things that we judge to be aesthetic are pleasing precisely because we can use them more efficiently or easily. In this case, if you, or more likely your graphic designers, think that a website is aesthetically pleasing – because this is a shared concept, the interface is far more likely to be aesthetic as opposed to purely functional. As with quality, errors in applying this principle only mainly occur because a stakeholder (or a usability specialist) does not understand the importance of an aesthetically pleasing design.

Questions to think about as you design your prototype:

1. Is the design beautiful?
2. Does the design maximise enticement?
3. Does the visual design reduce complexity and is the design, minimalist?
4. Will the user perceive aesthetic quality?
5. Is the design current and does it convey the desired ‘image’?

12.5.3 Facilitate Flow

Flow is a difficult concept to convey – and [we'll be looking at it in more detail](#) – however, for this chapter, and in the context of the user experience, you should be striving to include some of the more ‘mechanical’ aspects of flow¹² within your design. Flow is all about providing the user with an optimal experience that they will find rewarding. This is difficult to do at the principal level. However, there are some aspects that can

¹²The concept of flow also impacts on [learnability](#), [progressiveness](#), and [situational](#).

help flow be achieved. Perhaps one of the most important is to encourage your users to participate in structured tasks that have clear goals and feedback, you should also cut down on the amount of extraneous information which is not part of the current flow activity so that the user can concentrate on the task at hand, and so that a users actions and their awareness of those actions become merged. These are difficult to accomplish, and it is, even more, difficult to understand if they are occurring. However, there are two ways in which you may be able to validate this principle in the future. The first is to see whether the user loses their self-consciousness when completing a flow task; the second, is to investigate if the users experience a transformation in time, such that their perception of their task completion time – which may have taken them five minutes – may actually be much less then reality – maybe only one minute or a few seconds.

Questions to think about as you design your prototype:

1. Does the visual flow support the interactive flow?
2. Do real world and virtual world touch-points drive the flow?
3. Is there a defined beginning and end?
4. Is there a narrative flow (people remember narratives better than instructions)?
5. Is there an absence of cyclic or repetitious flow?

12.5.4 Facilitate Pleasantness

In my egocentric view, the principle of pleasure and enticement, occur as a prefix to, and as part of the main use of the interface or interaction. We derive pleasure from anticipating the experience and progressing through a good experience. In reality, the terms you use do not matter and here I'm using the terms pleasurable and enticing as a hat-rack to hang the assertion that you must make sure that a user's anticipation and use of the system are emotionally satisfied.

Questions to think about as you design your prototype:

1. Do you expect this design to fulfil and please the user?
2. Will the expected emotions support positive anticipation?
3. Will they be satisfied as they progress through the interactivity?
4. Do you dovetail into their perceived satisfaction?
5. If nothing else will, the emotional responses here be positive?

12.5.5 Facilitate Satisfaction

Again I use satisfying, fulfilling and rewarding as a hat-rack to convey the idea that at the end of the interaction we should feel rewarded from an accurate execution of that experience; and that on reflection we should feel satisfaction and fulfilment from that rewarding experience. If [we look back](#), satisfaction and reward should occur.

Questions to think about as you design your prototype:

1. Will the user find the interactions rewarding?
2. Will the expected emotions support positive remembrances?
3. Will the user remember a pleasing experience, if the system is work based?

4. Are there any tangible rewards?
5. Have you allowed them to register satisfaction (or not) by using, say, a 'star' rating?

Remember, these are just the next five of our total principles - for the next batch you can skip ahead. But just because you can, maybe you shouldn't take your time to read the rest!

Caveat

'Ling's Cars'¹³ is a tricky site to pigeon hole into these principles (see [Figure: Ling's Cars](#)). If our principles and rationale are correct - no-one should go to this site, and no-one should like it; but they do go to the site, and they do like it – much to the consternation of the user experience Illuminati.

Why is this and what makes Ling's Cars a high traffic site – discounting people like me who are just there to look at the spectacle, or design – 'car crash', whichever phrase you prefer? There seems to be no real answer to this, the site itself breaks most of the principles of good design, usability, and accessibility. If I had to audit this site, I would imagine that most users would not want to interact with it, that most users would find it too complicated, and that most users would find it aesthetically displeasing. However, for some unknown reason, it seems to attract many users, who then go on to buy the cars Ling is selling, so what's the reason? Well this is the main question that has been on the lips of most UX people the world over, but there still seems to be little agreement.

I think that this site is all about Ling, to such an extent that the site has become a meme. This may be because it seems to be a window onto Ling's personality, that you get the feeling that Ling is in control of everything (even if those things are bad) and that she is present – all the time – behind the scenes with her tweeting and forum updates. Further, you may think that in these times of present economic unrest, where trusted organisations have become untrustworthy, this personal touch seems very attractive. It may be due to the current situation, the current zeitgeist of its users, its egocentricity, it may be an outlier, the one in a thousand which proves the experts wrong, and succeed when it should not.

However, let's consider this again, it's egocentric, it has Ling's presence, Ling is very much in control, it is her vision, and not that of a focus group, in short it has her personality. For good or for bad it has a personality, and that personality has more emotional value and emotional engagement, and then any of the other five principles listed above, this made me think there is one principle that is superior to the rest.

¹³Owned and run by Ling Valentine (aka 'Crazy Ling').



Figure: Ling's Cars. Ling's Cars. —Image Credit: <http://www.lingscars.com>.

12.5.6 Facilitate Personality

This principle is the most superior in this section, possibly of all the principles listed in this text. If your interface can exude personality, such that it seems to be a meme – or an avatar – of who you are, a direct window into your personality, then this principle is superior to the rest.

If your interface has personality, good or bad aesthetics, quality, flow, satisfaction, or fulfilment are not important; I'd probably even go as far as saying that usability is not important either. Personality trumps all the rest because it is the only one that can give the user an emotionally valuable engagement with the software engineering artefact. There are no questions for this principle, if it has a personality you'll know it!

12.6 Summary

As does Norman, let us finish with a quote: “*If you want a golden rule that will fit everybody, this is it: have nothing in your house that you do not know to be useful, or believe to be beautiful.*”

Here, Morris ‘hits the mark’ we can *know* things to be useful, but we can only assert *belief* that a thing is beautiful. In our case, this relates to our set of affective principles and how they exhibit: Quality; Aesthetics; Flow; Pleasantness; Satisfaction; and Personality. Remember, building systems that capture emotional concepts, is complicated, error-prone, and prone to misinterpretation. Emotion is all about subjectivity, and this subjectivity occurs temporally, culturally, and psychologically. When you think of subjective terms such as ‘good’, ‘reduce’, ‘maximise’, etc., then you should recall the Yerkes-Dodson Law; in this case, what you should mean is, dead centre of the ‘Hebbian version’ of the ‘Yerkes-Dodson Curve’.

Also, remember that if you are unsure of how to proceed, choose good co-workers with more experience in the area that you are trying to address - such as illustration, or graphic design (i.e., ‘**Creatives**’) - but keep track that they conform to the brief.

Finally, even though personality is king, pleasing, attractive sites work better and are more usable because they reflect the biological origins of the brain. Pleasing interactions and attractive interfaces produce positive emotions causing mental processes to be more creative and more tolerant of minor difficulties and faults within the interface design or the interactive experience. In short attractiveness affects human behaviour that therefore makes it a critical component of the design and implementation of the user experience.

12.6.1 Optional Further Reading

- [M. Csikszentmihalyi.] *Flow: the psychology of optimal experience*. Harper & Row, New York, 1st ed edition, 1990.
- [P. A. Fishwick.] *Aesthetic computing*. Leonardo. MIT Press, Cambridge, Mass., 2006.
- [B. J. Fogg.] *Persuasive technology: using computers to change what we think and do*. Morgan Kaufmann Publishers, Amsterdam, 2003.
- [P. W. Jordan.] *Designing pleasurable products: an introduction to the new human factors*. Taylor & Francis e-Library, London, 2003.
- [D. A. Norman.] *Emotional design: why we love (or hate) everyday things*. Basic Books, New York, 2004.
- [B. Reeves] and C. I. Nass. *The media equation: how people treat computers, television, and new media like real people and places*. CSLI Publications, Stanford, Calif., 1996.
- [A Miniukovich] and A De Angeli. “Computation of Interface Aesthetics.” *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015. doi:10.1145/2702123.2702575.



Self Assessment Questions

Try these without reference to the text:

1. Why is it difficult to know if the affective principles have been captured in software correctly?
2. Why is affective computing different to affective experiences?
3. How does Aesthetics and Visual Complexity relate to each other?
4. How does narrative art relate to the principle of Flow?
5. Why is Emotion difficult to quantify? What is one possible solution?

13. Principles of Engagement (Digital Umami)

Play has been seen as one of the most important and fundamentally human of activities.

– Johan Huizinga

I intended to call this chapter ‘Digital Umami’¹ to convey the concept of something that is imperceptibly delicious. However, after much more reading over the years I decided on ‘Dynamics’ in part from Fogg’s [Fogg, 2003] – elaboration on Reeves and Nass [Reeves and Nass, 1996] – Social Dynamics. But then just last year I realised it should be called ‘Engagement’ and so here we are. The topics we will be looking at here focus on fun, enjoyment, cooperation, collaborative activities, and what has come to be known as ‘gamification’.

Gamification is causing a degree of controversy at the moment, but what is it? Well, Wikipedia tells us that: “*The use of game play mechanics for non-game applications... particularly consumer-oriented web and mobile sites, to encourage people to adopt the applications. It also strives to encourage users to engage in desired behaviours in connection with the applications. Gamification works by making technology more engaging, and by encouraging desired behaviours, taking advantage of humans' psychological predisposition to engage in gaming.*

The UX camp, though, is split on its value. Pro-Gamification proponents argue “*As the critics point out, some gamified products are just poorly executed. Just because you saw something in a game once doesn't mean it'll be fun in your product. But I think that most of the critics of gamification fail to take into account the wide range of execution that's possible. Gamification can be applied as a superficial afterthought, or as a useful or even fundamental integration.*”²

While critics say: “*More specifically, gamification is... invented by consultants as a means to capture the wild, coveted beast that is videogames and to domesticate it for use in the grey, hopeless wasteland of big business... The rhetorical power of the word 'gamification' is enormous, and it does precisely what the 'they' want: it takes games – a mysterious, magical, powerful medium that has captured the attention of millions of people – and it makes them accessible in the context of contemporary business... Gamification is reassuring. It gives Vice Presidents and Brand Managers comfort: they're doing everything right, and they can do even better by adding 'a games strategy' to their existing products, slathering on 'gaminess' like aioli on ciabatta at the consultant's indulgent sales lunch.*”³.

¹A category of taste in food (besides sweet, sour, salty, and bitter), corresponding to the flavour of glutamate, especially monosodium glutamate. ORIGIN Japanese, literally ‘deliciousness’.

²Audrey Crane, A Gamification Framework for Interaction Designers, ARTICLE NO. 678 May 24, 2011, <http://tinyurl.com/bq57vg>

³Ian Bogost, My position statement at the Wharton Gamification Symposium, August 8, 2011, <http://bogo.st/wm>.

For a moment, however, let's just forget all the buzzwords and concentrate on the real concepts that underpin the ideas of 'gamification', 'funology', and 'social dynamics'. The key concepts [Anderson, 2011], as I see them, relate to a level of interaction beyond that which would normally be expected from standard interfaces and applications. This additional level may involve collaborations and partnerships with others, competitions, or the pursuit of a tangible prize or reward. But in all cases, a premium is placed upon the enjoyment, fun, and enhanced interactivity; in short 'engagement'⁴. What's more, I would also refer to my original thoughts on the subject – there is a certain added 'deliciousness', which seems to be imperceptible; as with Affective, 'You'll know it when you see it'.

13.1 Group Dynamics

Dynamics are discussed variously as 'Social Dynamics' [Fogg, 2003], 'Social Roles' [Reeves and Nass, 1996], 'Socio-Pleasure' [Jordan, 2003], 'Social Animals' [Wein-schenk, 2011], and 'Playful Seduction' [Anderson, 2011]; I'm sure you'll be able to find plenty more. However you conceive it, group dynamics is trying to 'firm up' quite a tricky and ill-defined concept – that of social interaction with other users, or via a more humanistic, naturalistic, or conversational interface. The idea is that adding the energy that is often found in human interactions, to our interactions with the application, will propel us into having better user experiences because they are closer to our expectations of person-to-person interactions.



Figure: Microsoft Courier Imitating a Real World Notebook. Microsoft Courier – Imitating a Real World Notebook. –Image Credit: Microsoft.

These more naturalistic interactions are based on our knowledge of human psyche and the human condition (one nice example is the Microsoft Courier Tablet – a prototype – see [Figure: Microsoft Courier Imitating a Real World Notebook](#)). We know that in most cases humans are socially motivated and live in groups with a strong tie to people in collaborative situations. Indeed, performing tasks in groups bonds humans to each other, as well as the group; in this case, there is an increased emotional benefit as people are comfortable feeling part of the whole. Further, we also know that people

⁴To urge, exhort, persuade, or induce –OED.

are likely to imitate and to empathise behaviour and body language; and our brains respond differently to people we know personally or don't know, and situations we know and don't know. It seems only appropriate then that we should work better in social situations where the interface can imitate some of these social niceties, or we can perform interactive tasks as part of the group, or interactions that make us feel like part of a group. Indeed, people expect online interactions to follow social rules, this is logical in some regard because the application software has been created by a human (who was also following a set of social rules).

Hierarchies often form as part of the social interactions for these groups, and so in the interactive environment, we need to decide whether we feel that the computer is a subservient to the user. Or whether the computer is a master and should be an autonomous guide with the user only consuming information with no regard as to how it was generated. However, from teammate research, the relationship seems to be one that should be a peer relationship. Certainly by creating a peer relationship we get the best of both worlds in that the computer can do much of the work we would not like to do ourselves while also dovetailing into the user's feeling of enhanced self-image and group membership. The computer is neither subservient, nor master, and this partnership of equals often works better for a task to be accurately accomplished.

In these cases, how may we pursue the dynamics via our interactions with the interface? Without a doubt, this is a difficult one to answer, and there seems to be no definitive idea so far. Most experts discuss this kind of group dynamics in the context of a pre-existing example, but are often very wary of creating practical principles or giving practical advice to developers and UX specialists; again most refer to this using the 'You'll know it when you see it' get-out-of-gaol-free card.

However, it seems that you may wish to think about going beyond what you would normally see as being the functional requirements of your interface. By knowing that we are interested in groups, collaborations with peer groups, that we feel more secure and more likely to perform better within groups, you may wish to include functionality that may not seem directly relevant but may increase performance. Here I'm thinking about systems that may link up members of a particular team, allow the team to see the results or the working practices of other teams, or to bond by friendly competition. There may be a facility for online chat or transcripts such that users can be supported within their team by each other. Further, the interface may conform to the social norms expected within the society it is deployed. In this case you may wish to alter your language files, or which profiles, to use common local dialects, colloquialisms, or jargon; we'll see this a little more detail [in the cockney rhyming slang ATM examples](#) next.



Figure: Microsoft Courier Exhibits a High Level of Dynamics.
Microsoft Courier – High Level of Dynamics. –Image Credit:
Microsoft.

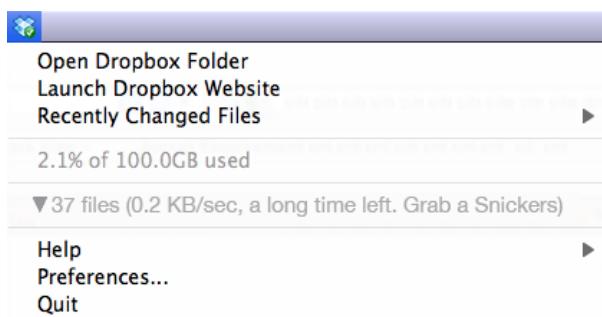


Figure: Dropbox Snickers. Dropbox Snickers. – Image Credit: Dropbox.

and ‘Techie Haiku’ to Amazon’s ‘Listmania!’ good group and social dynamics can be captured in most interactive software situations where there is a *perceived* positive outcome to the user. The main thing to remember, with most attempts at implementing group dynamics, is to be subtle and understated. If it is obvious that group dynamics is being applied with little gain for the user, their implementation is probably not going to be effective or produce the desired beneficial UX.

13.2 Funology

*“Play has been seen as one of the most important and fundamentally human of activities. Perhaps the best-known study on the subject of play is *Homo Ludens* by Johan Huizinga. In it, he argues that play is not only a defining characteristic of the human being but that it is also at the root of all human culture. He claimed that play (in both representation and contest) is the basis of all myth and ritual and therefore behind all the great ‘forces of civilised life’ law, commerce, art, literature, and science.” – Huizinga, 1950*

Funology [Blythe, 2003], is a difficult area to define because the majority of viewpoints are so very certain regarding the theories and frameworks they present, seemingly without any direct scientific support. Indeed, most of these seem to be derived from common sense, anecdotal evidence or, paradoxically, the view of a higher well-regarded source; with the results placed within a fun-o-logical framework. Indeed, even when user case studies are presented, most of the work points towards software and hardware development that does not have any summative user testing and relies solely on the views of a small set of individuals. In this way, it is very akin to the individualist experiences expressed as acceptable in Law 2009.

This may not be a problem for work coming from product design (the main proponents are from a product design background), and indeed, this fits in with the evolution of user experience and the parts of it which emanate from artefact design and marketing; there are very few hard tangible results here, however.

That said I still find the sentiment within this area to be important in that the experience of the user, their enjoyment, their delight, and the deliciousness of the software or application are often forgotten or ignored in a headlong rush to implement effective and

There are plenty of places where group dynamics already form an integral part of an application or interaction. These range from game based leaderboards found in an increasing number of online Web applications (such as [backcountry.com⁵](http://www.backcountry.com/store/rankings.html?page=1)) to the more subtle interactive group dynamics of the ill-fated Microsoft Courier Tablet (see Figure: Microsoft Courier Exhibits a High Level of Dynamics). From ThinkGeeks [OMGWTFUN⁶](http://www.thinkgeek.com/omgwtfun/) customer ‘Action Shots’

⁵<http://www.backcountry.com/store/rankings.html?page=1>

⁶<http://www.thinkgeek.com/omgwtfun/>

efficient interfaces. Indeed, this enjoyment aspect goes into my current thinking on the practice of interface and interaction engineering within the software engineering domain.

But how do we synthesise the useful aspects of mostly anecdotal evidence presented within this area? In this case my method is simple, I have listed all of the pertinent frameworks and well-found theories from each different author, and have then looked for similarities or overlap within the sentiment of these concepts. At the end arriving at a list of principles of enjoyment and experience and are much like those in accessibility or usability, which may be easily remembered within the development process.

- Personalisation and Customisation: ‘relevance’; ‘surpass expectations’; ‘decision-making authority of the user’; ‘appropriateness’; ‘the users needs’; ‘don’t think labels, think expressiveness and identity’; and ‘users interests’.
- Intangible Enjoyment: ‘triviality’, ‘enjoyment of the experience’; ‘users desires’; ‘sensory richness’; ‘don’t think products, think experiences’; ‘don’t think ease of use, I think enjoyment of the experience’; ‘satisfaction’; ‘pleasure’; and ‘appealingness’; and ‘emotional thread’.
- Tangible Action: ‘goal and action mode’; ‘manipulation’; ‘don’t hide, don’t represent, show’; ‘hit me, touch me, and I know how you feel’; ‘don’t think of thinking, just do’; ‘don’t think of affordances, think irresistible’; ‘evocation’; ‘sensual thread’; and ‘spectacle and aesthetics’.
- Narrative Aids Interaction: ‘don’t think beauty in appearance, think beauty in interaction’; ‘possibilities to create one’s own story or ritual’; ‘don’t think buttons, think actions’; ‘connection’; ‘interpretation’; ‘reflection’; ‘recounting’; ‘repetition and progression’; ‘anticipation’; and ‘compositional thread’.
- Mimic Metaphor: ‘metaphor does not suck’; ‘instead of representing complexity, trigger it in the mind of the user’; ‘think of meaning, not information’; ‘simulation’; ‘identification’; and ‘evocation’; and ‘spatiotemporal thread’.
- Communal Activity: ‘social opportunities’ in terms of ‘connectivity’ and ‘social cohesion’; ‘variation’; ‘multiple opportunities’; and ‘co-activity’.
- Learning and Skills Acquisition support Memory: ‘repetition and progression’; ‘develop skills’; ‘user control on participation’, with ‘appropriate challenges’; ‘the users skill’; ‘transgression and commitment’; ‘goal and action mode’; and ‘instead of representing complexity, bootstrap off it’.

There are of course many different ways that fun like elements can be applied, and here there can be a blur between funology and gamification. Indeed, it is probably not particularly useful to draw a hard and fast distinction between what will be fun and will be a game, but one of the most common places for a fun component to be introduced into an interaction is by using amusing/unexpected language or concepts. This may be as simplistic as Dropbox’s ‘Grab a Snickers’ when file transfers have a long time left to run (see [Figure: Dropbox Snickers](#)), through to HandBrake’s notification



Figure: HandBrake Cocktails. HandBrake Cocktails.
—Image Credit: HandBrake.

This may be as simplistic as Dropbox’s ‘Grab a Snickers’ when file transfers have a long time left to run (see [Figure: Dropbox Snickers](#)), through to HandBrake’s notification

to put down the cocktail you've been drinking while waiting for the HandBrake queue to complete (see [Figure: HandBrake Cocktails](#)).



Figure: Cockney ATM Language Selection. ATM in Cockney Rhyming Slang – Language Selection. – Image Credit: Wikipedia.

Of course, how to apply these will be different based on the individual development underway. For instance, 'Bank Machine', a cash machine operator, has introduced Cockney rhyming slang to many of its Automated Teller Machines (ATM) in East London. Cockney rhyming slang is a form of phrase construction in the English language and originates in dialectal British English from the East End of London. Slang examples include: 'dog-and-bone' = telephone; 'trouble-and-strife' = wife; 'mince pies' = eyes; and 'plates of meat' = feet. In this case, people using the ATM's can opt to have their prompts and options given to them in rhyming slang (see [Figure: Cockney ATM Language Selection](#)); as a result they will be asked to enter their Huckleberry Finn, rather than their Pin (see [Figure: Cockney ATM PIN](#)), and will have to select how much sausage and mash (cash) they want (see [Figure: Cockney ATM Cash Withdrawal](#)).



Figure: Cockney ATM Cash Withdrawal. ATM in Cockney Rhyming Slang – Cash Withdrawal. –Image Credit: Wikipedia.



Figure: Cockney ATM PIN. ATM in Cockney Rhyming Slang – PIN Entry. –Image Credit: Wikipedia.

It is, however, difficult to understand if this level of ‘fun’ is popular with users. Indeed, in some cases, it may be seen as patronising, or an obvious ploy to curry favour with a section of the community, which may lead to resentment by those other sections of the community not catered for. I would also suggest that testing for the presence of ‘fun’ will be difficult because a fun experience is very individualistic and often intangible. However, it may be easier to look for its absence - or barriers to its realisation - as an appropriate proxy measure of its presence.

13.3 Gamification

“Gamification typically involves applying game design thinking to non-game applications to make them more fun and engaging. Gamification has been called one of the

most important trends in technology by several industry experts. Gamification can potentially be applied to any industry and almost anything to create fun and engaging experiences, converting users into players.” –<http://gamification.org>

Gamification seems to include (I say seems as there are many different takes on this – as with much of this ‘Dynamics’ chapter): adding game-like visual elements or copy (usually visual design or copy driven); wedging in easy-to-add-on game elements, such as badges or adjacent products (usually marketing driven); including more subtle, deeply integrated elements like percentage-complete (usually, interaction design driven); and making the entire offering a game (usually product driven). I’d also suggest that gamification isn’t applicable all over.

It seems most gamification advocates suggest that game elements can be added in stages to a non-game interface. I think these can be separated into ‘Elementary’, ‘Bolt-On’, ‘Ground-up’:

- Elementary (Few Interactions are Gameplay Like): here I mean visual elements, badges, and cute phraseology – in some cases you could think of these elemental gamification points as more funology than gamification – or at least at the intersection of the two;
- Bolt-On: implies a set of game elements that are more deeply related to games and gameplay, but are easy to add to a pre-existing development and which imply progress and reward, such as leaderboards, stages and levels, percentage complete, product configuration (mimicking, build your character), etc.; and
- Ground-up (All Interactions are Gameplay Like): here the game elements were planned from the start and were indivisible from the main development⁷⁸, or an existing development has evolved (exhibited emergent behaviour) from the use of the game elements into a system that is gameplay at the heart of its interaction scenarios.

⁷⁷Such as ‘Waze’ which is gamified crowdsourcing of traffic data. Waze say that “Waze is all about contributing to the ‘common good’ out there on the road. By connecting drivers to one another, we help people create local driving communities that work together to improve the quality of everyone’s daily driving (thanks for pointing this ground-up gamified app out Nima Ara).

⁷⁸Kolibree is a connected electric toothbrush, equipped with sonic technology to make your teeth brushing more efficient. Its sensors interact with mobile apps for adults and children to make brushing an enjoyable experience. Kolibree helps the whole family learn about and improve their oral hygiene [Various, 2014a, 2006].

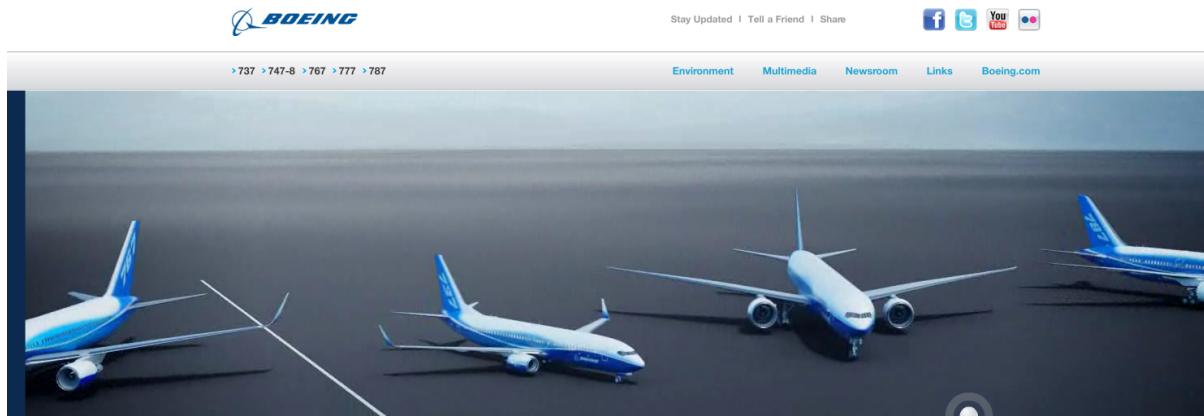


Figure: Boeing’s ‘New Airplane’ Site. Boeing’s Airliner Range – Displayed as animated game like components. –Image Credit: Boeing.

It is difficult to try and describe the kinds of game activities which can be added at different stages, or the tools and techniques available as the area is still pretty new; however an understanding of play [[Salen and Zimmerman, 2006](#)] will help. Roughly, you should be thinking of, illustrations, comic elements, realer-than-life 3D renderings, and known badged content if you intend to add elemental gamification to your interaction. In this case, you are just wanting to enhance the visual look and feel of your system by using elements that will be familiar to gamers. You can see this in [Boeing’s ‘New Airplane’ site⁹](#) which adds game like animated visuals to display its range of airliners (see [Figure: Boeing’s ‘New Airplane’ Site](#)).

Put your Timmy sticker on the map!

Our Timmy stickers are all over the world—and we want to see where! Add yours to our Sticker Map and be entered to randomly win a monthly \$100 ThinkGeek gift certificate. Stick Timmy with care and get started at:

<http://thinkgeek.com/stickermap>

Figure: ThinkGeeks’s Timmy Sticker Map. ThinkGeeks’s ‘Timmy’ Sticker Map. –Image Credit: ThinkGeeks.

[Timmy](#)) – and visually mimicking a ‘[Mechanical Turk](#)’. Further, once you receive your purchase, a ‘Timmy’ sticker arrives with exhorting you to place your location on the map, cleverly conjoining the real and virtual worlds and making your virtual interactions in some way feel realer and personal (see [Figure: ThinkGeeks’s Timmy Sticker Map](#)).

Finally, we come to ground-up gamification in which the game elements are indivisible from the software system (its interactions, interfaces, and ethos) itself. In honesty, I’ve

Now consider the next step to bolt on gamification, here additional elements are added to a pre-existing piece of software which mimic the structural elements of the game such as those associated with stages levels and leaderboards, etc., or whereby game like elements are added to increase the fun of interacting with your application. In this case consider the bolt-on gamification found in ThinkGeek, in which their reoccurring ‘fun’ character ‘Timmy the Monkey’ will decide the product you should purchase based on an answer to a question (see [Figure: ThinkGeeks’s Timmy Sticker Map](#)).

⁹<http://www.newairplane.com/>

seen no really good examples of this, which are both game and non-game (if that makes any sense).

Now, you should also know that I've recently been having a conversation on twitter about gamification in the context of UX [Zichermann and Cunningham, 2011]. Now I see myself as far more circumspect than most, in that I see gamification as a convenient term to describe 'digital umami'. The object of gamification is to make non-game software more engaging to users. So I think this is mainly only useful for applications that take relatively little interaction, and for systems that you may use a lot for a time but then stop – let's call it transient interaction or use. Many gamification aficionados suggest that deeper gamification will have better results – although I'm not convinced how deep this goes; although to argue against myself I could propose that non-game applications that use game interactivity (not just visual design) dovetails more efficiently into the pre-learnt interaction characteristics and expectations already familiar to game-playing users.

Now if we transfer the positive aspects of gamification do we also transfer the negative characteristics too. So what are these? I think games can be loosely characterised as having the following (possibly negative) properties:

- Increasing difficulty of task completion: Games rely on the increasing difficulty – in other words as you increase levels you increase the difficulty to maintain interest. This suggests to me that loss of interest may very well occur in the gamified domains too – yet usability will always want to reduce difficulty not increase it;
- Goal attainment: Once your goal has been attained you mostly stop using the game – goal attainment finally leads to non-use. This may be a real problem in the gamified world unless the outcome can be varied such that it may not be attained on a regular basis – that being the case the 'Real' goal must never be the gamified goal – but rather a stop on the path the gamified goal – which is transient, can change, or attainment can fail – while the real goal is always attained;
- Limited scope for different kinds of interaction: Game interaction is limited as indicated by the simplicity of the controller. Complex touch screens as on EPOS systems or Keyboard support for general purpose computing do not exist. This suggests that rich interactivity may not be possible;
- Boredom: Games fail over time as people become bored with the game – once they 'know' it, they often leave it;
- Little keyboard input: Keyboard input does not occur much in gaming and so the level of jobs that can be accomplished through gamification may be reduced or be at least inappropriate; and
- Non-use in general: A game is finite.

Consider an investigation I recently performed, comprising two ATMs – one gamified, one not – testing both these systems I found that the non-gamified system took 23 seconds to dispense cash and the gamified one took 36 seconds. You don't know this until you've used them once – but anecdotally, from a quick customer survey, people only choose the gamified one if there is no choice. In short, there may be many more problems for which we need to make accommodations. It seems that gamification may be useful to add a little spice but that without answers to the possible transfer of negative game playing traits the amount of value-added may turn out to be smaller than we imagine.

13.4 Collated Concepts of Engagement

Again, as with other collated material, the first thing to notice about concepts of engagement is that there are many of them and many different ones are proposed in many different sources. However there are differences from usability principles, in that – as with affective interaction – these are concepts and not principles. Remember, for the most part, the concepts coming up (even when I turn them into principles) are not likely to be directly testable. In fact, it is accepted within the domain that we probably won't be able to check that a principle is present. But – again as with affective interaction – we may be able to check that the expected affect is absent.

Again, you will notice in Table: Collated Dynamic Concept that the left column describes the concept (these are sometimes used interchangeably between the different authors)¹⁰; while on the right side the authors are listed along with a footnote pointing to the text from which the concept is derived. In collating these concepts, I have not followed slavishly the nomenclature proposed by each author but have instead placed them in categories that I believe have the same conceptual value even if the naming of that concept does not follow that derived in the source.

Table: Collated Dynamic Concepts. Dynamic Concepts Collated by Source.

Concepts	Appears in Source
Contextual Comms.	Duggan ¹¹ .
Challenge	Anderson ¹² ; Csikszentmihalyi ¹³ .
Delight	Weinschenk ¹⁴ .
Drive	Paharia ¹⁵ .
Encourage	Paharia; Anderson.
Engage	Paharia; Anderson.
Enjoyable	Sharp; Rogers and Preece ¹⁶ ; Weinschenk; Blythe ¹⁷ .
Entertaining	Sharp; Rogers and Preece; Csikszentmihalyi.
Enticement	Khaslavsky ** Shedroff ¹⁸ ; Crane ¹⁹ .



Figure: ThinkGeeks's Timmy. ThinkGeeks's 'The Amazing Timmy'.—Image Credit: ThinkGeeks.

¹⁰Again, we'll work them into Principles later on.

¹¹K. Duggan, Badgerville <http://www.badgerville.com/>.

¹²S. P. Anderson\index{Anderson, S. P.}. Seductive Interaction design: creating playful, fun, and effective user experiences. New Riders, Berkeley, CA, 2011.}

¹³M. Csikszentmihalyi. Flow: the psychology of optimal experience. Harper & Row, New York, 1st ed edition, 1990.}

¹⁴S. Weinschenk. 100 things every designer needs to know about people. Voices that matter. New Riders, Berkeley, CA, 2011.}

¹⁵R. Paharia, Bunchball <http://www.bunchball.com>.

¹⁶H. Sharp, Y. Rogers, and J. Preece. Interaction design: beyond human-computer interaction. Wiley, Chichester, 2nd ed edition, 2007.}

¹⁷M. A. Blythe. Funology: from usability to enjoyment, volume v. 3. Kluwer Academic Publishers, Dordrecht, 2003.}

¹⁸J. Khaslavsky and N. Shedroff. Understanding the seductive experience. Commun. ACM, 42:45–49, May 1999.}

¹⁹

Concepts	Appears in Source
Fun	Sharp; Rogers and Preece; Norman ²⁰ .
Goals	Csikszentmihalyi.
Interest	Weinschenk.
Learning	Blythe.
Look and Feel (Game)	Crane.
Metaphor	Blythe.
Motivating	Sharp; Rogers and Preece; Crane; Duggan; Weinschenk; Anderson.
Narrative	Blythe.
Personalisation	Blythe.
Progression	Anderson; Csikszentmihalyi.
Rewards	Zichermann and Cunningham ²¹ ; Duggan; Weinschenk; Anderson.
Social	Weinschenk; Anderson; Blythe.

We can see that there are plenty of overlaps here which you should be aware of. Now, I'm going to discard some of these concepts because I think they are very niche because I don't agree with them, because I don't think they are helpful, or because there seems to be little consensus.

- 'Challenge'. While challenge is a necessary component of play and games, it may not be useful in the non-game context, where I would see this as more related to goals, rewards, and motivation.
- 'Delight'. Delight is difficult to quantify, as with most concepts here, however, it seems that the light is better expressed in this case as enjoyment.
- 'Drive' + 'Encourage'. Drive and encouragement both seem to fit together, however, I think they are better expressed as part of the motivation, reward, and goals aspects of non-game software.
- 'Enticement'. We have [already covered enticement in our affective principles](#).
- 'Engage'. Engagement is, again, better represented in entertainment.
- 'Interest'. interest, and maintaining that interest, seems to be a concept that arises from a good application of the others, and so trying to design for this may not be appropriate.
- 'Learning'. We have [already covered enticement in our efficient principles](#).
- 'Look and Feel (Game)'. The aesthetic qualities and the look and feel of the application have [already been covered in our affective principles](#).
- 'Metaphor'. We have [already covered enticement in our efficient principles](#).
- 'Personalisation'. We have [already covered enticement in our effective principles](#).
- 'Tangible Action'. We have [already covered enticement in our efficient principles](#).

Now the 'carnage' is over, let's look at the concepts that will stay and/or be amalgamated:

- 'Contextual Communication'. Contextual communication, indeed communications, in general, is important for many aspects of interaction and interactivity, including engagement, and is particularly related to group dynamics and social interaction.

A. Crane, DesignMap <http://www.designmap.com/>.

²⁰D. A. Norman. Emotional design: why we love (or hate) everyday things. Basic Books, New York, 2004.}

²¹G. Zichermann and C. Cunningham. Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps. 2011. Particularly his SAPS (status, access, power, and stuff) model.}

- ‘Fun’ + ‘Enjoyable’ + ‘Entertaining’. These three concepts seem to go together in my opinion and so can be amalgamated into one principle (albeit incredibly difficult to test).
- ‘Motivation’ + ‘Reward’ + ‘Goals’. Again these three concepts are suitable for amalgamation and are mainly focused on dealing with propelling the user forward, based on some possible end-prize even if that prize is in some way intangible (such as an increase in status, etc.).
- ‘Narrative’ + ‘Progression’. Facilitating a user’s progression through interaction is of primary importance for good interactive dynamics. The progression may be reward based, based on fun, based on social and cultural norms, or indeed based on a combination of all three.
- ‘Social’. Social interactions are, again, very important for interaction and interface dynamics. This is because, as we have already seen, they dovetail into our natural social understanding.

13.5 Potted Principles of Dynamic User Experience

As with our [previous discussions](#), when considering these upcoming principles, you must remember: that their presence cannot be individually or directly tested. That you are far more likely to be able to understand if they are not present, than if they are present; they contain many subjective words (such as Good) which make them difficult to quantify. And the engaging experiences to which they relate are likely to be subjective, temporal, cultural, and based in the user’s psychology.

13.5.1 Facilitate Social Dynamics

We’ve already discussed social dynamics, contextual communication, and group dynamics, and so I don’t propose to rehash that information here. Instead, think back to participatory design and your interactions with your user group, the dynamics of the focus groups, and the interaction between users. This kind of feeling is what you’re trying to create in users of your system. It may be that the system facilitates support via automated social methods, or by real social methods such as Microsoft’s ‘AskMSR’. However, in both cases, you should be trying to make sure that users have an understanding that there are other people also interacting with the system, and who are immediately available for support and task collaboration.

You should also not underestimate the effect language can have on an interaction and users emotional engagement with the system. You can support this by adding amusing phrases – or enabling users to add these phrases, ‘in’ jokes or business jargon (understood by the group) are also a good way of making users think of the systems as a ‘person’. In short, including the social aspects will help to tie better the real world to the virtual world in a more tangible way.

Questions to think about as you design your prototype:

1. Do you include suitable functionality to facilitate collaboration?
2. Are aspects such as social communication accounted for?
3. Do you link the real and virtual to facilitate better user engagement?

4. Can team or group members interact and support each other?
5. Have you used language and terminology that users may find playful?

13.5.2 Facilitate Progression

Think back to all the fun and games that you have ever participated in. The commonality is that they all have a defined endpoint such that a reward can occur, they are based on some form of challenge, and that challenge can be mitigated by different levels (or mode) of ability. And, in most cases you have the ability to move through different phrases such that the challenge becomes increasingly difficult while still being tailored to your capabilities.

It may seem difficult to think about how this may be applied in a non-game like setting. However, it is possible with some judicious use of metaphor. Certainly, you could include the number of times, and the efficiency, a user has participated in a certain interactive task as an indicator of their expertise level. You could also move the user through a staged process, with a certain percentage completeness, to indicate moving to different levels or stages. And, finally, you may be able to display that user's information once they've achieved the final level on some communal leader-board, or some resource that represents status by rewarding that user with a specific designation (such as novice, intermediate, improving, expert) of their user skills.

Indeed, this kind of progression facilitates motivation, but also allows you to harvest data in a way that is more acceptable to the user. It is far easier to harvest information that you might find useful to modify the user experience, from a user who is trying to increase their percentage information complete for a perceived virtual reward (ThinkGeek do this with Monkey Points), than it is just to ask for the information outright. This kind of motivation also works in the real world, from a user experience point of view, this can be seen when trying to recruit participants for testing interactions. You'll get very few people who contribute a free 15 minutes of their time, but give them a Snickers bar (which cost you fifty pence) and they will spend an hour telling you everything you ever wanted to know about their experiences with your development. All this works just because you provide motivation by goals or rewards and then facilitate the users progression (in some way) toward those goals and rewards.

Questions to think about as you design your prototype:

1. Have you thought about attainment and goals, via stages and levels?
2. Do you facilitate motivation and reward?
3. Have you included a narrative flow through each interaction?
4. Are there opportunities for friendly competition?
5. Is progress also social?

13.5.3 Facilitate Play

The deeper aspects of play and gamification are incredibly difficult to define accurately, who is, however, more work on the play and the translation of games into a non-game environment. You can add deeper elements that are specifically game based,

but nothing like the normal interaction workflow, but progress the user a long that workflow via some means which dovetails into their understanding of what the play is; by mimicking a game. We have already seen that this kind of thing works with ThinkGeeks' 'Mechanical Timmy', 'Google' adds elements of fun by its random search, and 'Ford' allows you to customise a car in much the same way as you would customise a character in a game.

However you use these elements of play or game, you must make sure that they are in keeping with the interactive style and ethos of the interface and the interactions it facilitates; they should blend, add value, and assist the user does not jar the interactivity into a style other than that which was expected. In all cases this is subjective, and so – as with all of the principles within this section – you need to do thorough user validation before you can assert that your playful engagements are having a positive effect on the user experience.

Questions to think about as you design your prototype:

1. Is the look and feel playful and game like?
2. Have you included playful, and game like social elements?
3. Will users leave with a feeling of fun and enjoyment?
4. Are there playful games included, or game elements that make the interaction seem like a game, and not... work?
5. Do the elements of play, included, really enhance the user experience?

Caveat

The main thing to remember is that adding engagement such as social or group dynamics, funology, or gamification may just not be the right thing to do for your development. Indeed, without extensive user testing adding these elements may harm the user experience and not enhance it; in some cases one person is 'playful' is another person's 'frustrating'. This is why one of the [first principles we met](#) is very important because by enabling flexibility within the user experience we can account for the subjective and individual desires of the user, allowing game like elements to those who will most benefit.

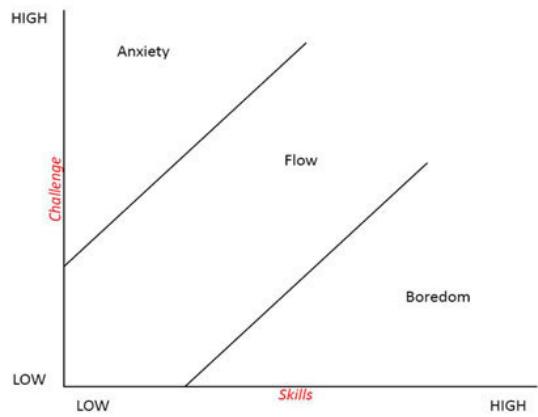


Figure: Csikszentmihalyi's 'Flow' Diagram. 'Flow' Diagram – M. Csikszentmihalyi. —Image Credit: Flow: the psychology of optimal experience. Harper & Row, New York, 1st ed edition, 1990.

Likewise, including one or all of the principles listed above, may again not be appropriate. Some parts of each principle – in a mix ‘n’ match fashion – may be appropriate, or indeed, a constrained set of principles (i.e. not all three) may also be more useful in enhancing the user experience. All this is based on your understanding of the application domain, as well as your understanding of how the principles should be applied, and the effects that those principles are having on the users.

Remember that as with all principles that relate to the subjective emotional, fun, and playful elements of development, you must pay specific attention to the target reactions of the users, and remember ‘good’ is not defined as more, but as the dead centre of the ‘Hebbian version’ of the [‘Yerkes Dodson Curve’](#). Indeed, this view is also backed up by work within the emotional literature too.

For instance, Csikszentmihalyi proposes a diagram (see [Figure: Csikszentmihalyi’s ‘Flow’ Diagram](#)) which places ‘skills’ against ‘challenge’. Here optimal flow is seen as an appropriate level of skill to an appropriate level of challenge, if there are too high a challenge and too low a skill-set than anxiety occurs; likewise if there is too low a challenge and to high skill-set then boredom occurs. This is similar to how we may think of naive and expert users and the system adaptations that should occur to support each.

In this case, adding too many opportunities for engagement may increase the boredom of the expert user who wishes to continue with their job as quickly as possible; conversely adding too few may make the naive user feel out of their depth. Of course, it may also be that expert users will also have better experiences with fun or playful dynamics as long as the task at hand is not time critical.



Figure: Normal Egg Cup. A ‘Normal’ Egg Cup. —
Image Credit: [Wikipeadia](#).

13.6 Summary

The benefits that may be derived from adding engaging aspects such as gamification, funology, and collaborative group elements are still very much tentative and anecdotal. However, it does seem clear that these kinds of elements are beneficial about the entire user experience, even if it is sometimes difficult to translate these engagement principles into live interfaces and interactions. While we can see that it may be useful to add elements of fun to the interface, to some extent the jury is still out on gamification. Indeed, you may notice that I have not touched on the wider topic of ‘Ground-up Gamification’ in which all interactions are gameplay like, specifically because its application is still contentious²². While there is much talk of this kind of interaction, and there are one or two (mainly website) instances of its occurrence, for the most part, it is my opinion that gamification from the ground-up equals a game, and is not equal a gamified non-game.

If you are finding it difficult to apply the principles as described in this chapter, I would suggest that you should still keep in mind the overriding design goals and intentions behind them. Think seriously about the fun elements of your design; make sure that you at least consider if game elements may be useful in helping – or encouraging – a user through a more complicated interaction; see if there is scope for collaboration and peer support within the system, and interactions, that you are building; and finally, try to the level of seduction and understanding of the humane aspects of the system which can so very often be ignored in the race for tangible effectiveness and efficiency.

In short, this chapter has been about the search for the intangibles that optimise our experiences and change a dry interface and interaction into one that joyful and is therefore actively sought. As a parting thought - which of these two pictures ([Figure: Normal Egg Cup](#) and [Figure: Fun Egg Cups](#)) of egg cups makes you smile?

13.6.1 Optional Further Reading

- [S. P. Anderson]. Seductive Interaction design: creating playful, fun, and effective user experiences. New Riders, Berkeley, CA, 2011.
- [M. A. Blythe]. Funology: from usability to enjoyment, volume v. 3. Kluwer Academic Publishers, Dordrecht, 2003.
- [K. Salen] and E. Zimmerman. The game design reader: a Rules of play anthology. MIT Press, Cambridge, Mass., 2006.
- [S. Weinschenk]. 100 things every designer needs to know about people. Voices that matter. New Riders, Berkeley, CA, 2011.



Figure: Fun Egg Cups. ‘Fun’ Cartoon Egg Cups. – Image Credit: Wikipeadia.

wider topic of ‘Ground-up Gamification’ in which all interactions are gameplay like, specifically because its application is still contentious²². While there is much talk of this kind of interaction, and there are one or two (mainly website) instances of its occurrence, for the most part, it is my opinion that gamification from the ground-up equals a game, and is not equal a gamified non-game.

²²...and because I'm not yet wholly convinced.

- [G. Zichermann] and C. Cunningham. Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps. 2011.



Self Assessment Questions

Try these without reference to the text:

1. What are the pros and cons of gamification?
2. How could you include social / group dynamics into your system?
3. How can you enhance the users perception of fun?
4. What is the 'skeptic' view of Gamification?
5. List a principle and describe it.

Part IV: Validating the User Experience

14. User Evaluation

Life is a full circle, widening until it joins the circle motions of the infinite

– Anais Nin

Evaluation methodologies are the principal means by which the UX specialist answers the questions ‘What needs to be done?’, and after an interface manipulation has occurred, ‘What is the effect of this manipulation?’

There are many ways to categorise different evaluation methodologies within UX. Here I equate naturalistic field evaluation with the qualitative methods of anthropologists, I see quantitative methods, characterised by sociological surveys, as being between fieldwork and laboratory experimentation ([we’re already looked at these](#)). Finally, I see laboratory-based evaluation, mainly in non-naturalistic settings, as being the main way of enacting true experimental methods most often used within experimental psychology and the cognitive sciences.



Links to Modelling Requirements

Before starting, it is useful to point backwards to [Modelling Requirements](#). The collection and modelling of user data shares similarities and also some of these techniques can be used in Modelling Requirements and vice-versa.

Each of these key categories has a unique way of viewing the world and a particular set of strengths arising from the kind of evaluation and answers needed. There are many debates within each particular discipline as to the kind of methods that give the best outcomes. However, in UX we can take a more pragmatic approach and select the appropriate evaluation method from each category, combining them all together in our final experimental design, so that we can build an end-to-end story.

If you notice, I’m using the term ‘evaluation’ quite frequently. This is because these methods all evolved from scientific evaluation methods within a particular academic discipline. But do not be confused, their applicability to real-world scenarios and their use within practical settings, of the kind we often face in application and interface development, is what has made them popular and particularly suited to the kinds of questions the UX specialist is required to answer.

Evaluation methods are slightly different from the kind of requirements analysis and requirements elicitation [scenarios that we have seen](#). As we state there, requirements analysis and elicitation, along with the models that are created from them, are often engineering – or craft based – as opposed to having their roots within empirical scientific methodologies; certainly [this is the case informally](#). In this case, they are more akin to the qualitative evaluation methods of participant observation, interviewing, or focus group discussion. However, these three methods are not bound with the same kind of system architecture design metaphors and methods as are those of requirements analysis. It is my opinion that this makes requirements analysis far weaker than the

evaluation methods we will discuss here. I would only hazard a guess, which this is the case because the final testing of the software system will validate or invalidate the requirements model, whereas, there is no such concept of a final implementation validating an evaluation method in the domain of anthology or sociology. This means that, if the model is wrong, a large amount of time is needed to fix it at the end of the project. In some cases, [this means that the interface is left unfixed because time and cost constraints dictate a speedy completion.](#)

Understanding the requirements – and having the tools to create experimental methods to test the interface is correctly created from those requirements – are key to the user experience. As such, this chapter should be one of your primary references when creating testing and evaluation plans involving the user.

14.1 Expert Evaluation via the Audit

Walk-throughs and heuristic evaluations are closely linked and listed here because they are slightly different from the other techniques introduced. They differ because they mainly occur before formal evaluation with participants and are often conducted by the evaluator, or the UX specialist responsible for creating the interface, themselves.

14.1.1 Walk-throughs

There are many flavours of walk-throughs including the cognitive walk-through and the barrier walk-through, along with the code walk-through. However, in all cases, the evaluator formally addresses each step of the system based on the interaction that is required and the system components that are required to enact that interaction. At each stage, the outputs, inputs, and performance will be evaluated, and this presupposes that the walk-through is far more about an evaluation of the system performing correctly than it is about the aesthetic or design nature of the interface itself. In this case, walk-throughs can also be used to understand how easy the system is to learn and whether aspects of usability, such as progressive disclosure, or accessibility are present. However, to create a reasonably accurate walk-through the evaluator needs to have a good understanding or description of the prototype system; a description or understanding of the task the users are to perform; the action that is required to complete the task; and an indicator of who the users will be. This last indicator is particularly difficult in that it presupposes the evaluator understands all aspects of a user's behaviour and character that, [as we have seen](#), can be particularly difficult to assess.

14.1.2 Heuristic Evaluation

Related to the walk-through is the heuristic evaluation. This approach differs from the walk-through only in that there are specific aspects that need to be assessed as the evaluation proceeds. These aspects are based upon the general principles [that have already been covered](#). However in this case, as opposed to the developer walking through the different scenarios of user interaction, a set of evaluators are asked to independently answer questions regarding the usability of the interface, rating different aspects as

they go. Once complete, average ratings can be generated for all aspects of the interface based on the consensus opinion of the evaluators. This is a reasonably effective method and is often used as an initial test before the main evaluation begins.

In both cases an expert user, or key participant (informer), is required to perform these evaluations and the results of the evaluation are very much based on the skills, ability, and knowledge of the evaluator. In reality, this means that the principles that you have learnt should be reapplied back into development, but this time as questions. These questions may take the form of ‘Is the need to Facilitate Progressive Disclosure met?’, ‘How is Progressive Disclosure met?’, or the questions to think about when designing your prototype to Facilitate Progressive Disclosure) – such as ‘Is there a tight logical hierarchy of actions?’ – could be used as metrics for understanding the success of failure.

14.2 Qualitative (Fieldwork) Methods

Anthropologists and sociologists describe field evaluation variously as fieldwork, ethnography, case study, qualitative evaluation, interpretative procedures, and field evaluation. However, among anthropologists fieldwork is synonymous [with the collection of data using observational methods](#). For the sociologist, the term often describes the collection of data [using a social survey](#). While it was often thought that these two competing methods of qualitative and quantitative evaluation are disjoint, many sociologists also utilise participant observation, structured interviews, and documentary evidence as interpretive methods. These methods owe much to social anthropologists following the theoretical tradition of ‘interactionism’; interactionists place emphasis on understanding the actions of participants by their active experience of the world and the ways in which their actions arise and are reflected back on the experience. This is useful for the UX specialist as the interactionists component of the method makes it quite suitable investigating subjective aspects.

To support these methods and strategies, many suggest the simultaneous collection and analysis of data. This implies the keeping of substantive field notes consisting of a continuous record of the situations events and conversations in which the practitioner participates [\[Van Maanen, 2011\]](#); along with methodological notes consisting of personal reflections on the activities of the observer as opposed to the observed. The field notes should be preliminary analysed within the field and be indexed and categorised using the [standard method of ‘coding’](#) different sections of the notes to preliminary categories that can be further refined once the fieldwork has concluded.

The most used methods on qualitative evaluation are participant observation, interviewing, archival and unobtrusive methods. I suggest that these methods are mainly used for building a body of evidence that is deep but narrow in extent and scope. The evidence can then be used to help generate hypotheses, and understanding, to be confirmed in later tests. In this case, aspects of Mills Method of Agreement can be used to build evidence before the computational artefact, the application, utility, software, or system is created. Indeed, these methods will give you an idea of what to build or what is wrong with whatever already exists.

14.2.1 Unobtrusive Methods

Imagine you have been employed by the local library to review the number and placement of cataloguing terminals that can be used for book searches by the general public. If these terminals are all placed together large queues form, and besides, members of the public must return from the area of the library, they are in, to the area that houses the centralised terminals if they wish to evaluate the catalogue. To determine how many terminals are required, and in what locations, you may wish to conduct an analysis of floor tile wear, by visual inspection and also by consulting the maintenance records of the library. In this case, you will be able to understand the amount of traffic to each of the different library sections and optimise the placement of terminals along these routes and in these areas. If you decided to use this approach then, you would be using a methodology that is unobtrusive to the participants (in this case library users).

'Unobtrusive methods' is a phrase first coined in 1965/66 and the book in which it was first proposed has since become a classic [[Webb, 1966](#)]. Simply, unobtrusive methods propose that evaluations should look for traces of current activity in a similar way to the way [archival material](#) is used as a past bye-product of normal human activities. This unobtrusive way of investigation is important because in direct experimentation unintended changes can occur, as part of the investigator intervention, which skew the findings ([think 'Bias'](#)).

These sources of invalidity can be roughly categorise as 'reactive measurement effect' or 'errors from the respondent' such as:

- *The guinea pig effect*, whereby people feel like guinea pigs being tested in experiments and so, therefore, change their behaviour patterns;
- *Role selection*, whereby participants see the experimenter as taking a certain role, having an elevated status above the participant, who therefore follows that experimenters lead;
- *Measurement as change agent*, in which aspects of the initial measurement activity introduces real changes in what's being measured; and finally,
- *Response sets*, whereby respondents will more frequently endorse a statement than disagree with its opposite.

Also, errors from the investigator can also be introduced. These range from the interviewer effect, whereby characteristics of the interviewer contributes to the variance in findings because interviewees respond differently to different kinds of interviewer based on the visible and audio cues which that interviewer gives. And changes in the evaluation instrument, whereby the measuring instrument is frequently an interviewer, whose characteristics we have just shown may alter responses, yet that interviewer changes over the course of the investigation. To overcome these possible errors, all contact with participants is removed, and the practitioner bases their findings on observation – Simple Observation – of both the participants and the environment.

Simple observation is the practice of observing exterior physical signs of people as they are going around their normal business along with the expressivity of their movement and their physical location in conjunction. This kind of observation can be extended to include conversation sampling, and time duration sampling for certain observable tasks. Of course, enhanced observational techniques known as contrived observation

may also be undertaken. Here techniques such as hardware instrumentation can be particularly useful for different kinds of computer-based activity as long as ethical considerations are taken into account. In general, unobtrusive methods take a holistic approach of the participant, the task or activity, and the environment. By observing, but not intervening or questioning, the UX specialist can understand the interaction activities and interface issues of individuals enacting a real system in a natural setting without disturbing or affecting that system. We can see how unobtrusive methods can be applied to understanding the user experience from a social perspective in.

14.3 Quantitative & Hybrid Methods

As we have seen, qualitative methods are mainly used for building a body of evidence that is deep but narrow in extent and scope. The evidence can be used to help generate hypotheses, and extend understanding, to be confirmed in later experiments. Simply, hybrid and quantitative methods give the UX specialist the tools and techniques to enact these confirmatory experiments [**Tullis and Albert, 2008**]. Questionnaires, also known as survey methods, are probably the most flexible and useful tools we have for gathering this kind of confirmatory information. They are widely used in the social sciences, as the main form of a systematic method for empirical investigation and critical analysis, to develop and refine a body of knowledge about human social structure and activity.

However, questionnaires have some facets that need careful consideration if the quantitative results produced by their application are to be valid. For instance, questionnaires already make many assumptions regarding the domain under investigation, obviously, the mere activity of asking a specific question has some very implicit assertions associated with it. Therefore, even questionnaires that look to be appropriate may in fact, be biased. Indeed, practitioners have criticised the tradition that has allowed questionnaires to become the methodological sanctuary to which many UX specialists retreat. In this context, the most fertile search for validity comes from a combined series of different measures each with its idiosyncratic weaknesses each pointing to a single hypothesis. In this case, when a hypothesis can survive the confrontation of a series of complementary methods of testing it contains a degree of validity unattainable by one tested within the more constricted framework of the single method. Therefore, practitioners have proposed the hybrid method; also known as mixed methods or triangulation. Here, many complimentary methods are used and indeed this approach is the one I would espouse for most UX work.

Methods that I classify as between field and laboratory are meant to signify quantitative methods used to retest knowledge derived from qualitative investigations and confirm the initial hypothesis selection process. While, quantitative methods are often used as the only method applied to many social science questions, in UX they do not stand up as verifiable when evaluating or testing an interface or human facing system. A more rigorous approach is required in this case in which experimental metrics can be directly applied in a controlled environment.

14.3.1 Card Sorting

There are several well-understood experimental methodologies used for knowledge elicitation. One such methodology is card sorting techniques used along with triadic elicitation techniques to capture the way people compare and order different interfaces based on different criteria. This framework allows the UXer to investigate both qualitative and quantitative aspects of the user experience while recognising that participants are difficult to recruit. By using card sorting methods, you can produce a quantitative analysis with a definite error rate and statistical significance, and by using triadic elicitation, you can also accommodate the most illusive aspects of the user experience and add depth to the quantitative data.

Card sorting is the simplest form of sorting. During this procedure, the participant is given many cards each displaying the name of a concept (or images / wireframes / screenshots, etc.). The participant has the task of repeatedly sorting the cards into piles such that the cards in each pile have something in common. By voicing what each pile has in common, or the difference between each pile, or description of the characteristics of each pile, the participant is vocalising implicit knowledge they have about the things on the cards.

Suppose we wish to find the attributes of a Web page, by which it is judged as simple or complex. Here, the cards are a screen-print of each Web page that was used for testing. With the continuous sorting, the participant is unintentionally giving information on the attributes and values to describe the characteristics of each Web page, describing the reasons for the perceived complexity.

Triadic elicitation is often used along with card sorting techniques. During this technique, the user is asked about what they think is similar and different about three randomly chosen concepts and in what way two of them similar and different. This technique is used to elicit attributes that are not immediately and easily articulated by the user and helps to determine the characteristics of the card sorted concepts. Further, picking three cards forces us into identifying differences between them – there will always be two that are closer together, although which two cards that are may differ

Participants\Sites	S1	S2	S3	S4	S5
P1	7	19	3	17	15
P2	12	2	3	16	11
P3	9	10	1	20	17
P4	6	19	7	17	8
P5	4	19	12	15	8
P6	5	17	4	16	12
P7	14	4	2	13	17
P8	9	20	4	11	13
P9	12	5	2	20	16
P10	11	15	2	19	14
P11	10	9	2	15	14
P12	9	13	3	16	17

Participants\Sites	S1	S2	S3	S4	S5
P1	M3	C7	S4	C2	C4
P2	C5	S2	S3	C6	M4
P3	S7	M5	S1	C5	C1
P4	S5	C2	S2	M11	M4
P5	S3	C2	M1	M11	M6
P6	M1	C5	S4	C4	M3
P7	M8	S6	S3	M7	C2
P8	S6	C9	S4	C2	M2
P9	M4	S8	S2	C6	C3
P10	M2	M7	S3	C3	M4
P11	M1	S9	S2	C1	C2
P12	M4	C1	S3	M8	C4

Figure: Card Sorting Results. Card Sorting Results (Fragment). Sn represents the Site. Pn represents the Participant. The number in the top table represents its position in the card sort concerning visual complexity. The coloured code in the bottom table is defined as (Blue) S=Simple, (Yellow) M=Medium, and (Orange) C=Complex; the number represents its position within the S/M/C grouping. —Image Credit: Michailidou, E. (2005) Metrics of Visual Complexity. Masters Thesis, The University of Manchester.

depending on your perspective. The application is very simple. Basically, you select three cards at random, you then identify which two cards are the most similar. Now analyse what makes them similar and what makes them different.

14.3.2 Socio / Unobtrusive Methods

There are many ways to conduct unobtrusive observations within the user experience domain, these might range from remote observations in the real world – of peoples mobile phone usage, say – to collecting data via proxy methods and based on website usage, etc. UX can be unobtrusively measured, directly or indirectly; individually or collectively. And by having the proper metrics, UxD can be leveraged towards the constant improvement of products and services. And this can, I argue, be replicated and generalised across products and services. Let's have a look at kinds of metrics that can be used.



Figure: Google Analytics PULSE Example. Google Analytics PULSE Example. —Image Credit: Google.

Analytics (PULSE + HEART) [Rodden et al., 2010], Social Sensing or Net Promoter Score Are unobtrusive observational methods that collected and – better still – combined enable us to understand how people feel about a website or desktop application. PULSE – Page views, Uptime, Latency, Seven-day active users (i.e. the number of unique users who used the product at least once in the last week), and Earnings (for instance see [Google Analytics PULSE Example](#) – can be derived from standard quantitative data,

however HEART – Happiness, Engagement, Adoption, Retention, and Task success – requires a little more social networking and user monitoring. In both cases, by understanding the quantity, types, and return rates of users we can infer favourable experiences once we have some social sensing data. My rational here is that analytics provides us with information that is all inferential – people may return to the site not just because they like it but because they have no choice because they want to complain because they found it difficult last time. But if people are also tweeting, Facebook ‘liking’ then you can expect that if this figure is say 20% then over 60% will like the site but can’t be bothered to ‘Like’ it – this is the same with Net Promoter Score¹.

How to capture the user experience in a single session, this is difficult with any degree of accuracy. This could be thought of as the kind of user evaluation method you will become used to. In reality, one session does not make a good evaluation, you should think about the possibility of introducing proxies² to collect longitudinal usage data.

Mindshare goals – qualitative measures such as awareness, branding effectiveness. In general how much chatter is there in the media, around the coffee machine, water cooler, about your application or site. Lots either means love or hate, silence means mediocre. This is mainly a marketing metric, applied with few changes into the UX domain – indeed, there are some obvious similarities between Mindshare and Social Sensing as discussed in ‘Analytics...’.

Customer support responsiveness and Customer satisfaction evaluation. Quantitative and qualitative loyalty. This is a general purpose quantitative and qualitative interview or questionnaire in which consumer satisfaction can be elicited on a wide scale with deployed resources. You normally find this kind of thing in Social Science, and these techniques haven’t changed much in the move to UX. One interesting development is their combination with social metrics such that peer review is provided by giving star ratings to various resources, or as part of ‘Net Promoter’.

Now these methods should interest you (for example see Facebook ‘Gross National Happiness Index’) – not least because their creation, application, and the inferences made from the resultant data tie into user feedback without participant bias. As we’ve previously seen, UX pays more attention to the individual and subjective realm in which ‘intangibles’ are required to become tangible for testing purposes – so that user feedback can be factored into the new design.

¹The Net Promoter Score is obtained by asking customers a question such as ‘How likely is it that you would recommend our company to a friend or colleague?’ rated on a 10 point scale from ‘not’ to ‘very’.

²Such as UsaProxy, which is based upon an HTTP proxy approach. Logging is automatically done on an intermediate computer lying between the web browser and the web servers while multiple users surf the web. The assumption is that all page requests, the browsers make, go through the proxy – <http://fnuked.de/usaproxy/index.htm>.

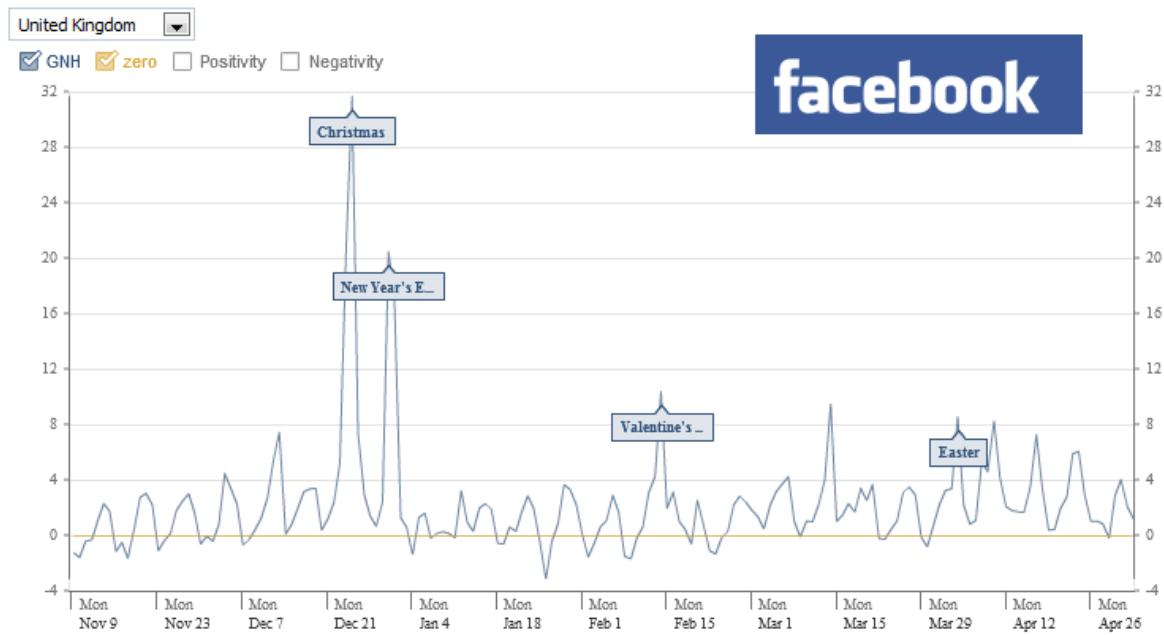


Figure: Facebook ‘Gross National Happiness Index’. Facebook ‘Gross National Happiness Index’. Tracks levels of happiness across different countries. Users rate their happiness as positive or negative. Gross National Happiness is the difference between the positivity and negativity scores. The model is taking into consideration the words used in users’ status updates breaking them out by positive or negative words, for later assess that the day as a whole is counted as happier than usual. Additionally, the model has been calibrated differently to ensure consistency for different countries which eliminates effects due to differences in the countries’ population and language use. —Image Credit: Facebook.

So how do we form these methods into a cohesive framework, well the jury is still out, but Google think it is via Goals, Signals, and Metrics.

Goals: “*The first step is identifying the goals of the product or feature, especially regarding user experience. What tasks do users need to accomplish? What is the redesign trying to achieve?*” Signals: “*Next, think about how success or failure in the goals might manifest itself in user behaviour or attitudes. What actions would indicate the goal had been met? What feelings or perceptions would correlate with success or failure? At this stage, you should consider what your data sources for these signals will be, e.g. for logs-based behavioural signals.*” Metrics: Finally, “*think about how these signals can be translated into specific metrics, suitable for tracking over time on a dashboard*” again playing into longitudinal observation —Google.

14.3.3 A Short Note on Longitudinal Observation

Observation of user behaviour when interacting with applications and the Web – especially for the skill gaining process – is better observed at a longitudinal fashion. This statement is founded on the fact that increased intervals enable the consolidation of declarative knowledge in long-term memory, where consolidation does not happen automatically, and it is not determined at the time it has been learned. To gain insights into the user experience in the context of your development, we should conduct a longitudinal analysis of those users if at all possible.

14.3.4 Think-Aloud

It is sometimes difficult to understand exactly what the user is thinking or, in some cases, doing when they are navigating a complex interface. This is especially the case when the user is familiar with the interface and interaction, and may even be undertaking different, but related, tasks at the same time as the primary task. In this case, to understand explicitly the activities and thoughts of the user, as they are performing the interaction, the think aloud methodology can be used [Lazar et al., 2010].

The think aloud methodology is a classic of the UX evaluation process evolving mainly from design-based approaches (created by Clayton Lewis while at IBM). It produces qualitative data and often occurs as part of an observational process, as opposed to a direct measurement of participant performance, as would be normal in laboratory settings. While it is true that think aloud requires tasks to be completed, the object is not the direct measurement of those tasks. Instead, it is the associated verbalisations of the participants as they progress through the task describing how they are feeling and what they think they need to do.

Think aloud is intended to produce data which is deeper than standard performance measures in that some understanding of the thoughts, feelings, and ideas that are running through the mind of the participant can be captured. The main problem with think aloud is, also its strength in that, it is very easy to set up and run, and therefore, the design aspect of the tasks can be ill conceived. In this way, it is often easy to implicitly influence the participant into providing outcomes that are positive regardless of the true nature of the interface or interaction. Indeed, the very act of verbalising their thoughts and feelings means that participants often change the way they interact with the system. It is for this reason that think aloud should not be used as a methodology on its own but should provide the qualitative aspects lacking in other quantitative or performance-based measures.

14.3.5 Co-Operative Evaluation & Participatory Design

As we have seen³ Co-operative evaluation and participatory design are closely related techniques that enable participants to take some form of ownership within the evaluation and design process. It is often thought that these participants will be in some way key informants, as we have seen in participant observation, and will therefore have an insight into the systems and interfaces that are required by the whole. Both methods are closely linked to the think aloud protocol, but instead of entirely focusing on evaluation the users are encouraged to expand their views with suggestions of improvements based on their knowledge of the system or interfaces that are required. Indeed, the participants are encouraged to criticise the system in an attempt to get to the real requirements. This means that in some cases a system design is created before the participatory or co-operative aspects have begun so that the participants have a starting point.

UCD and cooperative evaluation are related approaches that emphasize the involvement of users in the design and evaluation of interactive systems. While user-centred design focuses on incorporating user feedback throughout the design process, cooperative evaluation specifically addresses the collaborative and iterative evaluation of

³Here touched on for completeness.

a system with users. User-centred design involves understanding user needs, preferences, and behaviours through methods such as user research, interviews, and observations. This user understanding is used to inform the design of interfaces, products, or systems. Throughout the design process, user feedback is collected and integrated, ensuring that the final product meets user expectations and requirements.

Cooperative evaluation, on the other hand, is a specific technique within the user-centred design process. It involves actively involving users as partners in the evaluation and improvement of a system. Rather than solely relying on expert evaluations or usability testing, cooperative evaluation emphasizes the collaboration between designers, developers, and users to identify usability issues, gather feedback, and suggest improvements. In a cooperative evaluation, users are invited to provide input and insights on the system's usability, user experience, and functionality. They may participate in various evaluation activities, such as walk-throughs, user testing, think-aloud protocols, or focus groups. Through this cooperative process, users can contribute their first-hand experiences, identify potential issues or challenges, and provide suggestions for enhancements or refinements.

The relationship between user-centred design and cooperative evaluation lies in their shared commitment to involving users throughout the design and evaluation process. Both approaches recognize the value of user input, perspective, and feedback in creating usable and effective systems. By combining user-centred design with cooperative evaluation, designers can engage users as active collaborators, fostering a user-centric approach that leads to more user-friendly and satisfying experiences.

Cooperative evaluation and participatory design are likewise related approaches that share several similarities in their focus on user involvement and collaboration; including:

1. **User Participation:** Both cooperative evaluation and participatory design emphasize the active involvement of users in the design and evaluation process. Users are seen as valuable contributors who provide insights, perspectives, and expertise that shape the final product or system.
2. **Collaborative Approach:** Both approaches foster collaboration and cooperation between designers, developers, and end-users. They encourage open communication, dialogue, and knowledge sharing among all stakeholders involved in the design process.
3. **User Empowerment:** Cooperative evaluation and participatory design seek to empower users by giving them a voice in decision-making and allowing them to influence the design outcomes. Users are seen as experts in their own experiences and are encouraged to actively participate in shaping the design process.
4. **Iterative and Agile Process:** Both approaches embrace an iterative and agile design process. They involve multiple rounds of feedback, testing, and refinement to ensure that user needs and expectations are addressed. The design is continuously adapted and improved based on the insights and feedback gathered from user participation.
5. **Contextual Understanding:** Both cooperative evaluation and participatory design emphasize the importance of understanding the users' context, needs, and goals. They aim to design solutions that are relevant and meaningful within the specific user context, promoting user satisfaction and usability.

6. User-Centric Design: Both approaches prioritize designing for users' needs and preferences, focusing on creating user-centric solutions. The insights gained through cooperative evaluation and participatory design help ensure that the final product or system aligns with user expectations and requirements.

While there are similarities, it's worth noting that cooperative evaluation is primarily focused on evaluating and refining an existing system, while participatory design is concerned with involving users in the entire design process from ideation to implementation. Nonetheless, both approaches share a user-centred philosophy and highlight the importance of user involvement in the design and evaluation of systems.

The UX specialist must understand that co-operative evaluation, and participatory design are not fast solutions. Indeed, they should only be used when a firm understanding of the boundaries of the system is possessed. Also, participatory design often runs as a focus group based activity and, therefore, active management of this scenario is also required. Enabling each to fully interact within the discussion process while the UX specialist remains outside of the discussion just acting as a facilitator for the participants views and thoughts is a key factor in the process design.

14.3.6 Survey Questionnaires – Reprise

How do you find out if the system or interface that you have designed and deployed is useful and has useful features? What kinds of improvements could be made and in what order should these improvements be prioritised? To answer these kinds of questions, it is useful to talk to a large number of people, far more than you could expect to recruit for a laboratory experiment. In this case, you may decide to use a questionnaire-based survey, recruiting as many users as you possibly can.

Question-based surveys are usually designed to provide statistical descriptions of people and their activities by asking questions of a specific sample and then generalising the results of that survey to a larger population [Bryman, 2008] (for example [Figure: Online Survey Example](#)). This means that the purpose of the survey is to produce statistics and that the main way of collecting information is by asking people questions. In this case, there are three different properties of a good survey, being probability sampling, standardised measurement, and the special-purpose design. Components of a survey sample are based around the question design, the interview method (the questionnaire in this case), and the mode of data collection (verbal or written); all being taken together as total survey design. Critical issues are the choice of how the sample is selected, randomly or non-randomly, creating a probability or non-probability sample; and the sample frame, the size of the sample, the sample design, and the rate of response. One fundamental premise of the survey process is that by describing the sample of people who respond, one can describe the target population. The second fundamental premise of survey evaluation processes is that the answers people give can be used to accurately describe characteristics of the respondent. The sample frame describes the part of the population who have a chance to be selected. Also, if the sample is not random, then the respondents who answer are likely to be different from the target population as a whole. Surveys normally capture two different aspects: objective facts and subjective states. Objective facts include things like the person's height, whereas subjective facts include, how much of the time the persons felt tired, say.

Designing questions to be good measures, which are reliable and provide valuable and valid answers, is an important step in maintaining the validity of a survey. Always avoid inadequate, incomplete, or optional wording while ensuring consistent, meaningful responses. Remove poorly defined terms and avoiding multiple questions conflated to be a single question. However, it is acceptable to include specialised wording for specialist groups. Remember, participants may be tempted to give incorrect responses if they have a lack of knowledge, or change their answers if they find it socially desirable. This should be pre-empted in the designing of the questions, in which questions should be created as reliably as possible.

In addition, there are four different ways in which measurement can be carried out: nominal, people or events are sorted into unordered categories; ordinal, people or events are ordered or placed in all categories along a single dimension; interval, numbers are attached that provide meaningful information regarding the distance between ordered stimuli or classes; and ratio, in which numbers are assigned such that ratios between values are meaningful.

Survey questions should be evaluated before the survey is given using techniques such as focus groups, question drafting sessions, critical reviews, and more formal laboratory interviews. The questions should also be field tested before the main survey becomes available. Remember that survey interviewing can be a difficult job and the type of participant selection is critical in this case. For instance, the commonly used non-probabilistic quota based technique can be particularly troublesome as interviewers are left to survey a certain demographic profile to a certain quota size. This means that many aspects of the validity of a survey are left to the interviewer, who make non-random choices such as choosing houses that are of a higher value, in good areas, without pets or dogs; male interviewers will choose younger female respondents, and female interviewers will choose older male respondents. These biases should be accounted for within the questions and the design of the survey.

Survey methods can be very useful to the UX specialist for confirming qualitative work or evaluating systems that do not immediately lend themselves to the more rigorous laboratory-based methods that will be described in subsequent sections. In the real world, the UX specialist is often unlikely to be able to solicit enough respondents for completely accurate probabilistic methods, and it is more likely that non-probabilistic quota-based methods will be used. However, simple random sampling can be used if

The federal government should raise taxes so it can provide more help for people who need it:

Law-abiding citizens should have more access to guns:

Except in rare instances, such as when a woman's life is threatened, abortion should be illegal:

The United States must do more with its military power to fight terrorism around the world:

A same-sex couple should have access to the same marital benefits as those given to heterosexual couples:

The military should be given a timetable to leave Iraq:

The government should increase funding on embryonic stem cell research:

The death penalty should be available for those convicted of heinous crimes:

Affirmative action is still needed and should be used to promote diversity in the workplace and higher education:

Illegal immigrants who have shown they are productive members of society should be given amnesty and receive social services:

Figure: Online Survey Example. Online Survey Example. – Image Credit: <http://onlinesurveysx.info>.

the sample frame is tightly defined, and in this case readily available ordinal identification, such as employee number, could lend itself to the selection process. While surveys should not be the only method used, they are useful for understanding general points regarding systems and interactions, over a large set of users who could not normally be evaluated in a formal laboratory setting.

14.3.7 Hybrid Methods

The hybrid method; also known as mixed methods or triangulation are terms used to denote the use of many complimentary methods because the UX specialist recognises the inadequacies of a single method standing alone. Indeed, the hallmark of being a field practitioner is flexibility concerning theoretical and substantive problems on hand. Therefore, the use of ‘triangulation’ (a term borrowed from psychology reports) is used to refer to situations where the hypotheses can survive the confrontation of a series of complementary methods of testing. Triangulation can occur as ‘data triangulation’ via time, space, or person; ‘investigator triangulation’ in which more than one person exams the same situation; ‘theory triangulation’ in which alternative or competing theories are used in any one situation; and ‘methodological triangulation’ which involves within method triangulation using the same method used on different occasions, and between-method triangulation when different methods are used in relation to the same object of study. Indeed, mixed methods contrast quantitative and qualitative work, characterising them by behaviour versus meaning; theory and concepts tested in evaluation versus theory and concepts emergent from data; numbers versus words; and artificial versus natural. In reality, for the UX specialist, the confrontational aspects can be thought of as being purely complimentary.

To a large extent, the UX specialist does not need to concern themselves with the methodological debates that are often prevalent within the human sciences such as anthropology, sociology, social science, and psychology. This is mainly because these methodologies and the instruments which are used within them are not directly created as part of the human factors domain but are used and adapted in combination to enable a verifiable, refutable, and replicable evaluation of the technical resource. In UX, a single methodology would not normally ever be enough to support an evaluation or to understand the interaction of technology and user. However, the view I take of the evaluation domain is far more holistic than may be found in most UX or user experience books. By reliance on only the evaluation aspects of a specific technical interface we miss the possibility of understanding how to make that interface better, not just by metrics as shallow as time to task, but by a combined qualitative and quantitative understanding of the factors surrounding user interaction, both cognition and perception, for a particular software artefact or system architecture.

14.4 Tools of the Trade

As a UXer, there are many tools that you can use both in a laboratory-based setting or in the field. Most tools are portable and so, therefore, can be moved around to different sites and venues such that you are more reactive to the locational needs of

your participants; as opposed to expecting them to come to you⁴.

UX tools range from the very simple, such as the notebook, through audio recording devices, portable cameras and video cameras, screen capture and screen recorders, to the more complex (and costly) static and portable eye trackers, bio-feedback system such as galvanic skin response and heart rate monitors, through to neuro-feedback such as functional magnetic resonance imaging (fMRI), electro-encephalo-graphy (EEG — for example, see [Figure: EEG Data Plot Example](#)), event-related potentials (ERPs), and transcranial magnetic stimulation (TMS) systems. All these tools may be mobile, but now often some of the more expensive tools can only be applied in a laboratory setting, and certainly a laboratory-based setting is useful when you wish to control an evaluation; and the possible confounding factors that may apply to that evaluation.



Figure: EEG Data Plot Example. Electro-Encephalo-Graphy (EEG) Spike data Plot Example. —Image Credit: Wikimedia.
A figure showing a vertical stack of 19 EEG channels. Each channel displays a waveform representing brain activity over time. The channels are labeled on the left: O1, F3, F7, F8, F4, C3, C4, C9, C10, P3, P4, P7, P8, T3, T4, T5, T6, T9, T10, and Fz. The x-axis at the bottom shows time in seconds from 14s to 20s. The y-axis represents electrode positions.

These laboratories, known in the industry as ‘user labs’ or ‘usability labs’, often comprised of three rooms. The first room a user would enter is the reception room where there may be coffee tea and comfy sofas to place the user at ease. There would be a user testing room in which the user and, often, a UXer will sit and conduct the evaluations (this is where the ‘tools’ will be). Finally, there is normally an observation room in which other members of the UX team will observe the evaluations in progress. In some cases, only the user will be present in the user testing room, and only the UX specialists will be present in the observation room⁵ (see [Figure: User Observation](#)).



Figure: User Observation. User Observation. —Image Credit: Wikimedia.

⁴Interestingly this expectation was ripe in the 19th and early 20th century with anthropologist’s who conduct their work from the veranda of their bungalows in faraway countries, expecting their ‘subjects’ to come to them as opposed to them embedding with the population that they were studying; this led to the derogatory term ‘veranda anthropology’ that produced many incorrect, badly interpreted, florid, and in some cases sanitised results.

⁵The observation room is connected to the user room normally via one-way mirrors (of the type you’d probably see in popular crime dramas in which an observation room is linked to an interrogation room via a one-way mirror), or in some cases a closed-circuit television is used such that all angles can be covered, and all interaction observed.

As we have seen, there are many techniques in the UX specialists arsenal for investigating user behaviour, however, four of the most common listed below:

Performance Measures. Measuring performance is one of the most used techniques for assessing and evaluating interaction. The rationale is that if the task is completed faster than it was before the interactive component was either altered or created when the Interface design must be better as an enhancement has occurred. Common performance measures include: the time required by the user to complete a task; the time spent navigating the interface; the number of incorrect choices or errors created; the number of jobs completed, either correctly or incorrectly; the number of observations of user frustration (see facial expressions below); and finally the frequency of interface components or behaviour that is never used. While performance measures are the most used and most easy to describe to non-specialist audiences, there are some problems that can be introduced at the time the study is created. Indeed, it is often very easy to introduce bias into a set of tasks such that the desired outcome will always be the outcome that performs best. As a UX specialist, you must be especially careful when designing your studies to make sure that this is not the case.

Eye Tracking. Eye tracking technologies are now increasingly used in studies that analyse the user behaviour in a Web search or to reveal possible usability and accessibility problems. Simply, while reading, looking at a scene or searching for a component, the eye does not move smoothly over the visual field, but it makes continuous movements called saccades and between the saccades, our eyes remain relatively still during fixations for about 200-300 ms. A sequence of saccades provides the scanpath (for example see [Figure: Eye-Tracking Gaze Plot Example](#)) that the eye follows while looking. Fixations follow the saccades and are the periods that the eye is relatively immobile indicating where it pays more attention, hence, the component that is viewed. Mostly used for usability evaluations we can see their application in determining specific

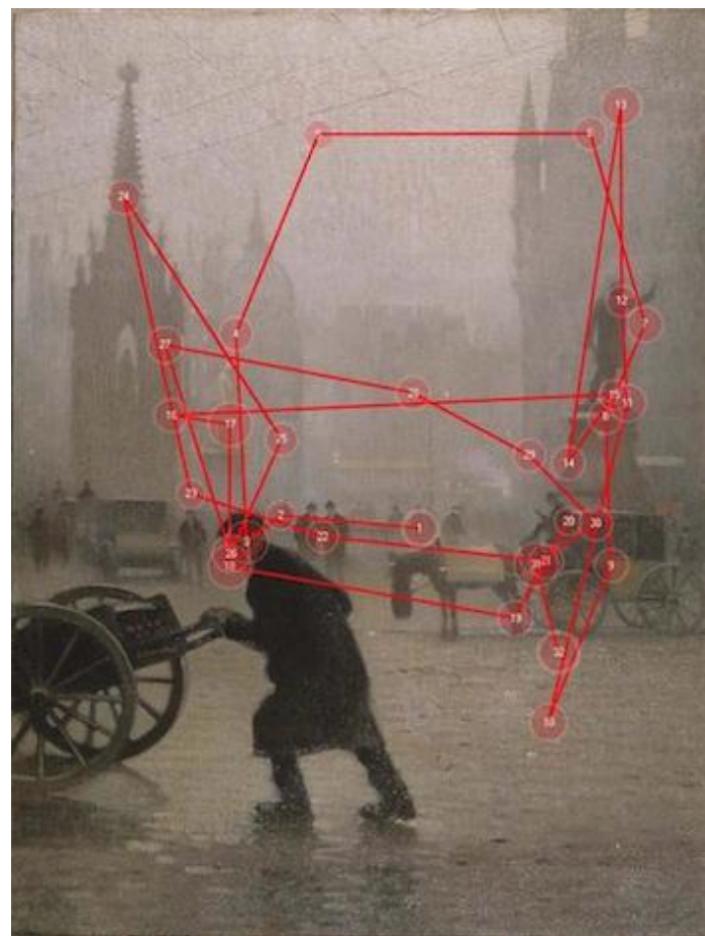


Figure: Eye-Tracking Gaze Plot Example. Eye-Tracking Gaze Plot Example. –Image Credit: UoM.

scanpaths about each interface component is highly useful. If each design is associated with a scanpath and fixation points, feedback can be provided for enhancing the design.

However, as technology has continued to evolve, applications where an understanding of human perception, attention, search, tracking and decision making are becoming increasingly important. This is because eye movements are driven both by properties of the visual world and processes in a person's mind. Indeed, tracking eye movements has now become a valuable way of understanding how people allocate their visual attention.

Facial Expression. There are many implicit cues in user behaviour which are difficult to measure by conventional means such as eye tracking or user performance. One way of capturing some of these implicit aspects is by understanding that most users will show these implicit behaviours, such as happiness or frustration, by their facial expressions. Techniques, therefore, exist in which the expression of the user is recorded via a standard computer camera, where each task is timed, and the time of the facial expression is then matched to the task being undertaken at that time. In this way, the specialist can collect a wealth of implicit information concerning the quality of the user experience, if not the participants performance. Again, the UX specialist should be careful to make sure their study is designed correctly and that the analysis of the facial expressions captured in the evaluation is as accurate as possible. Indeed, as we have said before it may be useful to present these expressions, for categorisation, to a disinterested party as a confirmatory step; remember, this is the best way of enhancing the validity of the evaluation when interpretation by the evaluator is required.

Biofeedback and Affective Measures [Picard, 1997].

As with facial expressions, biofeedback is an implicit evaluation process that involves measuring a participant's quantifiable bodily functions such as blood pressure, heart rate, skin temperature, sweat gland activity, and muscle tension, recording the information for later analysis. Within the UX domain the most often used biofeedback measurement is Galvanic Skin Response (see [Figure: Galvanic Skin Response](#)) which is a measure of the electrical resistance of the skin; this being a good indicator of the participant's stress levels. Also, more simplistic by a feedback mechanism such as heart rate and skin temperature can be used in a non-invasive manner to also ascertain the levels of comfort, excitement, or the stress of a participant. Most biofeedback measurements must be analysed in the context of the individual user in a relative format, so, therefore, increases from the baseline recorded when the user is relaxed and under normal conditions are more important than absolute measurements. One final thing to note for the UX specialist is that undisputed scientific evaluation into the possible application of biofeedback is lacking. This is not such a problem for evaluation and analysis but does indicate the immaturity of this kind of technique.

As a UX specialist you are being asked to perform these kinds of experiments and evaluations for some perceived gain. By conforming as closely as possible to the scientific principles of impartiality the evaluation methodologies and preterite tools, you will be able to maximise these gains, and exhibit a high degree of professionalism



Figure: Galvanic Skin Response. Galvanic Skin Response. – Image Credit: Wikimedia.

in what is often a practical engineering setting. Indeed, [as we shall see](#), pushing a bad interaction design to market will only necessitate a more costly redesign at a later date.

Caveat – Experimental Methods

You may have noticed that I've not mentioned any tightly controlled task based trials that measure performance directly - and mostly in laboratory-based settings. These kinds of tests are normally used in research and validation of human performance for critical systems or in 'hard-core' usability / HCI trials. I've steered away from these because in everyday UX you won't need to use them, and because we only have a limited time in which to cover UX, and these are not – in my option – primary to this domain; but rather human factors, ergonomics, cognitive science, and experimental psychology.

Laboratory-based evaluation using experimental methods has been mainly adopted within the human sciences by experimental or cognitive psychologists requiring similar empirical confirmations as their natural science counterparts. In this case, it is seen that the rigorous and formalised testing of participants can only occur in a controlled laboratory setting. While this is the major strength of laboratory-based evaluation, it is also acknowledged to be a possible problem in that the laboratory is not a naturalistic setting. In turn, the negative aspects are accentuated even beyond that of the survey questionnaire. However, in some cases, the UX specialist has little choice in performing laboratory experimentation because the quantifiable richness and rigour of the data produced is not available from any other source. The power of the arguments created from experimental work is often too strong to ignore, and this is why you will find that when only one opportunity for evaluation exists, the UX specialist will most naturally choose a laboratory-based experimental method; in some ways returning to their computer science / engineering roots.

There are some increasingly understood aspects of laboratory-based work that may be problematic:

- **Experimenter Bias:** The experimenter may bias the laboratory work via aspects such as the guinea pig effect or role bias;
- **Obtrusive Observation:** The act of observation changes the observed in some way.;
- **Not Longitudinal:** And so do not represent users changing states over time and experience; and therefore,
- **Not Ecologically Valid:** In that the results are only valid in the laboratory and not in the real world.

Further definitive aspects of laboratory-based work are the emphasis placed upon control and validity. Aspects of both can be seen at various points throughout both quantitative and qualitative methods however the focus is far more acute in laboratory-based experimental evaluation or evaluation. This means that various methods for designing and controlling laboratory-based experimental work have evolved both in psychology and in medicine concerning clinical trials, and we have already covered this to some extent. The key aspect of laboratory-based evaluation is the concept of internal

and external validity. External validity refers to the degree to which we can generalise the results of the study to other subjects, conditions, times, and places. While internal validity, is specifically focused on the validity of the experiment as it is carried out and the results that derive from the sample. Remember these terms [as we'll be looking at them in more detail](#), but for now, if you'd like more information on more experimental methods take an initial look at Graziano and Raulin [[Graziano and Raulin, 2010](#)].

14.5 Summary

In summary then, we can see that evaluation methodologies range from the very deep qualitative work undertaken by anthropologists, often resulting in an ethnography, through the broad quantitative work undertaken by social scientists, to the observational empirical work of the experimental or cognitive psychologist. Into this mix comes the interdisciplinary aspects of user experience based in software evaluation and design, and in the form of walk-throughs and think aloud protocols. In all cases, there is a need for a combinatorial approach to evaluation design if an accurate view of the user, and their interaction requirements and experiences, are to be formulated. The value of these aspects of the entire software design cannot be underestimated, without them the user experience cannot be assessed, and a bad user experience will directly affect the approval, and, therefore, sales, of the product under investigation. It's also not all about participants and numbers, remember, [Facebook did UX testing For Facebook Home \(With Fewer Than 60 People\)](#)⁶, ⁷. However, UX is not solely focused on the interface. Indeed, aspects of the interaction enable us to formulate a scientific perspective and enables us to understand more about the behaviour, cognition, and perception of the user, as opposed to purely focusing on changes to the interface; in this way, UX evaluation methodologies have both practical and scientific outcomes. While I would not suggest that the evaluation methodologies discussed here [can be applied in every setting](#), the UX specialist should attempt to create evaluations that can be undertaken in as near perfect conditions as possible.

So what does all this mean, well ‘methods maketh the discipline’, and I’d say that UX has some nice native methods in use with some others pulled in from other more traditional product marketing domains spliced up with advertising metrics. Importantly, the most interesting for me are HEART+PULSE, which together represent some very innovative thinking that – with minor mods – can be directly applied from UX back to the wider Human Factors CS domain.

14.5.1 Optional Further Reading

- [M. Agar.] The professional stranger: an informal introduction to ethnography. Academic Press, San Diego, 2nd ed edition, 1996.
- [A. Bryman.] Social research methods. Oxford University Press, Oxford, 3rd ed edition, 2008.
- [A. M. Graziano] and M. L. Raulin. Research methods: a process of inquiry. Allyn and Bacon, Boston, 7th ed edition, 2010.

⁶<http://www.fastcolabs.com/3008397/open-company/how-facebook-did-ux-testing-facebook-home-fewer-60-people>
⁷Thanks for pointing to this Josh Nolan.

- [J. Lazar,] J. H. Feng, and H. Hochheiser. Research methods in human-computer interaction. Wiley, Chichester, West Sussex, U.K., 2010.
- [J. Van Maanen.] Tales of the field: on writing ethnography. Chicago guides to writing, editing, and publishing. University of Chicago Press, Chicago, 2nd ed edition, 2011.
- [R. W. Picard.] Affective computing. MIT Press, Cambridge, Mass., 1997.
- [T. Tullis] and B. Albert. Measuring the user experience: collecting, analyzing, and presenting usability metrics. The Morgan Kaufmann interactive technologies series. Elsevier/Morgan Kaufmann, Amsterdam, 2008.



Self Assessment Questions

Try these without reference to the text:

1. What are qualitative methods and how do they differ from quantitative ones?
2. What are the key problems with laboratory-based work?
3. What problems may exist when undertaking single method evaluation?
4. Why is co-operative evaluation different from other methods?
5. What tools are at the disposal of the 'poor' UXer?

15. Human-in-the-Loop Systems and Digital Phenotyping

Systems simulations were a mix of hardware and digital simulations of every—and all aspects of—an Apollo mission which included man-in-the-loop simulations, making sure that a complete mission from start to finish would behave exactly as expected.

— Margaret H. Hamilton (1965) NASA.

In the bad old days, computer systems were highly inefficient took huge amounts of resources and were not user friendly. Indeed, the head of IBM through the 50s said that he could only see the need for two computers in the world. Obviously, that was incorrect and as time has progressed, users became more important and we became more aware of the fact that humans were an integral part of the system. And so human computer interaction and therefore user experience were created. As part of this, the idea that humans in the loop would be a necessary part of a computer system was not considered until the various Apollo missions whereby human interaction with the system, indeed control of the system became important. Indeed, we may think of it as human middleware became more important, especially with computer systems flying or at least controlling complex mechanical ones.

There is an accessibility saying: ‘nothing about us without us’ and we consider that this should be extended to the human and to the user such that users are integrated into most aspects of the build lifecycle. After all, the human will be controlling the system and even in terms of intelligence systems, artificial/hybrid intelligence and machine learning, have shown to benefit from human input.

Indeed, humanistic artificial intelligence are becoming increasingly more important with most large scale computational organisations. Acknowledging this fact and having large departments which are tailored to this kind of development so for all software engineering and development the human should be considered From the outset and the humans, control actions should also be factors when designing systems which wrap around the user

15.1 Human-in-the-Loop (HITL) Systems

Human-in-the-Loop (HITL) systems are a collaborative approach that combines the capabilities of both humans and machines in a loop or iterative process. It involves the interaction and collaboration between human experts or operators and automated systems or algorithms to achieve a desired outcome. In a HITL system, humans are actively involved in various stages of the decision-making process, providing input, feedback, and guidance to the automated systems. The automated systems, on the other hand, assist humans by performing tasks that can be automated, analyzing large amounts of data, or making predictions based on complex algorithms.

The purpose of HITL systems is to leverage the strengths of both humans and machines. Humans bring their domain expertise, intuition, and contextual understanding, while machines offer computational power, speed, and the ability to process vast amounts of data. By combining these capabilities, HITL systems aim to improve accuracy, efficiency, and decision-making in various domains, such as healthcare, customer service, autonomous vehicles, and cybersecurity.

HITL systems often involve an iterative process, where humans provide initial input or guidance, machines generate outputs or suggestions, and humans review, validate, or modify those outputs. This iterative feedback loop allows for continuous improvement and adaptation, with humans refining the system's performance and the system enhancing human capabilities. Overall, HITL systems enable the development of more robust, reliable, and trustworthy solutions by harnessing the power of human intelligence and machine capabilities in a symbiotic relationship.

HITL systems have been utilized for a long time, although the term itself may have gained prominence in recent years. The concept of involving humans in decision-making processes alongside automated systems has been present in various fields and industries for decades. One early example of HITL systems is found in aviation. Pilots have been working in collaboration with autopilot systems for many years, where they oversee and intervene when necessary, ensuring the safety and efficiency of flight operations. This demonstrates the integration of human expertise with automated systems.

NASA has embraced the concept of HITL systems across various aspects of its operations, including space exploration, mission control, and scientific research. Here are a few examples of how NASA has adopted HITL approaches. Human space exploration missions, such as those to the International Space Station (ISS) and beyond, heavily rely on HITL systems. Astronauts play a critical role in decision-making, performing experiments, and conducting repairs or maintenance tasks during their missions. While automation is present, human presence and decision-making capabilities are essential for handling unforeseen situations and ensuring mission success.

NASA's mission control centres, such as the Johnson Space Center's Mission Control Center in Houston, Texas, employ HITL systems to monitor and manage space missions. Teams of experts, including flight directors, engineers, and scientists, collaborate with astronauts to provide real-time support, make critical decisions, and troubleshoot issues during missions. NASA utilizes robotic systems in space exploration, such as the Mars rovers (e.g., Spirit, Opportunity, Curiosity, and Perseverance). While these robots operate autonomously to some extent, human operators on Earth are actively involved in planning, commanding, and interpreting the data collected by the rovers. Humans in mission control provide guidance, analyse results, and adjust mission objectives based on discoveries made by the robotic systems.



HITL & NASA

HITL systems were instigated (to a large extent) by Margaret Hamilton, an American computer scientist and systems engineer. Her pioneering efforts in software engineering and the development of onboard flight software had a profound impact on the integration of humans and machines in space missions. She recognized the criticality of software reliability in human spaceflight. Leading the team that developed the onboard flight software for the Apollo missions, including the Apollo Guidance Computer (AGC). Hamilton championed rigorous software engineering practices, introducing concepts like error detection and recovery, fault tolerance, and prioritization of critical tasks. Her emphasis on software quality and reliability helped ensure the safety of astronauts by minimizing errors and enabling human intervention when necessary. And was an advocate for a human-centred approach to software design and development. She emphasized the need for software systems to be adaptable and responsive to human operators' needs and capabilities. By considering human factors and usability, Hamilton aimed to create software interfaces that facilitated effective HITL interactions, enabling astronauts to understand and control the spacecraft's behaviour.

Hamilton's team developed innovative error detection and recovery mechanisms in the Apollo onboard flight software. This included the implementation of prioritized alarms, which alerted astronauts to critical issues and provided actionable information for corrective actions. By involving humans in the error detection and recovery process, Hamilton ensured that astronauts had the necessary information and control to respond to anomalies and make critical decisions. Recognizing the importance of both human supervision and autonomous decision-making in space missions. She designed the software to allow astronauts to override automated systems when necessary, maintaining a HITL approach. At the same time, she empowered the software to autonomously perform routine or less critical tasks, reducing the cognitive load on astronauts and enabling them to focus on high-level decision-making.

Hamilton's contributions to software engineering and her focus on HITL systems laid the foundation for safe and reliable human spaceflight. Her work not only influenced NASA's approach to software development but also shaped the broader field of human-computer interaction, emphasizing the need for human-centred design and the integration of humans and machines in complex systems.



Figure: Hamilton instigated HITL work at NASA Margaret Hamilton stands next to a stack of program listings from the Apollo Guidance Computer in a photograph taken in 1969. — Image Credit: Wikimedia Commons.

HITL systems are prevalent in data analysis and scientific research conducted by NASA. Scientists and researchers work alongside machine learning algorithms and data processing systems to analyse large volumes of space-related data, such as satellite imagery, telescope observations, and planetary data. Human expertise is crucial for interpreting results, identifying patterns, and making scientific discoveries.

Overall, HITL approaches are integrated into various aspects of NASA's operations, where human expertise is combined with automated systems to achieve mission objectives, ensure astronaut safety, and advance scientific knowledge in space exploration.

In the field of computer science and artificial intelligence, the idea of HITL systems has been explored since the early days of AI research. In the 1950's and 1960's, researchers were already investigating human-computer interaction and the combination of human intelligence with machine processing power. More recently, with advancements in machine learning, data analytics, and robotics, HITL systems have gained increased attention. They have been applied in various domains such as healthcare, where clinicians work alongside diagnostic algorithms to improve disease detection, treatment planning, and patient care. Additionally, HITL systems have become essential in the development and training of AI models. Human involvement is crucial for labelling and annotating training data, evaluating model performance, and ensuring ethical considerations are taken into account. While the exact introduction of HITL systems cannot be pinpointed to a specific date or event, their evolution and adoption have been shaped by the continuous advancements in technology and the recognition of the value of human expertise in conjunction with automated systems.

15.2 Digital Phenotyping

Digital Phenotyping (DP) can be seen as an extension (or at least has a very strong relationship to) HITL systems, in that the human in DP systems is carrying around a mobile device / wearable / or the like and their behaviour in both the real-world and in digital services. This behaviour is monitored and the collected data is used to make inferences about their behaviour.

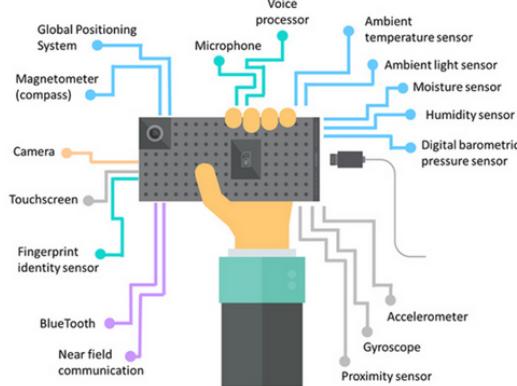


Figure: Digital Sensors. Digital Sensing.

breathing rate, and temperature, as well as factors like location, movement, and in-

Defined by Jukka-Pekka Onnela in 2015, but undertaken for over a decade before it was named, DP utilises digital technologies such as smartphones, wearable devices, and social media platforms to collect and analyse data on human behaviour and psychological states. This approach is used to monitor, measure, and analyse various aspects of human behaviour, including sleep patterns, physical activity, social interaction, and emotional states. And uses sensors embedded in smartphones and wearables to track various physiological and environmental parameters, such as heart rate,

teraction with the device. The data collected is then analysed using machine learning algorithms to generate insights into an individual's behaviour, including patterns of activity, stress levels, mental health, and well-being, and sleep quality. Roughly DP proceeds by:

1. Data Collection: Digital phenotyping relies on the collection of data from individuals using their smartphones, wearables, or other digital devices. This data can include GPS location, accelerometer and gyroscope readings, screen interaction patterns, call and text logs, app usage statistics, social media posts, and more. Sensors within the devices capture data related to movement, activity, and contextual information.
2. Data Processing and Feature Extraction: Once the data is collected, it undergoes processing and feature extraction. This involves converting the raw data into meaningful features or variables that represent specific behavioural or physiological aspects. For example, data from accelerometer readings can be transformed into activity levels or sleep quality indicators.
3. Machine Learning and Pattern Recognition: Digital phenotyping employs machine learning and pattern recognition techniques to analyse the extracted features and identify patterns, trends, or anomalies. Algorithms are trained on labelled data to recognize specific behavioural patterns or indicators related to mental health, cognitive function, or physical well-being.
4. Behaviour Modelling and Prediction: By analysing the collected data and applying machine learning models, digital phenotyping can develop behavioural models and predictive algorithms. These models can identify patterns and correlations between digital data and specific outcomes, such as predicting depressive episodes, detecting stress levels, or assessing cognitive performance.
5. Continuous Monitoring and Feedback: Digital phenotyping allows for continuous monitoring and tracking of individuals' behaviours and mental states over time. The data collected can provide real-time insights into changes in behaviour or well-being, enabling early intervention or personalized feedback.
6. Integration with Clinical and Research Applications: The insights generated through digital phenotyping can be integrated into clinical settings or research studies. Mental health professionals can use the data to inform treatment decisions, monitor patient progress, or identify potential relapses. Researchers can leverage the data for population-level studies, understanding disease patterns, or evaluating the efficacy of interventions.

It's important to note that digital phenotyping raises ethical concerns regarding privacy, data security, and informed consent. Safeguards must be in place to protect individuals' privacy and ensure transparent and responsible use of the collected data.

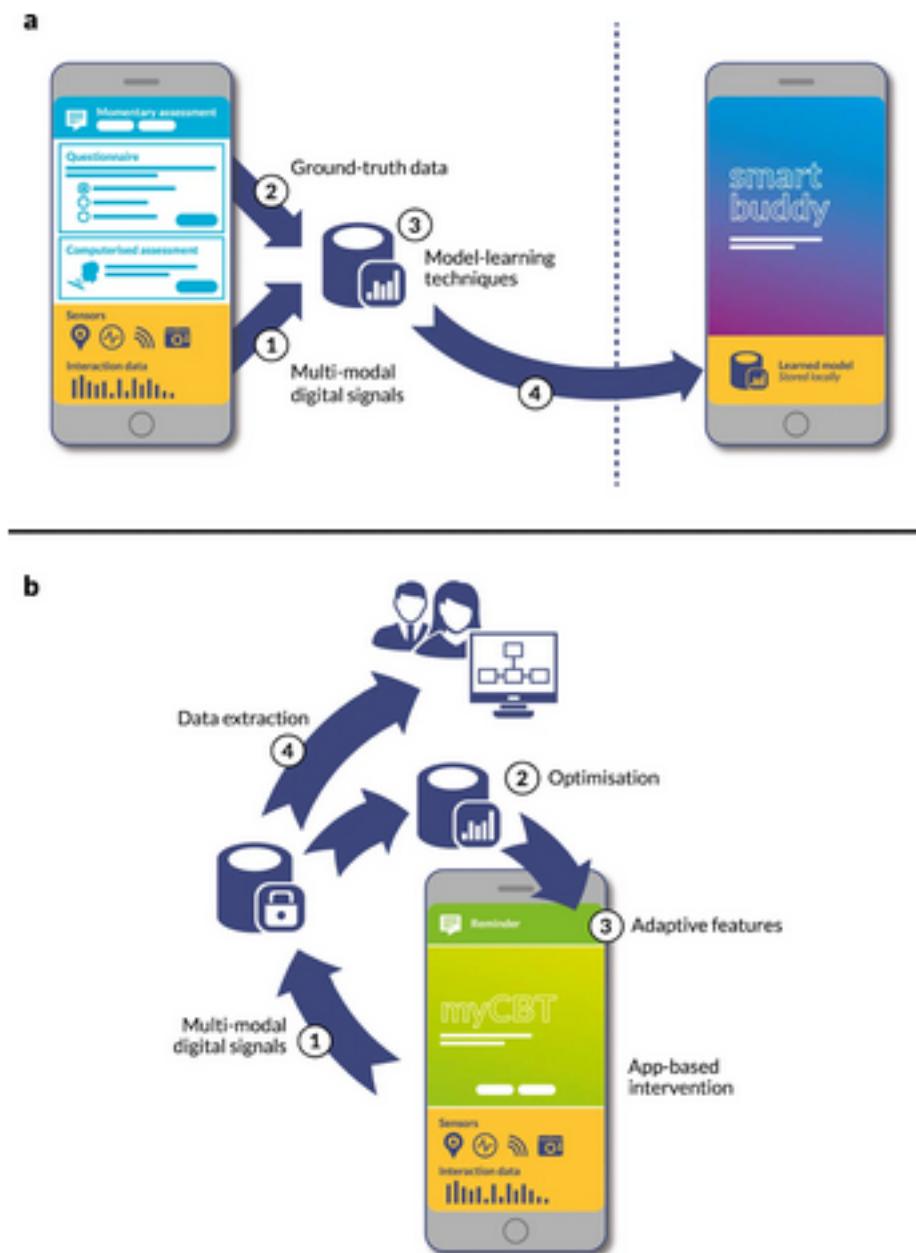


Figure: How Digital Phenotyping Works. How Digital Phenotyping Works. —Image Credit: Huckvale, K., Venkatesh, S. & Christensen, H. <https://doi.org/10.1038/s41746-019-0166-1>

DP as a term is evolved from the term ‘genotype’ which is the part of the genetic makeup of a cell, and therefore of any individual, which determines one of its characteristics (phenotype). While phenotype is the term used in genetics for the composite observable characteristics or traits of an organism. Behavioural phenotypes include cognitive, personality, and behavioural patterns. And the interaction of interactions with the environment. Therefore digital phenotyping implies monitoring; a moment-by-moment quantification of the individual-level human phenotype in situ using data from personal digital devices in particular smartphones.

The data can be divided into two subgroups, called active data and passive data, where the former refers to data that requires active input from the users to be generated,

whereas passive data, such as sensor data and phone usage patterns, are collected without requiring any active participation from the user. We might also equate Passive Monitoring / Sensing to Unobtrusive Monitoring and Active Monitoring / Sensing to Obtrusive Monitoring.

DP conventionally has small participant numbers however the data collected is the key and when you're collecting ~750,000 records/person/day adaption and personalisation to a user is critical! We can think of Data is the 'Population' where small N becomes big N, in this case, DP supports personalised data analysis. Generalized Estimating Equations and Generalized Linear Mixed Models create a population-based model instead of a personalised one however, personal models are however needed in many complex applications. Machine and Deep Learning algorithms don't prioritise clinical knowledge but rather structure of the data and assume the distribution of the training set is static, but human behaviour is not and often requires tuning, this requires expertise which in practice would not be available for each patient.

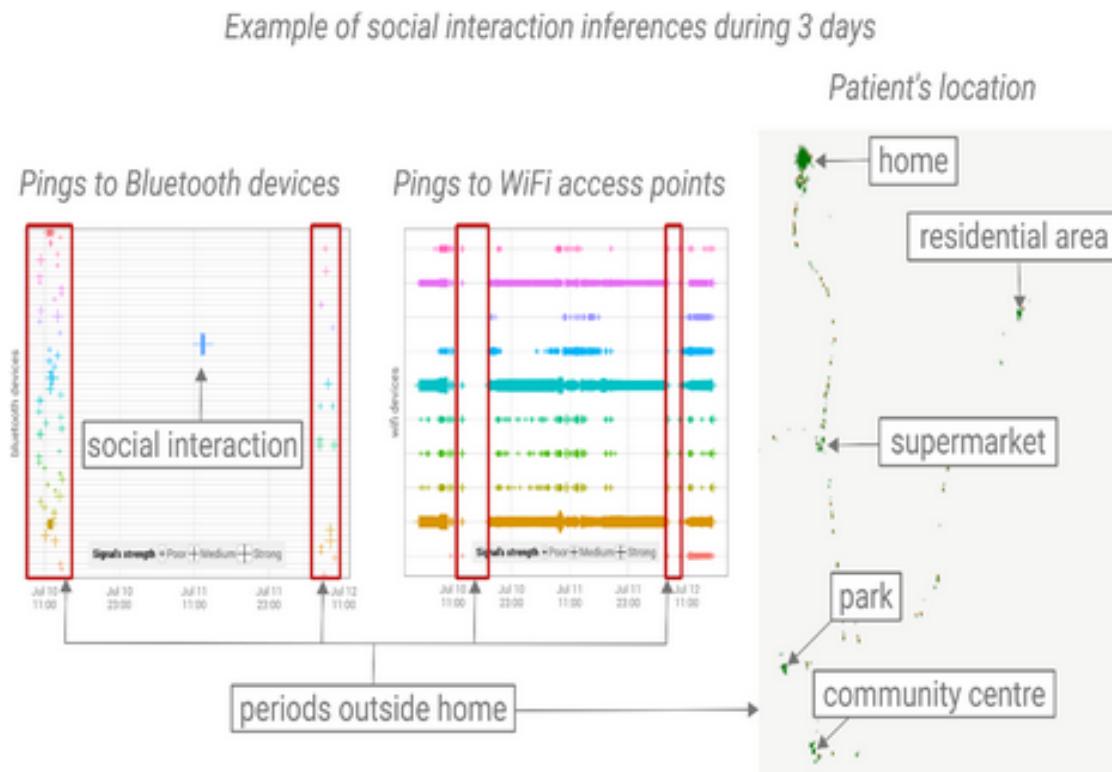


Figure: Just what can be collected? Understanding Parkinson's Disease. —Image Credit: Vega & Harper

Digital phenotyping has many potential applications in healthcare and mental health research. For example, it could be used to monitor and manage chronic conditions such as diabetes or to identify early signs of mental health issues such as depression or anxiety. It could also be used to provide personalized interventions and support for individuals based on their unique patterns of behaviour and psychological state. However, DP has diverse application domains across mental health, and behaviour

monitoring among others:

1. Mental Health Monitoring: Digital phenotyping enables the continuous monitoring of individuals' mental health by analysing digital data such as smartphone usage, social media activity, and communication patterns. It can help detect early signs of mental health disorders, track symptoms, and monitor treatment progress.
2. Mood and Stress Management: Digital phenotyping can assess and track individuals' mood states and stress levels by analysing behavioural patterns, including activity levels, sleep quality, communication patterns, and social media content. It can provide insights into triggers and patterns related to mood changes and help individuals manage stress more effectively.
3. Cognitive Function Assessment: By analysing smartphone usage patterns, digital phenotyping can provide insights into individuals' cognitive function, attention, and memory. It can help assess cognitive impairments, monitor cognitive changes over time, and provide personalized interventions or reminders.
4. Physical Health Monitoring: Digital phenotyping can be used to monitor individuals' physical health by analysing data from wearable devices, such as heart rate, sleep patterns, and activity levels. It can track physical activity, sleep quality, and identify deviations that may indicate health issues or support healthy behaviour change.
5. Personalized Interventions and Treatment: Digital phenotyping insights can be used to deliver personalized interventions, recommendations, or treatment plans. By understanding individuals' behaviours, triggers, and context, interventions can be tailored to their specific needs and preferences, enhancing treatment outcomes.
6. Population Health Studies: Digital phenotyping can be applied in large-scale population health studies to understand disease patterns, identify risk factors, and evaluate the effectiveness of interventions. By analysing aggregated and anonymized digital data, researchers can gain insights into population-level health trends and inform public health strategies.
7. Behaviour Change and Wellness Promotion: Digital phenotyping can support behaviour change interventions by providing real-time feedback, personalized recommendations, and tracking progress towards health and wellness goals. It can motivate individuals to adopt healthier behaviours and sustain positive changes.

As technology advances and our understanding of human behaviour and health improves, digital phenotyping is likely to find further applications in various fields related to well-being, healthcare, and personalized interventions.

15.2.1 Can you build a Digital Twin using Digital Phenotyping?

While Digital twins and digital phenotyping are related concepts in the realm of technology and data-driven analysis, but they focus on different aspects and applications.

Digital twins are virtual replicas of physical objects, systems, or processes. They are used to monitor, analyse, and simulate the behaviour of their real-world counterparts. Digital twins are often associated with industrial applications, such as manufacturing, energy, and healthcare, where they help optimize operations, predict maintenance

needs, and improve efficiency. They involve real-time data integration, simulation, and visualization to provide insights and predictions about the physical entity they represent. Digital twins are more about replicating the entire behaviour of a physical entity in a digital environment. On the other-hand, Digital phenotyping focuses on the collection and analysis of data related to an individual's behaviour, activities, and physiological responses using digital devices. It is often used in healthcare and psychology to monitor mental and physical health, track disease progression, and understand behavioural patterns. Digital phenotyping involves the use of smartphones, wearables, and other digital sensors to gather data like movement, sleep patterns, communication style, and more. The goal of digital phenotyping is to gain insights into an individual's health and well-being by analysing patterns and changes in their digital behaviour.

This said, it is possible to incorporate digital phenotyping techniques into the development and enhancement of a digital twin, which are in many ways complimentary. By integrating data collected through digital phenotyping methods, you can create a more accurate and comprehensive representation of the physical entity within the digital twin environment.

Using digital devices such as smartphones, wearables, and other sensors to collect behavioural and physiological data from the physical entity you want to model in your digital twin. This could include data on movement, activity levels, sleep patterns, heart rate, communication patterns, and more. Then integrate the digital phenotyping data with the data streams from other sources that contribute to your digital twin's functionality. For instance, if you're creating a digital twin of a human body for healthcare purposes, you might combine phenotypic data with medical records, genetic information, and environmental data. Incorporate the collected data into the digital twin's model. Depending on the complexity of your digital twin, this could involve refining the simulation algorithms to better mimic the behaviour of the physical entity based on the behavioural and physiological data you've collected. Analyse the digital phenotyping data to identify patterns, trends, and anomalies in the behaviour of the physical entity. This analysis can inform the simulation algorithms and contribute to a more accurate representation within the digital twin. Use the integrated data to make predictions and gain insights. For example, if your digital twin represents a person's health, you could predict potential health issues based on changes in behavioural patterns and physiological data. Continuously update the digital twin with new data from digital phenotyping to ensure that the virtual representation stays aligned with the real-world counterpart. This real-time monitoring can help detect changes and provide early warnings for potential issues. Implement a feedback loop where insights and predictions generated by the digital twin can influence how data is collected through digital phenotyping. This can help optimize the data collection process and improve the accuracy of both the digital twin and the phenotyping analysis.

By combining digital phenotyping with the concept of a digital twin, you can create a more dynamic and accurate representation of a physical entity, enabling better insights, predictions, and decision-making in various fields such as healthcare, sports science, and more. You can [skip backwards to understand how digital twins relate to requirements elicitation](#) if you missed this.

15.3 Summary

Digital phenotyping and HITL systems are related concepts that can complement each other in various ways. Digital phenotyping relies on the collection and analysis of digital data from various sources such as smartphones, wearables, and sensors. HITL systems can play a role in ensuring the accuracy and reliability of the collected data. Humans can validate and verify the collected data, identify errors or inconsistencies, and provide feedback to improve the quality of the data used in digital phenotyping algorithms. HITL systems can contribute to the development and validation of digital phenotyping algorithms. Human experts, such as clinicians or researchers, can provide their expertise and domain knowledge to guide the development of algorithms that capture meaningful behavioural or health-related patterns. Humans can also participate in the evaluation and validation of the algorithms, providing insights and judgments to assess their performance and effectiveness.

Digital phenotyping algorithms generate insights and predictions based on the analysis of collected data. HITL systems can assist in the interpretation and contextualization of these results. Human experts can provide a deeper understanding of the implications of the findings, identify potential confounders or biases, and help translate the results into actionable information for healthcare providers, researchers, or individuals. HITL systems enable continuous feedback loops for digital phenotyping. Users or individuals can provide feedback on the insights or predictions generated by digital phenotyping algorithms. This feedback can help refine and improve the algorithms over time, ensuring that the system becomes more accurate, sensitive, and tailored to individual needs. Further, HITL systems play a crucial role in addressing ethical considerations related to digital phenotyping. Humans can ensure that privacy concerns, data security, informed consent, and fairness are appropriately addressed. Human judgment and decision-making can guide the responsible and ethical use of digital phenotyping technologies.

Overall, HITL systems can enhance the reliability, interpretability, and ethical considerations of digital phenotyping. By involving humans in the data collection, algorithm development, interpretation of results, and feedback loops, we can create more robust and responsible digital phenotyping systems that align with user needs, expert knowledge, and ethical standards.

15.3.1 Optional Further Reading

- [Mackay, R. S. (1968).] – Biomedical telemetry. Sensing and transmitting biological information from animals and man.
- [Webb, E. J., Campbell, D. T., Schwartz, R. D., & Sechrest, L. (1966)] – Unobtrusive measures: Nonreactive research in the social sciences (Vol. 111). Chicago: Rand McNally.
- [Licklider, J. C. R. (1960)] Man-Computer Symbiosis, IRE Transactions on Human Factors in Electronics, volume HFE-1, pages 4-11, March 1960.
- [Onnela, Jukka-Pekka; Rauch, Scott L. (June 2016)] – Harnessing Smartphone-Based Digital Phenotyping to Enhance Behavioral and Mental Health. Neuropsychopharmacology. 41 (7): 1691–1696. doi:10.1038/npp.2016.7. ISSN 0893-133X. PMC 4869063. PMID 26818126.

- [Harald Baumeister (Editor), Christian Montag (Editor) (2019)] — Digital Phenotyping and Mobile Sensing: New Developments in Psychoinformatics (Studies in Neuroscience, Psychology and Behavioral Economics) 1st Edition
- [David Nunes, Jorge Sa Silva, Fernando Boavida (2017)] — A Practical Introduction to Human-in-the-Loop Cyber-Physical Systems (IEEE Press) 1st Edition



Self Assessment Questions

Try these without reference to the text:

1. What are Human in the Loop Systems?
2. What is Digital Phenotyping?
3. How do Human in the Loop Systems and Digital Phenotyping relate to each other?
4. Who was the main instigator or HITL systems and why was it thought to be so necessary?
5. When and why should you use Digital Phenotyping?

16. Evaluation Analysis

You can design and create, and build the most wonderful place in the world. But it takes people to make the dream a reality.

– Walt Disney

Designing your evaluations is one of the most important aspects of any user experience process. If these evaluations are designed badly you will not be able to apply the correct analysis, and if you cannot apply the correct analysis you will not be able to make any conclusions as to the applicability or success of your interventions at the interface or interactive level [**Baggini and Fosli, 2010**]. In reality, this means that if this is not done correctly the previous ≈200 pages of this book have been, to a large extent, pointless.



Links to Gathering User Requirements

Before starting, it is useful to point backwards to [Gathering User Requirements](#)). The gathering and analysis of user data shares similarities and also some of these techniques can be used in Gathering User Requirements and vice-versa.

As we shall see, good design is based upon an understanding of how data should be collected, how that data should be analysed, and how the ethical aspects of both the collection and analysis should proceed. However, the twin most important aspects of any evaluation are an understanding of science, and more specifically the scientific method; and an understanding of the importance of how you select your participants (called ‘subjects’ in the *old-days* and still referred to as such in some current texts).

You probably think you already know what science is, how it is conducted, and how science progresses; but you are probably wrong. We will not be going very deep into the philosophy of science, but cover it as an overview as to how you should think about your evaluations and how the scientific method will help you in understanding how to collect and analyse your data; to enable you to progress your understanding of how your interventions assist users; key to this is the selection of a representative sample of participants.

Science works by the concept of generalisation – as do statistics, in which we expect to test our hypothesis by observing recording and measuring user behaviour in certain conditions. The results of this analysis enabling us to disprove our hypothesis, or support it, thereby applying our results – collected from a small subset of the population (called a sample) – to the entirety of that population. You might find a video demonstrating this concept more helpful [**Veritasium, 2013**].

The way this works can be via two methods:

1. *Inductive* reasoning, which evaluates and then applies to the general ‘population’ abstractions of observations of individual instances of [members of the same population, for example](#); and,

2. *Deductive* reasoning, which evaluates a set of premises which then necessitate a conclusion¹ – for example: {(1) Herbivores only eat plant matter; (2) All vegetables contain only plant matter; (3) All cows are herbivores} → Therefore, vegetables are a suitable food source for Cows².

Now, let's have a look at the scientific method in a little more detail.

16.1 Scientific Bedrock

Testing and evaluation are the key enablers behind all user experience interaction work, both from a scientific and practical perspective. These twin concepts are based within a key aspect of HCI, the scientific method, and its application as a means of discovering information contained within the twin domains of human behaviour and use of the computer. It is then appropriate that you understand this key point before I move on to talking in more detail about the actual implementation of the scientific method in the form of validating your interface design.

As you may have noticed, UX takes many of its core principles from disjoint disciplines such as sociology and psychology as well as interface design and ergonomics, however, laying at its heart is empiricism³. The scientific landscape, certainly within the domain of the human sciences, is contingent on the concept of logical positivism. Positivism attempts to develop a principled way of approaching enquiry through a combination of empiricism, methods employed within the natural sciences, and rationalism⁴. The combination of empiricism (observation) and rationalism (deduction) make a powerful union that combines the best points of both to form the scientific method.

The scientific method is a body of techniques for investigation and knowledge acquisition [Rosenberg, 2012]. To be scientific, a method of inquiry must be based on the gathering of observable, empirical and measurable evidence, and be subject to specific principles of reasoning. It consists of the collection of data through observation and experimentation, and one of the first to outline the specifics of the scientific method was John Stuart Mill. Mill's method of agreement argues that if two or more instances of a phenomenon under investigation have only one of several possible causal circumstances in common, then the circumstance in which all the instances agree is the cause of the phenomenon of interest. However, a more strict method called the indirect method of difference can also be applied. Firstly, the method of agreement is applied to a specific case of interest, and then the same principle of the agreement is applied to the inverse case of interest.

¹Therefore, the conclusion must be true provided that the premises are true.

²Note that you could not say 'Therefore, all cows eat vegetables' because fruit also contains only plant matter; as do grass and trees.

³In philosophy, empiricism is a theory of knowledge which asserts that knowledge arises from experience. Empiricism is one of several competing views about how we know 'things,' part of the branch of philosophy called epistemology, or 'the Theory of Knowledge'. Empiricism emphasises the role of experience and evidence, especially sensory perception, in the formation of ideas, while discounting the notion of innate ideas.}

⁴In which the criterion of the truth is not sensory but intellectual and deductive.

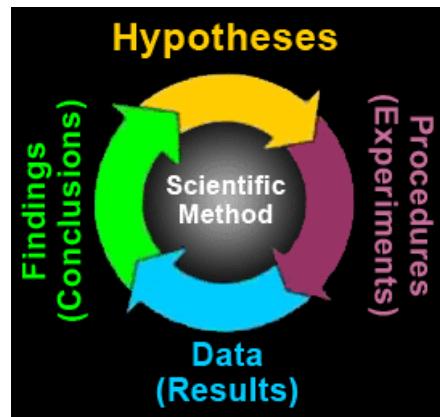


Figure: The Scientific Method. The Scientific Method. —Image Credit: Wikimedia.

Philosophical debate has found aspects of the method to be weak and has generated additional layers of reasoning which augment the method. For instance, the concept of ‘refutability’ was first proposed by the philosopher Karl Popper. This concept suggests that any assertion made must have the possibility of being falsified (or refuted). This does not mean that the assertion is false, but just that it is possible to refute the statement. This refutability is an important concept in science. Indeed, the term ‘testability’ is related and means that an assertion can be falsified through experimentation alone.

So we can see that put simply (see [Figure: The Scientific Method](#)), the scientific method enables us to test if the things we believe to be true, are in fact true. At its most basic the method progresses along the following lines:

1. Firstly, we create a hypothesis that, in the best case, cannot be otherwise interpreted and is ‘refutable’; for example we might make the statement ‘all swans are white’. In this case, we may have travelled widely and tried to observe swans in every country and continent in an attempt to support our hypothesis.
2. While, we may be able to amass many observations of white swans we must also realise that a statement must be refutable. If the hypothesis remains intact it must be correct; in our example, we may try to observe every swan that exists in, say, the UK, or Europe, or the Americas, which is not white.
3. However, one instance of an observation of a non-white swan will disapprove our hypothesis; in this case, when we arrive in Australia we discover a black swan, in this case, we can see all swans are not white and our hypothesis is found to be incorrect.

There have been many debates regarding the question of whether inductive reasoning leads to truth. In general, we can make some inductive leaps if they are based on good science, and these leaps may not be accurate, but may well assist us in our understanding. In the UX domain, we use mathematical (statistical) methods to help us understand these points. For example, if we collect a set of data which compares the time it takes to select a point of the screen with the mouse then we can make sure that data is valid within just the set of people who participated; this is called internal validity. However, we then want to generalise these results to enable us to say something about the wider population, here we can use well formed and tested statistical tests that enable us to mathematically generalise to a population; this is

called external validity. In the human sciences, there is often no 100% certainty, all we have is a level of confidence in how a particular test relates to the population, and, therefore, how useful the knowledge generated from it is.

16.1.1 Variables

Many variable types have arisen in the human sciences methodology to enable better discussion and more accurate planning of these more controlled experiments. In general, evaluations can be characterised by the presence of a subject variable, a behavioural variable (the behavioural variables are really the user demographics, and these are really only used as part of the analysis or in participant selection to create a representative sample of the population.), a stimulus variable, and an observable response. For, the UX specialist this means that the behavioural variable can be equated to the user, the stimulus variables can be equated to the interface or the computer system, and the observable response is a thing that we often measure to understand if there is a benefit after we have manipulated the stimulus. In more detail, the subject variable discusses aspects such as the participant and the characteristics of that participant that can be used to classify them; so this would mean factors such as age, weight, gender, etc. The experimental evaluation also relies on independent and dependent variables. The confusing aspect of all of this is the same variable is sometimes named differently in different subjects; so:

- Behavioural = demographics, or subject variables, or conditions;
- stimulus = independent variable, or input, or cause; and
- response = dependent variable, or output, or effect;

The independent variable is the thing that we manipulate, in UX, this is normally an aspect of the interface that is under investigation; for instance the level of a menu item, or the position of a click button. The dependent variable is the thing that we measure, the response, in UX, this is often the response of a user interacting with the interface or system. Further, a constant is any variable that is prevented from varying and an extraneous variable is any variable other than the independent variable that might affect the dependent measure and might confound the result. Therefore, we can see that the definition of the variables along with an understanding of their presence and the means by which to control them are key to creating a well-found experimental study.

There are many different factors that may affect the internal or external validity of an experiment; called confounding variables. The effects of confounding variables can be decreased by adequate preparation of the laboratory setting for the experimental work such that participants feel comfortable; single or double blind procedures in which the UX specialist is applying the evaluation does not know the desired outcome. Or, in the case of triple blind procedures, where the practitioner applying the evaluation and the practitioner analysing the results does not know the desired outcome. Multiple observers may also be used in trying to reduce the confounding variables as well as specifically objective measures and in some cases automation of the process. Finally, one of the key pillars of evaluation is that the work can be replicated by a third party, and this may be one of the most useful approaches when tried to minimise the introduction of confounding variables, because all aspects of the experimentation, except the methodology and hypotheses, are different.

16.1.2 Measuring Variables

Measurement is another factor that needs accurate control. There are three main types of scales to facilitate measurement, these being the nominal scale, which denotes identity; the ordinal scale, which denotes identity and magnitude; and the interval scale, which denotes identity, magnitude and has the benefit of equal intervals. There is also a fourth scale called the ratio scale that has the positive properties of the three we have already seen as well as a true zero point. Thus, ratio scales provide the best match to the real number system, and we can carry out all of the possible mathematical operations using such scales. These scales are seen to be hierarchical in rigour such that the nominal scale is the least rigorous and the ratio scale is the most rigorous. These are often misunderstood so let me clarify in a different way⁵:

1. **Nominal Variable** (plural nominal variables). A variable with values that have no numerical value, such as gender or occupation. For example: *opposite, alternate, whorled*. Also known as **Categorical Variable** (plural categorical variables).
2. **Ordinal Variable** (plural ordinal variables). A variable with values whose order is significant, but on which no meaningful arithmetic-like operations can be performed. For example: *fast < very fast < very, very fast*
3. **Cardinal Variable (not very used - here for completeness)** (plural cardinal variables). A variable whose values are ordered that can be multiplied by a scalar, and for which the magnitude of differences in values is meaningful. The difference between the values j and the value j+1 is the same as the difference between the values k and k +1. For example: *wages, population*.
4. **Interval Variable** (plural interval variables). An ordinal variable with the additional property that the magnitudes of the differences between two values are meaningful. For example: *10PM (today) > 8PM (today) – 10PM (today) - 8PM (today) = 2 hours*.
5. **Ratio Variable** (plural ratio variables). A variable with the features of interval variable and, additionally, whose any two values have a meaningful ratio, making the operations of multiplication and division meaningful. For example: *10 meters per second > 8 meters per second – 10 mps - 8 mps = 2 mps*.

Also, the scale can be distorted so the observation is no longer accurately reflect reality, this is called measurement error and is part of the operational definition of the variable; i.e. the definition of the variable regarding the procedures used to measure or manipulate that variable. Further, measurement has an aspect of reliability that must also be understood and refers to the measure's reproducibility. This reliability can be subdivided into interrater reliability, test-retest reliability, and internal consistency reliability. The interrater reliability requires that when a judgement by a UX specialist is required, then more than one practitioner should judge the same aspect, and these judgements should be matching. Test-retest reliability requires that if a set of variables should be stable over time, then if tested at period one and then retested at period two, the answer from those measures should be consistent. Finally internal consistency reliability is the aspect of reliability which means that when participants are tested, or observed, under several different conditions the outcomes of those observations are consistent.

⁵http://en.wiktionary.org/wiki/nominal_variable

16.1.3 Hypothesis Testing

Variables and their measurement are important because they inform the experimental design process and the kind of analysis that will be possible once the data has been collected. In general, the lower the number of independent variables, the more confident we can be about the data collected and the results of the analysis. Collecting many different types of variable data at the same time may be a pragmatic choice, in that the participant has already been recruited and is already in the laboratory environment. However, it will make data analysis much more difficult. Indeed, variable measurement is key to the dominant principle of the scientific method, namely, hypothesis testing. The hypothesis is a statement that has the additional aspect of being refutable and in this case, it has a ‘null hypothesis’⁶ which dictates that there is no difference between two conditions beyond chance differences. In addition to this null hypothesis, we have the confounding variable hypothesis that states that a statistically significant difference in the predicted direction can be found at the surety of the observed differences being due to an independent variable as opposed to some extent this variable cannot be supported. Finally, the causal hypothesis states that the independent variable has the predicted effect on the dependent variable.

16.2 Evaluation Design and Analysis

The design of experiments (DOE, DOX, or experimental design) is the design of any task that aims to describe and explain the variation of information under conditions that are hypothesized to reflect the variation. UX works with observational experiments for which you must know what you are trying to understand, gather the correct data, use the correct analysis tool to uncover knowledge, and make sure you can test your results. An experiment is supposed to be dispassionate ...but it rarely is. Typically you are collecting data to discover. You are actively looking for something, looking for relationships in collected data, so you can model some interaction, behaviour, or process, so you are able to better build a system.

Now, the scientific method is not applied in-and-of itself, but is captured as part of evaluation methods that you will need to build into an evaluation design such that the questions you want to be answered, are answered; let's look at that now.

Before getting started just consider that you need to [include ethics in your design here too](#).

Evaluation design and analysis are two of the most important aspects of preparing to validate the user experience. Indeed for any scientific discipline, designing the validation methodology and analysis techniques from the very beginning enables you to understand and allay any possible problems about missing data or the general conduct of the evaluation process [**Floridi, 2004**]. Also, a full evaluation plan, including analytic techniques, is often one of the key aspects of [UX work required for ethical purposes](#) – and also demonstrates due diligence.

⁶Null being from the Latin ‘nullus’, which means not any; hence, not any difference between the two conditions beyond chance variations.

16.2.1 Design

Evaluation Design is so important that attempts to formalise it exist through the human sciences; the methodological debates of anthropology and sociology stem in some regard from the problems and inadequacies of single method research. In general, there are some stages that the UX'er must undertake to try to understand the domain of investigation. Indeed, research must begin with ideas and so the idea generating stage is of key importance; for the UX specialist, this may be a moot point as these ideas may already have been developed as part of the complete system design. The UX'er will also need to understand and define the various problems that may be encountered within both the system, and the study, and design a set of inter-related procedures which will enable data to be collected and any gaps in that data, identified. Once these procedures have been completed, you may move into the observation stage whereby the evaluations will be conducted. Once the data has been collected the analysis stage can begin, in which additional collection may be required if after analysis there are perceived but unforeseen weaknesses within the research design. Next, the UX'er can interpret the data based on their experience, knowledge of other work within the field, and based on the results of the data analysis stage. Finally, complete information is to be communicated to the stakeholders. It is often advisable at this stage to create a technical report whereby all the data, findings, and analysis are listed in minute detail. The UX specialist can then create executive summaries or extended abstracts for faster consumption while still ensuring that the basis of those shortened reports is available if required.

Often, different subject domains will espouse, or, at least, be heavily based upon, a single favourite research methodology. In anthropology this is often participant observation, in the social sciences this is often the questionnaire, in psychology this is mainly non-naturalistic experimentation, in history this is the firsthand account, and in UX, this is very often the qualitative summative evaluation. While it is correct that hybrid approaches are often espoused by communities within each of these domains the general predisposition for single methodology research designs are still prevalent. In an interdisciplinary subject area such as UX, this is not acceptable. Indeed, this is especially the case when the number of participants for a particular evaluation is in the order of 10's as opposed to the 100's or even 1000's often found in social science quantitative survey work. By using single methodology designs with small numbers of participants, the UX specialist cannot be assured of the validity of the outcomes. Indeed, over-reliance on summative evaluation suggests that assessing the impact of the interface, or the system design, in general, has not been taken seriously by the developers, [or maybe seen as a convenient post-hoc add-on](#).

Quantitative methods often provide a broad overview but the richness of the data is thin; qualitative methods are often very specific but provide a very rich and deep set of data; finally, laboratory-based methods usually provide exact experimental validation with the key aspect of refutation, but again are often limited in the number of participants tested, and are mostly ecologically unsound (as they are not naturalistic). In this case, the UX specialist should design their research methodology to include aspects of qualitative, quantitative, and laboratory-based evaluation. Building a qualitative body of evidence using combinations of participant observation, unobtrusive methods, and interview technique will enable well-found hypotheses to be formed ([we'll look at these in more detail](#)). These hypotheses can be initially tested by the use of quantitative

survey based approaches such as questionnaires or walk-through style approaches. A combination of this work can then be used to design laboratory-based experimental work using scenarios that are as natural as possible. Again, quantitative approaches can be revisited as confirmation of the laboratory-based testing, and to gain more insight into the interaction situation after the introduction of the interface components. Using all these techniques in combination means that the development can be supported with statistical information in a quantitative fashion, but with the added advantage of a deeper understanding which can only be found through qualitative work. Finally, the hypotheses and assertions that underpinned the project can be evaluated in the laboratory, under true experimental conditions, which enable the possibility of refutation.

16.2.2 Analysis

Once you've designed your evaluation, are sure you are getting back the data that will enable you to answer your UX questions, you'll need to analyse that data.

Now, data analysis enables you to do three things: firstly, it enables you to describe the results of your work in an objective way. Secondly, it enables you to generalise these results into the wider population. And finally, it enables you to support your evaluation hypotheses that were defined at the start of the validation process. Only by fulfilling these aspects of the evaluation design can your evaluation be considered valid, and useful for understanding if your interface alterations have been successful in the context of the user experience. You will need to be able to plan your data analysis from the outset of the work, and this analysis will inform the way that you design aspects of the data collection. Indeed to some extent, the analysis will dictate the different kinds of methodologies employed [so that a spread of research methodologies are used](#).

Even when it comes to qualitative data analysis, you will find that there is a predisposition to try and quantify aspects of the work. In general, this quantification is known as coding and involves categorising phrases, sentences, and aspects of the quantitative work such that patterns within an individual observation, and within a sample of observations, may become apparent. Coding is a very familiar technique within anthropology and sociology. However, it can also be used when analysing archival material; in some contexts, this is known as narrative analysis.

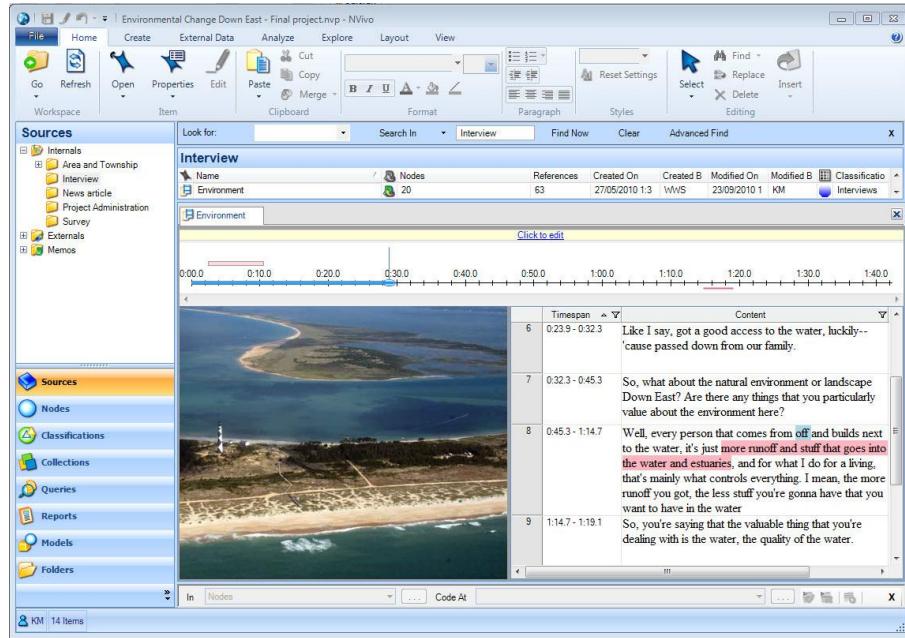


Figure: NVivo Tool. NVivo by QSR International. –Image Credit: Wikimedia.

Luckily there are a number of tools – called Computer Assisted/Aided Qualitative Data Analysis (CAQDAS) – to aid qualitative research such as transcription analysis, coding and text interpretation, recursive abstraction, content analysis, discourse analysis, grounded theory methodology, etc. (see [Figure: NVivo Tool](#)); and include both open source and closed systems.

These include:

- A.nnotate;
- Aquad;
- Atlas.ti;
- Coding Analysis Toolkit (CAT);
- Compendium;
- HyperRESEARCH;
- MAXQDA;
- NVivo;
- Widget Research & Design;
- Qiqqa;
- RQDA;
- Transana;
- Weft QDA; and
- XSight.

However, ‘NVivo’ is probably the best known and most used.

When analysing qualitative work, and when coding is not readily applicable, the UX specialist often resorts to inference and deduction. However, in this case, additional methods of confirmation must then be employed so that the validity of the analysis is maintained; this is often referred to as ‘in situ’ analysis. Remember, most quantitative work spans, at the very minimum, a number of months and therefore keeping running

field notes, observations, inferences and deductions, and applying ad-hoc retests, for confirmation purposes, in the field is both normal and advisable; data collection is often not possible once the practitioner has left the field work setting.

This said, the vast majority of UX analysis work uses statistical tests to support the internal and external validity of the research being undertaken. Statistical tests can be applied in both a quantitative and laboratory-based setting. As we have already discussed, internal validity is the correctness of the data contained within the sample itself. While external validity is the generalisability of the results of the analysis of the sample; i.e. whether the sample can be accurately generalised to the population from which the sample has been drawn ([we'll discuss this a little later](#)). Remember, the value of sampling is that it is an abbreviated technique for assessing an entire population, which will often be too large for every member to be sampled individually. In this case, the sample is intended to be generalizable and applicable across an entire sample so that it is representative of the population; an analysis of every individual within the population is known as a census.

Statistics then are the cornerstone of much analysis work within the UX domain. As a UX specialist, you'll be expected to understand how statistics relates to your work and how to use them to describe the data you have collected, and support your general assertions about that data. I do not intend to go into full treatises of statistical analysis at this point ([there is more coming](#)), there are many texts that cover this in far more detail than I can. In general, you will be using statistical packages such as SPSS (or PSPP in Linux) to perform certain kinds of tests. It is, therefore, important that you understand the types of tests you wish to apply based on the data that has been collected. In general, there are two main types of statistical tests: descriptive, to support internal validity; and inferential, to support external validity.

Descriptive statistics, are mainly concerned with analysing your data concerning the standard normal distribution; that you could expect an entire population to exhibit. Most of these descriptive tests are to enable you to understand how much your data differs from a perfect standard distribution and in this way enable you to describe and understand its characteristics. These tests are often known as measures of central tendency and are coupled with variance and standard deviation. It is often useful to graph your data so that you can understand its characteristics concerning aspects such as skew and kurtosis. By understanding the nature and characteristics of the data you've collected, you will be able to decide which kinds of inferential testing you require. This is because most statistical tests assume a standard normal distribution. If your data does not exhibit this distribution, within certain bounds, then you will need to choose different tests so that you can make accurate analyses.

Inferential statistics, are used to support the generalisability, of your samples' characteristics, about that of the general population from which the sample is drawn. The key aspect of inferential testing is the t-test, where 't' is a distribution, just like a normal distribution and is simply an adjusted version of the normal distribution that works better for small sample sizes. Inferential testing is divided into two types: parametric, and non-parametric. Parametric tests are the set of statistical tests that are applied to data that exhibit a standard normal distribution. Whereas, non-parametric tests are those which are applied to data that are not normally distributed. Modern parametric methods, General Linear Model (GLM), do exist which work on normally distributed data and incorporate many different statistical models such as the ANOVA and ANCOVA.

However, as a UX specialist, you will normally be collecting data from a small number of users (often between 15 and 40), which means that because your sample size is relatively small, its distribution will often not be normally distributed, in this case, you will mainly be concerned with non-parametric testing.

16.2.3 Participants

Participants – users – are the most important aspect of your methodological plan because without choosing a representative sample you cannot hope to validate your work accurately. Originally the sampling procedures for participants came from human facing disciplines such as anthropology and sociology; and so quite naturally, much of the terminology is still specific to these disciplines. Do not let this worry you, once you get the hang of it, the terminology is very easy to understand and apply.

The most important thing to remember is that the participants you select, regardless of the way in which you select them, need to be a representative sample of the wider population of your users; i.e. the people who will use your application, interfaces, and their interactive components. This means that if you expect your user base to be, say, above the age of 50, then, testing your application with a mixture of users including those below the age of 50 is not appropriate. The sample is not representative and therefore, any results that you obtain will be (possibly) invalid.

Making sure that you only get the correct participants for your work is also very difficult. This is because you will be tempted – participants normally being in such short supply – to include as many people as possible, with many different demographics, to make up your participant numbers⁷, you'll be tempted to increase your participants because the alpha (α - also known as the probability ' p ') value in any statistical test is very responsive to participant numbers – often the more, the better. This is known as having a heterogeneous set, but in most cases you want a homogeneous set.

Good sampling procedures must be used when selecting individuals for inclusion in the sample. The details of the sample design its size and the specific procedures used for selecting individuals will influence the tests precision. There are three characteristics of a 'Sample Frame', these being comprehensiveness, whether the probability of selection can be calculated; efficiency, how easily can the sample be selected and recruited, and method:

1. **Simple Random Sampling 'Probabilistic'** – Simple random sampling equates to drawing balls at a tombola. The selection of the first has no bearing and is fully independent of, the second or the third, and so forth. This is often accomplished in the real world by the use of random number tables or, with the advent of computer technology, by random number generators;
2. **Systematic Sampling Probabilistic** – Systematic samples are a variation of random sampling whereby each possible participant is allocated a number, with participants being selected based on some systematic algorithm. In the real world we may list participants numbering them from, say, one to three hundred and picking every seventh participant, for instance;

⁷...also, as we'll see this later

3. **Stratified Sampling** *Probabilistic* — Stratified samples are used to reduce the normal sampling variation that is often introduced in random sampling methods. This means that certain aspects of the sample may become apparent as that sample is selected. In this case, subsequent samples are selected based on these characteristics, this means that a sample can be produced that is more likely to look like the total population than a simple random sample;
4. **Multistage Sampling** *Probabilistic* — Multistage sampling is a strategy for linking populations to some grouping. If a sample was drawn from, say, the University of Manchester then this may not be representative of all universities within the United Kingdom. In this case, multistage sampling could be used whereby a random sample is drawn from multiple different universities independently and then integrated. In this way we can ensure the generalisability of the findings; and
5. **Quota Sampling** '*Non-Probabilistic*' — Almost all non-governmental polling groups or market research companies rely heavily on non-probability sampling methods; the most accurate of these is seen to be quota based sampling. Here, a certain demographic profile is used to drive the selection process, with participants often approached on the street. In this case, a certain number of participants are selected, based on each point in the demographic profile, to ensure that an accurate cross-section of the population is selected;
6. **Snowball Sampling** *Non-Probabilistic* — The process of snowball sampling is much like asking your participants to nominate another person with the same trait as them.
7. **Convenience Sampling** *Non-Probabilistic* — The participants are selected just because they are easiest to recruit for the study and the UX'er did not consider selecting participants that are representative of the entire population.
8. **Judgmental Sampling** *Non-Probabilistic* — This type of sampling technique is also known as purposive sampling and authoritative sampling. Purposive sampling is used in cases where the speciality of an authority can select a more representative sample that can bring more accurate results than by using other probability sampling techniques.

For the normal UX evaluation ‘Quota’ or ‘Judgmental’ sampling are used, however ‘Snowball’ and ‘Convenience’ sampling are also popular choices if you subscribe to the premise that your users are in proximity to you, or that most users know each other. The relative size of the sample is not important, however, the absolute size is, a sample size of between 50 and 200 is optimal as anything greater than 200 starts to give highly diminished returns for the effort required to select the sample.

Finally, you should understand Bias – or sampling error – which is a critical component of total survey design, in which the sampling process can affect the quality of the survey estimates. If the frame excludes some people who should be included, there will be a bias, and if the sampling processes are not probabilistic, there can also be a bias. Finally, the failure to collect answers from everybody is a third potential source of bias. Indeed, non-response rates are important because there is often a reason respondents do not return their surveys, and it may be this reason that is important to the survey investigators. There are multiple ways of reducing non-response. However, non-response should be taken into account when it comes to probability samples.

16.2.4 Evaluation++

In the first part of this chapter, we have placed the scientific bedrock and discussed how to design and analyse evaluations, which in our case, will mostly be qualitative. However, evaluations do not stop at the qualitative level but move on to quantitative work and work that is both laboratory-based and very well controlled. While I do not propose an in-depth treatise of experimental work, I expect that some aspects might be useful, especially for communications inside the organisation and as a basic primer if you are required to create these more tightly controlled evaluations. [There is more discussion on controlled laboratory work](#) and you can think of this as Evaluation++⁸. Evaluation++ is the gold standard for deeply rigorous scientific work and is mostly used in the scientific research domain.

To make things easier, let's change our terminology from Evaluation++ to 'experimental', to signify the higher degree of rigour and the fundamental use of the scientific method (which mostly cannot be used for qualitative work, or work which used Grounded Theory⁹) found in these methods.

An experiment has five separate conditions that must be met. Firstly, there must be a hypothesis that predicts the causal effects of the independent variable (the computer interface, say) on the dependent variable (the user). Secondly, there should be two different sets of independent variables. Thirdly, participants are assigned in an unbiased manner. Fourthly, procedures for testing hypotheses are present. Fifthly, there must be some control concerning accentuate internal validity. There are several different kinds of evaluation design ranging from a single variable, independent group designs, through single variable, correlated groups designs, to factorial designs. However, in this text we will primarily focus on the first; single variable, independent group designs.

Single Group, Post-Test – In this case, the evaluation only occurs after the computer-based artefact has been created and also there is only one group of participants for user testing. This means that we cannot understand if there has been an increase or improvement in interaction due to purely to the computer-based artefact, and also, we cannot tell if the results are only due to the single sample group.

Single Group, Pre-Test and Post-Test – In this case, the evaluation occurs both before and after the computer-based artefact has been introduced. Therefore, we can determine if the improvement in interaction is due purely to the computer-based artefact but we cannot tell if the results, indicating either an improvement or not, are due to the single sample group.

Natural Control Group, Pre-Test and Post-Test – This design is similar to the single group, pre-test and post-test however a control group is added but this control group is not randomly selected but is rather naturally occurring; you can think of this as a control group of convenience or opportunity. This means that we can determine if the interaction is due to the computer-based artefact and we can also tell if there is variance between the sample groups.

Randomised Control Group, Pre-Test and Post-Test – This design is the same as

⁸Taken from Kernighan and Ritchie's 'c' programming language, which was called 'c++' by Stroustrup, when he extended it.

⁹...which, rather than beginning with a hypothesis, actually starts with data collection, from which understanding emerges, which then help hypotheses/assertions/theories to be formed. This is almost in a reverse fashion from scientific research and therefore, many scientists feel it be in contradiction to the scientific method.

a natural control group, pre-test and post-test, however, it has the added validity of selecting participants for the control group from a randomised sample.

Within-Subjects – A within-subjects design means that all participants are exposed to all different conditions of the independent variable. In reality, this means that the participants will often see both the pre-test state of the interface and the post-test state of the interface, and from this exposure, a comparison, usually based on a statistical ANOVA test (see later), can be undertaken.

Others – These five designs are not the only ones available however they do represent basic set of experimental designs increasing validity from the first to see last. You may wish to investigate another experiment to design such as Solomon's four group design or multilevel randomised between subject designs etc.

To an extent, the selection of the type of experimental design will be determined by more pragmatic factors than those associated purely with the 'true experiment'. Remember, your objective is not to prove that a certain interface is either good or bad. By exhibiting a certain degree of agnosticism regarding the outcome of the interaction design, you will be better able to make improvements to that design that will enhance its marketability

16.2.5 Pre and Post Test and A/B Testing

A/B testing, also known as split testing, is a statistical method used to compare two versions of something (usually a webpage, advertisement, or application feature) to determine which one performs better in achieving a specific goal or objective. The goal could be anything from increasing click-through rates, conversion rates, or user engagement to improving revenue or user satisfaction.

You first need to clearly define what you want to improve or optimize. For example, you may want to increase the number of users signing up for your service or buying a product. Develop two (or more) different versions of the element you want to test. For instance, if you're testing a website's landing page, you could create two different designs with varying layouts, colours, or calls-to-action (CTAs). Randomly split your website visitors or users into two equal and mutually exclusive groups, A and B (hence the name A/B testing). Group A will see one version (often referred to as the "control"), and group B will see the other version (often called the "variant"). Allow the test to run for a predetermined period, during which both versions will collect data on the chosen metrics, such as click-through rates or conversion rates. A/B testing is a powerful technique because it provides objective data-driven insights, allowing businesses to make informed decisions about design, content, and user experience. By continuously testing and optimizing different elements, organizations can improve their overall performance and achieve their goals more effectively.

Pre and post-test and A/B testing are two different experimental designs used in research and analysis, but they share some similarities and can be used together in certain scenarios. In this design, Pre and Post-Test participants are assessed twice – once before they are exposed to an intervention (pre-test), and once after they have undergone the intervention (post-test). The change between the pre-test and post-test measurements helps evaluate the impact of the intervention. A/B Testing involves comparing two or more versions of something to see which one performs better. So

while this may not be ‘Within-Subjects’ the idea of testing two different interventions is the same.

The main objective of Pre and Post-Test design is to evaluate the effect of an intervention or treatment. Researchers want to determine if there’s a significant change in the outcome variable between the pre-test and post-test measurements. While the primary objective of A/B Testing is to compare two or more versions to identify which one yields better results in achieving a specific goal, such as increased conversion rates or user engagement. Again the two are very similar. Pre and post-test and A/B testing are distinct experimental designs with different objectives. Pre and post-test evaluates the impact of an intervention over time, while A/B testing compares different versions simultaneously to determine which one is more effective in achieving a particular goal. However, they can complement each other when studying the effects of changes or interventions over time and simultaneously evaluating different versions for performance improvements.

16.3 A Note on Statistics & How To Use & Interpret

Most car drivers do not understand the workings of the internal combustion engine or – for that matter – any of the machinery which goes to make up the car. While some people may find it rewarding to understand how to make basic repairs and conduct routine maintenance, this knowledge is not required if all you wish to do is drive the vehicle. Most people do not know much beyond the basics of how to operate the car, choose the right fuel, and know where the wiper fluid goes. It is exactly the same with statistics. While it is true that the ‘engineers’ of statistical methods, the statistician, need to understand all aspects of statistical analysis from first principles; and while it is also true that ‘mechanics’ need to understand how to fix and adapt those methods to their own domains; most users of statistics see themselves purely as ‘drivers’ of these complex statistical machines and as such think that they don’t need to know anything more than the basics required to decide when and how to apply those methods.

Now, my view is not to be that cavalier on the subject, most statistical work is conducted by non-statisticians and in many cases these non-statisticians use a plug and play mentality. They choose the most popular statistical test as opposed to the correct one to fit their datasets, they are not sure why they are applying the data to a certain test, and they not absolutely sure why they are required to hit a 0.05 or 0.01 p-value (whatever ‘p’ is). As an example let’s consider a critique of the recently acclaimed book ‘Academically Adrift’ the principle finding of which claims that “45 percent of the more than 2,000 students tested in the study failed to show significant gains in reasoning and writing skills during their freshman and sophomore years.” suggesting that “Many students aren’t learning very much at all in their first two years of college”.

As Alexander W. Astin¹⁰ discusses in his critique¹¹ “The goal was to see how much they had improved, and to use certain statistical procedures to make a judgment about whether the degree of improvement shown by each student was ‘statistically significant’.”

¹⁰Professor emeritus of higher education and organisational change at the University of California at Los Angeles.

¹¹In ‘Academically Adrift,’ Data Don’t Back Up Sweeping Claim

Austin goes on to say “the method used to determine whether a student’s sophomore score was ‘significantly’ better than his or her freshman score is ill suited to the researchers’ conclusion. The authors compared the difference between the two scores—how much improvement a student showed—with something called the ‘standard error of the difference’ between his or her two scores. If the improvement was at least roughly twice as large as the standard error (specifically, at least 1.96 times larger, which corresponds to the ‘.05 level of confidence’, they concluded that the student ‘improved.’ By that standard, 55 percent of the students showed ‘significant’ improvement—which led, erroneously, to the assertion that 45 percent of the students showed no improvement.”

He continues, ‘The first thing to realise is that, for the purposes of determining how many students failed to learn, the yardstick of ‘significance’ used here—the .05 level of confidence—is utterly arbitrary. Such tests are supposed to control for what statisticians call “Type I errors,” the type you commit when you conclude that there is a real difference, when in fact there is not. But they cannot be used to prove that a student’s score did not improve.’ Stating that, “the basic problem is that the authors used procedures that have been designed to control for Type I errors in order to reach a conclusion that is subject to Type II errors. In plainer English: Just because the amount of improvement in a student’s CLA score is not large enough to be declared ‘statistically significant’ does not prove that the student failed to improve his or her reasoning and writing skills. As a matter of fact, the more stringently we try to control Type I errors, the more Type II errors we commit.”

Finally, Austin tells us that “To show how arbitrary the ‘.05 level’ standard is when used to prove how many students failed to learn, we only have to realise that the authors could have created a far more sensational report if they had instead employed the .01 level, which would have raised the 45 percent figure substantially, perhaps to 60 or 70 percent! On the other hand, if they had used a less-stringent level, say, the .10 level of confidence, the ‘nonsignificant’ percent would have dropped way down, from 45 to perhaps as low as 20 percent. Such a figure would not have been very good grist for the mill of higher-education critics.”

Austin discusses many more problems in the instruments, collection, and analysis than those discussed above, but these should indicate just how open to mis-application, mis-analyses, and mis-interpretation statistics are. You don’t want to be like these kinds of uninformed statistical-tool users; but instead, a mechanic, adapting the tools to your domain can only be accurately done by understanding their application, the situations in which they should be applied, and what their meaning is. This takes a little more knowledge than simple application, but once you realise their simplicity – and power – using statistics becomes a lot less scary, and a lot more exciting!

16.3.1 Fisherian and Bayesian Inference

The relationship between Fisherian inference and Bayesian inference lies in their fundamental approaches to statistical inference, but they differ in their underlying principles and methodologies.

Fisherian inference, also known as classical or frequentist inference, was developed by Sir Ronald A. Fisher and is based on the concept of repeated sampling from a

population. In Fisherian inference, probabilities are associated with the data and are used to make inferences about population parameters.

Fisherian inference heavily relies on Null Hypothesis Significance Testing (NHST), where a null hypothesis is formulated, and statistical tests are performed to determine whether the observed data provide sufficient evidence to reject the null hypothesis in favor of an alternative hypothesis. Fisherian inference considers the sampling distribution of a statistic under repeated sampling to make inferences about the population parameter of interest. The focus is on estimating and testing hypotheses about fixed but unknown parameters. Fisherian inference uses point estimation to estimate population parameters. The most common method is maximum likelihood estimation (MLE), which aims to find the parameter values that maximize the likelihood function given the observed data.

On the other hand, Bayesian inference, developed by Thomas Bayes, takes a different approach by incorporating prior knowledge and updating it with observed data to obtain posterior probabilities.

Bayesian inference assigns probabilities to both the observed data and the parameters. It starts with a prior probability distribution that reflects prior beliefs or knowledge about the parameters and updates it using Bayes' theorem to obtain the posterior distribution. Bayesian inference uses the posterior distribution to estimate population parameters. Instead of providing a single point estimate, Bayesian estimation provides a full posterior distribution, which incorporates both prior knowledge and observed data. Bayesian inference allows for direct model comparison by comparing the posterior probabilities of different models. This enables the selection of the most plausible model given the data and prior beliefs.

While Fisherian inference and Bayesian inference have different philosophical foundations and computational approaches, they are not mutually exclusive. In fact, there are connections and overlaps between the two approaches. Bayesian inference can incorporate frequentist methods, such as maximum likelihood estimation, as special cases when certain assumptions are made. Bayesian inference can provide a coherent framework for incorporating prior information and updating it with new data, allowing for a more comprehensive and flexible analysis. Some Bayesian methods, such as empirical Bayes, borrow concepts from Fisherian inference to construct prior distributions or to estimate hyper-parameters.

In practice, the choice between Fisherian and Bayesian inference often depends on the specific problem, the available prior knowledge, computational resources, and the preference of the analyst. Both approaches have their strengths and weaknesses, and their use depends on the context and goals of the statistical analysis.

In UX we currently use Fisherian inference which we would call ‘Statistics’ while Bayesian inference would be known as a principle component of ‘Machine Learning’.

16.3.2 What Are Statistics?

Before I answer this question, let me make it clear that designing your evaluations is one of the most important aspects of the analysis process. If your evaluations are designed badly you will not be able to apply the correct analysis, and if you cannot apply the correct analysis you will not be able to make any conclusions as to the applicability

or success of your interventions at the interface or interactive level. In reality this means that if this is not done correctly all of your previous work in the UX design and development phases will have been, to a large extent, pointless. Further, any statistical analysis you decide to do will give you incorrect results; these may look right, but they won't be.

Statistics then, are the cornerstone of much analysis work within the UX domain. As a UX specialist you'll be expected to understand how statistics relate to your work and how to use them to describe the data you have collected, and support your general assertions with regard to that data (In general, you will be using statistical packages such as SPSS – or PSPP in Linux – to perform certain kinds of tests.). It is therefore, important that you understand the types of tests you wish to apply based on the data that has been collected; but before this, you need to understand the foundations of statistical analysis. Now there are many texts which cover this in far more detail than I can; but these are mainly written by statisticians. This discussion is different because it is my non-statistician's view, and the topic is described in a slightly different (but I think easier way), based on years of grappling with the topic.

Let's get back to the question in our title "What Are Statistics?". Well, statistics are all about supporting the claims we want to make, but to make these claims we must first understand the populations from which our initial sampling data will be drawn. Don't worry about these two new terms – population and sampling data – they're very easy to understand and are the basis for all statistics; let's talk about them further, now.

First of all let's define a population, the OED tells us it is the 'totality of objects or individuals under consideration'. In this case when we talk about populations we do not necessarily mean the entire population of say the United Kingdom, but rather, the total objects or individuals under our consideration. This means that if we were talking about students taking the user experience course we would define them as the population. If we wanted to find something useful out about this population of UX students, such as the mean exam mark, we may work this out by taking all student marks from our population into account – In reality this is exactly what we would do because there are so few students that a census of this total population is easily accomplished; this is called a census, and the average mark we arrive at is not called a statistic, but rather a parameter. However, suppose that we had to work the mean out manually – because our computer systems are broken down and we had a deadline to meet – in this case arriving at a conclusion may be a long and tedious job. It would be much better if we could do less work and still arrive at an average score of the entire class which we can be reasonably certain of. This is where statistics come into their own; because statistics are related to a representative sample drawn from a larger population.

The vast majority of UX analysis work uses statistical tests to support the internal and external validity of the research being undertaken. As we have already discussed, internal validity is the correctness of the data contained within the sample itself, while external validity is the generalisability to the population of the results of the analysis of the sample; i.e. whether the sample can be accurately generalised to the population from which the sample has been drawn. Remember, the value of sampling is that it is an abbreviated technique for assessing an entire population, which will often be too large for every member to be sampled individually. In this case, the sample is intended to be generalisable and applicable across an entire sample so that it is representative of the

population; an analysis of every individual within the population; the census.

16.3.3 Distributions

You may not realise it but statistics are all about distribution. In the statistical sense the OED defines distribution as ‘The way in which a particular measurement or characteristic is spread over the members of a class’. We can think of these distributions as represented by graphs such that the frequency of appearance is plotted along the x-axis, while the measurement or characteristic is ordered from lowest to highest and then plotted along the y-axis. In statistics we know this as the variable; and in most cases this variable can be represented by the subject variable (such as: age, height, gender, weight, etc) or the observable response / dependent variable.

Let’s think about distributions in a little more detail. We have a theoretical distributions created by statisticians and built upon stable mathematical principles. These theoretical distributions are meant to represent the properties of a generic population. But we also have distributions which are created based on our real-world empirical work representing the actual results of our sampling. To understand how well our sample fits the distribution expected in the general population, we must compare the two; statistical analysis is all about this comparison. Indeed, we can say that the population is related to the theoretical distribution, while the statistic is related to the sample distribution

16.3.3.1 Normal Distributions

Normal Distributions are at the heart of most statistical analysis. These are theoretical distributions and basis for probability calculations which we shall come to in the future. This distribution are also known as a bell shaped curve whereby the highest point in the centre represents the mean (Mean: The average of all the scores.), median (Median: the number which denotes the 50th percentile in a distribution.), or mode (Mode: The number which occurs most frequently in a distribution.); and where approximately 68% of the population fall within one standard deviation unit of this position. These three measures of mean, median, and mode are collectively know as measures of central tendency; and try to describe the central point of the distribution.

16.3.3.2 Standard Deviation

The standard deviation is a simple measure of how spread out the scores in a distribution are. While the range (Range: difference between the highest and lowest scores.) and IQR (Inter-Quartile Range: difference between the scores ate the 25th and 75th percentiles.) are also measures of variability, the variance (Variance: the amount of dispersion in a distribution.) is more closely related to the standard deviation. While we won’t be touching formulas here it is useful to understand exactly what standard deviation really means. First, we measure the difference between a single score in a distribution and the mean score for the distribution; we then find the average of these differences, and we have the standard deviation.

16.3.4 A Quick Guide to Selecting a Statistical Test

There are four quadrants that we can place tests in. We have parametric and non-parametric test - this simply means those with an expected standard distribution (parametric) which would typically need to have a lot of participants over 100 say, and those not expected to follow a standard distribution (non-parametric) so less than 30 and this will normally be the range you are working in as UX'ers.

Next we have causation and correlation. This means that if something correlates it exhibits some kind of relationship but you can't decide of one causes the other to occur - and so these are the most common types of test and are slightly less strong, but easier to see. Causation on the other-hand is a stronger test and requires more data - you will typically no use these tests much. So you are looking at non-parametric correlation.

16.3.5 Correlation Tests

Parametric statistical tests are used to analyse data when certain assumptions about the data distribution and underlying population parameters are met.

- Student's t-test: The t-test is used to compare means between two independent groups. It assesses whether there is a significant difference between the means of the two groups.
- Paired t-test: The paired t-test is used to compare means within the same group or for related samples. It assesses whether there is a significant difference between the means of paired observations.
- Analysis of Variance (ANOVA): ANOVA is used to compare means between multiple independent groups. It determines whether there are significant differences in the means across different groups or levels of a categorical variable.
- Analysis of Covariance (ANCOVA): ANCOVA is an extension of ANOVA that incorporates one or more covariates (continuous variables) into the analysis. It examines whether there are significant differences in the means of the groups after controlling for the effects of the covariates.
- Pearson's correlation coefficient (Pearson's r): This is the most commonly used correlation test, which measures the linear relationship between two variables. It assesses both the strength and direction of the relationship. It assumes that the variables are normally distributed.
- Chi-square test: The chi-square test is used to determine if there is an association or dependence between two categorical variables. It assesses whether there is a significant difference between the observed and expected frequencies in the contingency table.

These are some of the main parametric statistical correlation tests used to assess the relationship between variables. The choice of which test to use depends on the nature of the variables and the specific research question or hypothesis being investigated.

Non-parametric statistical tests are used when the assumptions of parametric tests are violated or when the data is not normally distributed. Here are some of the main non-parametric correlation tests:

- Spearman's rank correlation coefficient (Spearman's rho): This non-parametric test is similar to the parametric Spearman's rank correlation coefficient. It assesses the monotonic relationship between two variables using ranks instead of the actual values of the variables. It is suitable for non-linear relationships and is less sensitive to outliers.
- Kendall's rank correlation coefficient (Kendall's tau): Kendall's tau is another non-parametric measure of the monotonic relationship between two variables. It uses ranks and determines the strength and direction of the association. It is robust to outliers and is suitable for non-linear relationships.
- Goodman and Kruskal's gamma: This non-parametric correlation test is used to assess the association between two ordinal variables. It measures the strength and direction of the relationship, particularly in cases where the data is not normally distributed.
- Somers' D: Somers' D is a non-parametric measure of association that is commonly used when one variable is ordinal, and the other is binary. It evaluates the strength and direction of the relationship between the two variables.
- Kendall's coefficient of concordance: This non-parametric test is used when there are multiple variables that are ranked by multiple observers or raters. It measures the agreement or concordance among the raters' rankings.
- Mann-Whitney U test: The Mann-Whitney U test, also known as the Wilcoxon rank-sum test, is a non-parametric test used to compare the distributions of two independent samples. It assesses whether there is a significant difference between the medians of the two groups.
- Kruskal-Wallis test: The Kruskal-Wallis test is a non-parametric alternative to ANOVA. It compares the distributions of three or more independent groups and determines whether there are significant differences among the medians.

These non-parametric correlation tests are useful alternatives to parametric tests when the assumptions of parametric tests are not met or when dealing with non-normally distributed data. They provide a robust and distribution-free approach to assessing the relationship between variables. Again, the choice of which test to use depends on the type of variables involved and the research question at hand.

16.3.6 Causation Tests

Regression analysis is a statistical method used to examine the relationship between a dependent variable and one or more independent variables. And is one of the main ways in which causation is supported, and aims to model the relationship and understand how changes in the independent variables impact the dependent variable.

The dependent variable, also known as the outcome variable or response variable, is the variable that is being predicted or explained. It is typically a continuous variable. The independent variables, also called predictor variables or regressors, are the variables that are hypothesized to have an influence on the dependent variable. These can be continuous or categorical variables.

The regression analysis estimates the coefficients of the independent variables to create a regression equation. The equation represents a line or curve that best fits the data points and describes the relationship between the variables. The equation

can be used to predict the value of the dependent variable based on the values of the independent variables. There are various types of regression analysis, depending on the nature of the data and the research question.

- Simple Linear Regression: This type of regression involves a single independent variable and a linear relationship between the independent and dependent variables. It estimates the slope and intercept of a straight line.
- Multiple Linear Regression: Multiple linear regression involves two or more independent variables. It estimates the coefficients for each independent variable, indicating the strength and direction of their relationship with the dependent variable while considering other variables.
- Polynomial Regression: Polynomial regression is used when the relationship between the variables is non-linear. It can model curves of different orders (e.g., quadratic or cubic) to capture more complex relationships.
- Logistic Regression: Logistic regression is used when the dependent variable is binary or categorical. It estimates the probabilities or odds of an event occurring based on the independent variables.

The regression analysis provides several statistical measures to assess the goodness of fit and the significance of the model. These include the coefficient of determination (R-squared), which indicates the proportion of the variance in the dependent variable explained by the independent variables, and p-values for the coefficients, indicating the significance of each variable's contribution.

16.4 Caveat

I was pleased recently to receive this (partial) review for a paper submitted to 'Computers in Human Behaviour'. It seems like this reviewer understands the practicalities of UX work. Instead of clinging to old and tired statistical methods more suited to large epidemiology – or sociology – studies this reviewer simply understands:

"The question is whether the data and conclusions already warrant reporting, as clearly this study in many respects is a preliminary one (in spite of the fact that the tool is relatively mature). Numbers of participants are small (8 only), numbers of tasks given are small (4 or 6 depending on how you count), the group is very heterogeneous in their computer literacy, and results are still very sketchy (no firm conclusions, lots of considerations mentioned that could be relevant). This suggests that one had better wait for a more thorough and more extensive study, involving larger numbers of people and test tasks, with a more homogeneous group. I would suggest not to do so. It is hard to get access to test subjects, let alone to a homogeneous group of them. But more importantly, I am convinced that the present report holds numerous valuable lessons for those involved in assisting technologies, particularly those for the elderly. Even though few clear-cut, directly implementable conclusions have been drawn, the article contains a wealth of considerations that are useful to take into account. Doing so would not only result in better assistive technology designs, but also in more sophisticated follow-up experiments in the research community."

Thanks, mystery reviewer! but this review opens up a wider discussion which you, as UX specialists should be aware of; yes, yet another cautionary note. Simply, there is no way that any UX work – *Any UX Work* – can maintain ‘External Validity’ with a single study, the participant numbers are just way too low and heterogeneous – even when we have 50 or 100 users. To suggest any different is both wrong and disingenuous; indeed, even for quota based samples – just answering a simple question – we need in the order of 1000 participants. I would rather, two laboratory-based evaluations using different methods, from different UX groups, using 10 participants, both come up with the same conclusions than a single study of 100 people. In UX, we just end up working with too few people for concrete generalisations to be made – do you think a sample of 100 people is representative of 60 million? I’m thinking not. And, what’s more, the type of study will also fox you... perform a lab evaluation which is tightly controlled for ‘Confounding Factors’ and you only get ‘Internal Validity’, do a more naturalistic study which is ‘ecologically’ valid, and you have the possibility of so many confounding variables that you cannot get generalisability.

‘But surely’, I hear you cry ‘Power Analysis (statistical power) will save us!’¹². ‘We can use power analysis to work out the number of participants, and then work to this number, giving us the external validity we need!’ – Oh if it was only so easy. In reality, ‘Statistical power is the probability of detecting a change given that a change has truly occurred. For a reasonable test of a hypothesis, power should be >0.8 for a test. A value of 0.9 for power translates into a 10% chance that we will miss conclude that a change has occurred when indeed it has not’. But power analysis assumes an alpha of 0.05 (normally), the larger the sample, the more accurate, and the bigger the effect size, the easier it is to find. So again large samples looking for a very easily visible effect (large effect), and without co-variates, gives better results. But, these results are all about accepting or rejecting the null hypothesis – that always states that the sample is no different from the general population, or that there is no difference between the results captured from ‘n’ samples. This, presuming the base case is a good proxy of the population (randomly selected) – which it may not be.

So is there any point in empirical work? Yes, internal validity suggests an application into the general population (which is why we are mostly looking at measures of internal ‘Statistical Validity’). What’s more, one internally valid study is a piece of the puzzle, bolstered with ‘n’ more small but internally valid studies allows us to make more concrete generalisations.

16.5 Summary

So then, UX is used in areas of computer science spectrum that may not seem as though they obviously lend themselves to interaction. Even algorithmic strategies in algorithms and complexity have some aspects of user dependency, as do the modelling and

¹²Read on – but be aware that you may need to skip ahead and then jump back here. I want to tell you this caveat now because it is important to discuss it early which you are still forming your opinions.

simulation aspects of computational science. In this regard, UX is very much at the edge of discovery and the application of that discovery. Indeed, UX requires a firm grasp of the key principles of science, scientific reasoning, and the philosophy of science. This philosophy, principles, and reasoning are required if a thorough understanding of the way humans interact with computers is to be achieved. Further, Schmettow tells us that user numbers cannot be prescribed before knowing just what the study is to accomplish, and Robertson tells us that the interplay between Likert-Type Scales, Statistical Methods, and Effect Sizes are more complicated than we imagine when performing usability evaluations. In a more specific sense if you need to understand, and show, that the improvements made over user interfaces to which you have a responsibility in fact real and quantifiable, and conform to the highest ethical frameworks.

16.5.1 Optional Further Reading

- [J. Baggini] and P. S. Fosl. The philosopher's toolkit: a compendium of philosophical concepts and methods. Wiley-Blackwell, Oxford, 2nd ed edition, 2010.
- [L. Floridi.] The Blackwell guide to the philosophy of computing and information. Blackwell Pub., Malden, MA, 2004.
- [A. Rosenberg.] Philosophy of Science: a contemporary introduction. Routledge contemporary introductions to philosophy. Routledge, New York, 3rd ed. edition, 2012.
- [J. Cohen] — Applied multiple regression/correlation analysis for the behavioral sciences. L. Erlbaum Associates, Mahwah, N.J., 3rd ed edition, 2003.
- [P. Dugard, P. File, J. B. Todman] — Single-case and small-n experimental designs: a practical guide to randomization tests, second edition. Routledge Academic, New York, NY, 2nd ed edition, 2012.
- [M. Forshaw] Easy statistics in psychology: a BPS guide. Blackwell Pub., Malden, MA, 2007.
- [J. J. Hox.] Multilevel analysis: techniques and applications. Quantitative methodology series. Routledge, New York, 2nd ed edition, 2010.
- [T. C. Urdan.] Statistics in plain English. Routledge, New York, 3rd ed edition, 2010.



Self Assessment Questions

Try these without reference to the text:

1. What is the scientific method and why is it important?
2. What do we mean by internal and external validity?
3. What is the single most important reason for having a set of ethical procedures?
4. What are the eight key ethical principles (give a brief rationale for each)?
5. Why is conforming to scientific principles key to good ethical designs?

Part V: In Real Life

17. In Real Life

What we do not understand we do not possess

– Johann Wolfgang von Goethe

UX work design espoused by the ‘standard’ textbooks often assume perfection in the process. They assume that there are no limitations on the time required for the work to occur; they assume that there are no limitations on the skills required for the work to occur, and they assume that there are no limitations on the instruments required for the work to occur – to name but three.

17.1 Realistic, Practical, Pragmatic, and Sloppy!

However, we can characterise UX as having many pragmatic limitations such as the knowledge required, weighted against the limitations of time, participants, resources, and skills, along with the practicalities associated with data capture, sampling, and analysis. Further, because domains are often so separated what may seem like a short amount of time to one researcher may seem like an incredibly long amount of time to another. This proves to be an even greater problem in the UX domain where often the work is highly interdisciplinary, and methodologies are used from many different and competing domains [Vogt et al., 2012]. A domain expert from one may feel that time is grossly over exaggerated or under exaggerated based on their experience of their specific domain. Further, it is often assumed that participants and their recruitment will be an easy administrative task while most textbooks admit that some form of remuneration is often required; they certainly expect that all participant recruitment will have taken place before experimentation commences such that the sequence of experimentation can be very linear and well-defined. Participant recruitment on the fly is often not discussed, and the difficulty in soliciting participants while maintaining an accurate sampling, and getting those participants to return for within sampling work is mainly assumed as a given. Also, resources are often taken for granted with textbooks making no suggestion as to how aspects such as instrumentation, financial reward, and experimenter availability should be addressed. Rather it is given that the correct instruments will be available, financial rewards for bursaries for participants and recruitment will be available, and in the case of double or triple blind trials, the requisite number of UXer’s will be on hand to assist in carrying out the work.

In cases where the working domain is interdisciplinary, there is also an implicit assumption that any UX'er can become an expert very quickly at any other research methodology. Indeed, many textbooks present a series of methodologies for the UXer's consumption and with an expectation that an understanding of the procedure is all that is required for an accurate application of the method. However, different kinds of interdisciplinary research are often very difficult to conduct, and for the best results, which include an understanding of all the pitfalls, the UX'er often needs to perform

many of these methodologies before even a basic understanding of the problems and best practices can arise. Indeed, this is one of the reasons I've created this chapter; to try and give the novice UX'er a head start in understanding the problems and contentious issues surrounding each of these working methodologies based on my failings and experiences over the last ten years of user experience work in Interaction Analysis and Modelling.

However, before we look more closely at the way things are done '[in-the-wild](#)' it is useful to [take a look at the textbook methods](#)). Have a look back to rediscover how things should be done in a well-defined world without limitations or practical considerations [\[Alkin, 2011\]](#); a world which does not allow reality to impinge on the method. In this chapter, we intend to give an overview of the possible trade-offs and limitations based on the real-world situations which often apply. These real-world situations have been experienced by our UX teams in both academic and contract work for third-party organisations (both commercial and non-commercial). Here we describe the kind of limitations that often affect evaluation design, and thereby enable new UX'ers to the field to gain an understand the kinds of decisions and choices that need to be made in different situations, while still enabling a valid scientific outcome to the end results [\[Gliner et al., 2009\]](#).

We have come to a more realistic understanding of the work that is often possible under applied conditions, as opposed to that found in most texts. In reality, these practical considerations centre around our understanding of the knowledge requirements and purpose of the work, the limitations which will always be imposed upon that work, and the availability of skills which are required to accomplish a satisfactory practical outcome. By understanding these aspects more realistically from the outset, the scientific value of the evaluations that are undertaken is much more likely to be maintained. This contrasts with a dogged following of textbooks methodologies which must be changed and modified 'on-the-fly' as the situation necessitates in, primarily, a reactionary way as opposed to our proactive methods; based on an expectation of imperfection within the process.

17.2 Expect Imperfection

Designing your evaluation is one of the most important aspects of any piece of UX work, certainly within the human factors domain. A good evaluation design will save time and energy from the outset and as the project proceeds. There is often a tendency to not fully think about the methodology and plan the steps to a suitable outcome from the start. However, this normally turns out to be an unproductive strategy because the methodology is not fully formed, and from the point of view of an ethics committee or grant review, the holes within that methodology suggest that not enough preparatory work has been undertaken. Further, this suggests that the planning and methodology are weak and that the desired outcomes of the work are unlikely to be successful.

Of course, this is the way most texts on the subject of evaluation methodologies and evaluation design justify their methodologies. Textbooks suggest that the UX'er proceed in a well ordered and structured manner; and with good reason. However, this is often not possible due to a number of different and conflicting factors. In some cases it may be because other parts of the software project have not been completed on time, or that

human factors work was not included within the planning of the original methodology or software design; engineers notoriously often forget users.

Even if the time for a user study is included, if the project planning is completed by someone without UX experience then it is likely that there will be an underestimate of the time and resources required for proper evaluations. In the case of academic work, this can often happen when the human factors testing and evaluation are added as an afterthought, or when the UX specialist is not consulted from the outset and is only presented with a testing regime into which significant experience in human factors is not being applied.

We will see that design in the real world is characterised mainly by limitations. However, many textbooks on the subject are characterised by assuming a perfect evaluation scenario, which the UX specialist only has to apply for a valid outcome to be forthcoming. In reality, this happens only very rarely or in cases where the UX component has been designed by the specialist who will be responsible for the output. Even then many implicit factors will weigh on the decision as to which evaluation methodologies to use; as opposed to a purely neutral decision based on the best methodology to elicit the knowledge required. These factors are often based around more realistic considerations such as over-familiarity of a particular methodology, the specialists' competence with the statistical analysis techniques required to support the outcome, and the pre-known outcomes which are preferable to the organisation paying for the evaluation; indeed in some case, user evaluation is just regarded as a 'box-ticking' activity to show due diligence. Indeed, there are many implicit assumptions about the kind of knowledge that is required or practicable. These assumptions are often drawn because certain experimental designs and research methodologies are often used in specific domains. It, therefore, becomes a 'fait accompli' that certain knowledge, or kinds of knowledge, is favoured beyond others in certain domains because certain research methodologies return that sort of knowledge. In reality, it is often the other way round in that a certain kind of knowledge is required and methodologies to collect that knowledge have evolved and have been developed within of a specific domain which requires this knowledge.

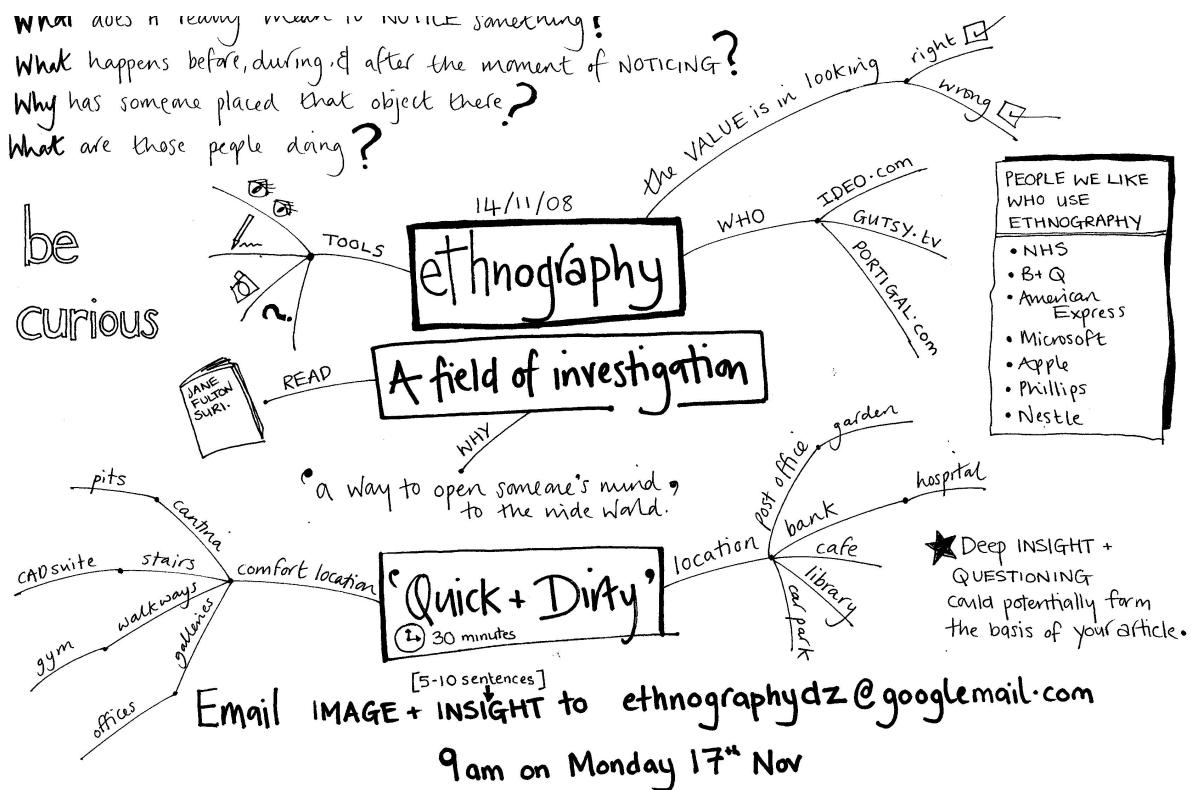


Figure: Ethnography? Ethnographic Methods - Expectations in Anthropology. —Image Credit: Lauren Currie, 2008 (<http://redjotter.wordpress.com/2008/11/14/introducing-ethnography/>).

In this case, many assumptions about the kind of knowledge that the reader of a certain textbook will collect or require are already made. For instance, in the anthropological or ethnographic domain, the kind of knowledge required is deep, qualitative, and reasonably specific to an individual or small group of individuals; the premise being that understanding small groups will allow us to extrapolate out to larger populations. This means that the research methodologies which have evolved in this domain are specifically keyed to produce this kind of research, and the experimental design is manipulated to use this kind of research methodologies. If you like, this can be thought of as a kind of ‘domain–group–think’ whereby a domain of enquiry thinks in a certain way and so the experimental design is often tailored to that way of thought (see [Figure: Ethnography?](#)). Indeed, in most instances a different design methodology is unlikely to be excepted if it diverges too much from the expected. In this way, we can see that it is very unlikely that an experimental methodology, such as is found in cognitive psychology, would be used as part of an anthropological study even though some of the resultant information may be useful.

17.3 Commissioning Constraints

For the most part, they are two different kinds of evaluation which are available to the UX specialist; these are the formative, and summative evaluation. As we have already seen, formative evaluations occur before a software artefact is developed or

a hypothesis is tested; and summative evaluations occur after a software artefact / intervention has been created. These two different kinds of evaluation enable us to do different kinds of UX work. Firstly, a formative evaluation allows us to understand the problem domain, test our understanding of that problem domain, guide our proposed hypotheses, and, therefore, the development of our software. In UX, and within the greater human-computer interaction domain, this formative aspect is sometimes not required. This is because the summative aspect can be applied to a software artefact to enable us to decide if it meets a pre-existing specification ([remember ‘Single Group, Post Test’](#)).

In the most basic case, we can assume that our domain understanding of the user experience, in general, will enable us to create a set of summative evaluations which will allow us to decide how well the software meets current best practice. In this case, we cannot run a true experiment because there is no way to compare the improvements which are gained between pre-test and post-tests of the user experience. Besides, this kind of testing does not enable us to say anything more about human behaviour, only that the participant evaluations showed that our software was to some degree beneficial; therefore complying with various human factors specification and testing regimes. In the second case, the summative evaluation can comprise two parts: the first being a pre-test and the second being a post-test in this way we can conform to the true experiment and also gauge the improvement in user experience by measuring the before modification, and after modification, variables. In reality, an experienced UX specialist will understand the real-world limitations of evaluating software in the wild, and in this case, build in a pre-test component as part of the formative evaluation such that hybrid methods are used; maybe an interview or survey followed by a small UX experiment within a laboratory setting: this last: would serve as the pre-test component of the summative experiment.

Textbook approaches to experimental design, and the choice of methodologies, often rely solely on selecting the best methodology for the job at hand. That methodology, as we have seen, assumes a certain knowledge of the application domain, with specifically favoured methodologies becoming popular in different domains. In reality, and certainly within the UX domain, things are not so straightforward. The real-world inflicts many constraints at this point and so unless the design is for academic work, which you have full control over, aspects of the design will be beyond our control. I have found that our lab (commercial user experience services) are often approached by third-party organisations which do not have the usability, and human testing experience that we possess. In this case, there is often no design requirement to use a certain research methodology, but there are limitations of time and budget allocated for the specific work. In this case, we have come to define these commissioning constraints as:

Just-In-Time Constraint. Often commissioners tend to leave aspects of the project planning which they know least about until the end, preferring not to go too deeply into this area because they become worried about their lack of understanding. Therefore, there is a tendency for the UX specialist to be approached just in time for the user testing to begin. This puts incredible pressure on the UX'er to design and instigate testing on a very short notice.

Too-Little-Time Constraint. Extending the ‘Just-In-Time Constraint’, the ‘Too-Little-Time Constraint’ suggests that in most cases, ‘user evaluation’ is all that is included within the project plan and the finer points of understanding how this testing should

proceed are neither understood all considered further. In this case, there has not been enough time allocated for post tests or nonlinear aspects of the work which could be run in parallel with other aspects of the project plan. As enough time has not been devoted to these aspects within the plan, testing is likely to fail or at best be very rushed.

Non-Specialist Constraint. There are additional variations of the two constraints previously listed; being those which include a very badly planned user evaluation, usually by a non-specialist, whereby because the planning stage is inflexible to higher-level management a none-scientific experimental evaluation looks like it needs to be considered.

Inadequately Funded Constraint. This constraint comes to the fore when user evaluation seems to be planned reasonably well, but the amount of funding required to accomplish it and participant payment within that funding model are not adequately considered.

Pre-Supposed Outcome Constraint. Planning in the real world often focuses on the achievement of the desired outcome. Indeed, most project planning and business planning revolve around achieving the correct outcome at each stage of the process. Therefore, a manager or engineer planning a piece of work will have the desired outcome in mind, which in reality will not just be a desire, but be a project requirement. In this case, there is an enormous amount of pressure on the UX specialist to deliver this requirement, support by the testing or not.

Implicit Overrun Constraint. Even when there has been enough funding and time allocated to an unplanned user evaluation, it is often accepted that it will be the final stage of the development, and, therefore, the plan is seen to fit into a linear dimension as one of the last parts of the project development. In this case, overruns in other aspects of the work, often considered to be more important than the UX testing, tend to eat into the time available for the linear plan.

Due Diligence Constraint. Normally as a function of the 'Inadequately Funded Constraint' and in combination with the 'Too-Little-Time Constraint' we see that in some cases the usability study is only an afterthought such that the software engineers can demonstrate due diligence; and in reality expect no more than a cursory ratification of the development. Here we can see that this is very difficult for the web ergonomics specialist a handle because acceptance often implies a Pre-Supposed Outcome. How this is handled, and whether the contract is undertaken by the specialist is a matter of personal choice

UnConstrained Experiment. Finally, there is that rare occasion whereby the work of the UX specialist is understood and respected by the project planner, and indeed, be commissioner of the development. This is often the most rewarding type of work undertaken and in many cases occurs in academia or in research and development work for large corporations whereby UX, human factors, and human behaviour are the central focus of the scientific outcomes which are expected as opposed to a standard software development. We describe this, in the Web Ergonomics Laboratory, as the unconstrained experiment. Here we can bring a well-designed experiment using multiple methodologies and hybrid techniques to bear upon a problem domain using a full range of tools instruments techniques and hypotheses.

These commissioning constraints not only have a direct bearing on the decision as to which UX methodologies to use but also how the experimental design is conducted.

17.4 Information Requirements

Organisations, especially those concerned with research and development, be they research, academic, or commercial often approach us with a set of information requirements. These requirements are mainly concerned with understanding aspects of a specific UX domain related directly to the organisation focus. However, these requirements can often be distilled into single line statements about the requirements that the information must fulfil. While the following statements are not an exhaustive set, they are particularly indicative of the real-world objectives which we have encountered:

“What is the general feeling for an area, give us the extent and scope of the problems we face; This is the extent and scope of the domain (as we understand it) confirm this is the case and come up with some ideas as to a solution or more formal testing; This is a specification, test our software to make sure it fulfils the user requirements of this specification; We know the extent and scope of the area, and indeed, have some ideas for solutions to the problems we face, do some informal testing so we are able to quantify which solution will be best; ‘We believe our software fulfils these three well-defined goals, test to make sure we are correct; We feel these problems exist in the domain, rigourously test our hypothesis to find out if we are correct; The system used to be like this before our changes, now it has been upgraded, now test to make sure there is a quantifiable benefit; and This was the state of our knowledge beforehand, we have made these predictions, now must test whether our predictions are correct in a rigourous and quantifiable manner.”

In this case, we can see that different information requirements require different ways of eliciting this information. Certainly, within our laboratory, the information requirements of the study are normally limited by the requirements and constraints of the commissioning organisations. In our case, this is normally research councils, and therefore, we have direct control of the planning process. In this way, most of our work is unconstrained and uses multiple hybrid techniques to gather as rich a dataset as is possible. However, we have also noticed that when commissioned to work for other organisation, we often progress from a very resource light to a very resource heavy, scenario based on the kind of information we feel is required. This normally means that we have five different levels of information elicitation:

Flavour Elicitation. This is the least resource hungry information elicitation process. In this case, we look to understand the related work within the area including standards and best practices; even the UX specialist cannot retain all aspects of accessibility, usability, and ergonomics. We then plan some exploratory studies often based around informal interviews and discussions to scope the extent of the problems we are working on and suggest possible solutions. This work gives a flavour for the problem at hand and allows us to better understand the extent of the domain and where our work should be placed within it.

Ideas Elicitation. Here we wish to firm up our understanding of the scope and extent of the work, and generate some ideas for how we could proceed into both a formal development and some more rigorous work; to uncover just exactly what problems exist. In some cases ideas and flavour can be combined, however for most work generating ideas

the specialist would normally have a very strong feeling for the domain, and, therefore, the flavour requirement may not be necessary.

Informal Elicitation. Informal testing enables us to understand if our initial ideas have some traction by allowing us to dynamically create assumptions, predictions, and hypotheses and test these on-the-fly in an informal setting. This setting is normally based around a reasonably rigorous interview, but the number of participants may be reasonably limited and maybe solicited without much reference to demographics or correct user sampling techniques. In this case, we are trying to decide if it is worth increasing our resources and placing more formal studies within the domain which will require much more commitment to the task at hand.

Formal Elicitation. Here we have decided that there is indeed some interesting information to be learnt and have decided that we should devote more resources to finding out and isolating this information. Here we use the previous three kinds of information elicitation to enable us to create and test a set of hypotheses. This conventionally involves formal surveying techniques and questionnaires such that numerical statistical analysis can be achieved. Eventually, however, if these hypotheses become reasonably concrete the testing may be accomplished within a laboratory setting.

Laboratory Elicitation. This is the most resource hungry form of information elicitation and can involve both pre and post-tests as a mature experiment. In some limited scenarios post-testing is sufficient although this means that it cannot be considered a true experiment.

By understanding the information requirements of the commissioning organisation we can create a flexible series of experiments which will enable us to fulfil those requirements; as long as we factor in both the design limitations and the available skills of the researchers.

17.5 Limitations

As well as considering the commissioning and information requirements, many more technical limitations within the UX design must be considered before a realistic programme of work can be embarked upon. In general, there are three factors which limit the effectiveness of a programme of work, these being time, participation, and resources.

17.5.1 Time

Time is a critical factor, not just within the planning and project management of the software process, but also when planning the finer details of the application of the UX methodologies. In reality, you will mostly have a finite and limited time for the entire evaluation. This time limitation can be further sub-divided based on the kinds of tasks that are required; the amount of writing which is undertaken; the amount of reporting and administrative components which make up the design; and the amount of interaction with other stakeholders which will always require more time within any methodology or design phase.

Additional time must also be planned for designs which involve iterative testing of software or other kinds of solutions. Iterative testing and development are one of the most critical aspects of applied empirical work because the results of the user study are a driver for the next iteration of development; such that the final post-test should demonstrate a significant improvement in the ergonomic aspects of the developed software. However, time is required for iterative development, and this means that time allocated for development cannot be used for the testing cycle, likewise development is stalled until the interim report from the iterative evaluation is complete.

The only way to mitigate these factors is to divide the user-facing aspects of the development into sections, testing each section while interweaving development on a different section together such that an efficient parallelisation of the iterative evaluation and development cycle can be accomplished.

17.5.2 Participants

Soliciting participants are one of the most intractable limitations in any user-facing evaluations. This is especially the case for niche user groups such as users with disabilities, older users, younger users, and users from certain ethnic or social backgrounds. Recruitment is often difficult and time-consuming, indeed, at the Web Ergonomics Laboratory we expect that the experimental phase will not proceed in orderly chunks but will be distributed over the space of a project based on the difficulties in soliciting users for studies.

There are some things which can mitigate these difficulties, such as bursaries and honorariums, or rewards such as vouchers, chocolates, or being placed into a lottery to win a prize; however, on their own these are often not enough. In reality, the UX'er should form pre-existing relationships, which are strong and reciprocal, with the community organisations which represent these participant groups. Having endorsement from a trusted third party, such as a community or advocacy group, enables the work to be undertaken in a far more efficient manner. Without this endorsement, it is often difficult to recruit participants in the first place, and then entice them to return if further testing is required. In any event, you should make sure your evaluation design includes long periods for recruiting participants.

In the real world this recruitment often occurs as empirical aspects of the research proceeds; in this case participant recruitment and UX, evaluation occurs in parallel as opposed to separate well-delineated chunks. This normally means that participant numbers are limited. It is worth remembering that in user experience we do not have the luxury of participant numbers such as those within the social sciences, clinical medicine, or epidemiology; to name but three. In reality, human factors work is often confined to small user groups and at my laboratory, we would consider ourselves lucky to recruit thirty general participants, or between five and ten from a niche user group. This level of participant recruitment will affect the UX methodologies which you will be able to choose; it also means that your data becomes very valuable and so should be collected in a very rigorous manner, and stored openly. If your data is stored openly, and the data of other UX specialists is likewise openly stored, then data reuse, a perfectly except a way of – certainly isolating problems for a post-test or formative evaluation – can occur.

17.5.3 Resources

The final piece of the puzzle is the availability of resources. In this case, we mean both funding arrangements for participant reimbursement, as well as funding for travel to and from site visits for the UX'ers, in addition to the number of UX specialists available. This last part is critical, in that it defines the kind of evaluation work and the rigour that can be accomplished.

With only one UX'er a double-blind or triple-blind trial is not feasible. When we talk about a double-blind trial we mean that the researcher who has designed the evaluation procedure and understands the UX outcomes which would be desirable for the programme of work is not the same person who enacts that methodology with participants. In this way, the desire for a certain specific outcome being implicitly transmitted to participants can be removed. But this means that more resources are required, and these may not be available.

Obviously, the most qualitative piece of work the less likely it is that a double-blind trial will be required. This is because interpretation and an understanding of the domain are key for these kinds of methods. However, the more quantitative or experimental the trial is, the more likely it is that a double-blind or triple-blind method should be used. In the case of a triple-blind methodology the specialist who has designed the work is different to the UXer, who is conducting the data collection, and both are different to the specialist who is performing an analysis of the data, once collected. This means that the statistical analysis of the data collected is not biased towards the desired outcome. However, for human factors work triple-blind trials are often inadvisable because the analysis of data collected from applied work is often nuanced and some level of domain understanding is required.

We can, therefore, see that these three kinds of limitations directly affect the way that research should be planned. It is only with a knowledge and understanding of the time required, the difficulty of participant recruitment, and the constraints on resources exhibited by a particular research team that an experimental design can be created to exhibit a successful experimental outcome.

17.6 Available Skills

The available skills of the specialist are often overlooked, both at the evaluation design stage and when the methodologies are chosen or are about to be undertaken. It is often assumed that the UX'er will have a basic set of research skills which will enable them to, in some way, perform all research tasks. While this may be true in some rare cases, the experience which a UX'er brings to a certain domain of investigation should not be overlooked. Skills, along with the experience of the specialist, are a key factor in understanding the amount of time that a piece of work will take. If the specialist is highly conversant and expert within a certain methodology, then we can assume that the process will be very much faster. However, if the specialist has to learn the skills on-the-job we can see that this may extend the time required for the methodology to be accomplished successfully. Indeed, inexperienced UX'ers, learning new methodologies on-the-job, often means that there is a significant amount of

backtracking and redoing of work; as experience is increasingly gained. In this case, we should not think that it is impossible for a UX'er to learn new techniques. However, we should make allowances for the mistakes which will inevitably be made and for the rectification of those mistakes.

17.6.1 Instrumentation

Understanding instrumentation and its availability to the researcher is key when designing experiments. Our experiences, at the Web Ergonomics Laboratory, suggest that instrumentation is often a useful addition to the evaluation design, but it should not obscure the abilities of the UX'er to come to a suitable and accurate outcome if the instrumentation is not available. For instance, when we started the laboratory ten years ago, all our work was conducted with a pen, paper, and questionnaires. This level of instrumentation enabled us to observe users performing tasks in both naturalistic settings and quantitative and qualitative ones. Indeed, in some cases, it also enabled us to understand user behaviour in laboratory settings. However, we found that a screen recorder with the ability to log information and facilities to accurately measure users' behaviour – and record it – made this job increasingly simple. We then acquired video and audio facilities to record all user movements and activity within a laboratory setting; and augmented this with the ability to eye-track users, thereby enabling us to make some connection between the cognition of the user and their interaction with the interface. Finally, many biofeedback devices enabled us to monitor different aspects of the user's state while interacting with specific interfaces.

In this way, instrumentation can be seen as useful but by no means a bar to accomplishing good UX work. However, in all cases, it took us some evaluation attempts to fully understand and become proficient in the instruments that we were using. In this case, the UX'er, who wishes to use more complex instruments, should understand that this is possible and often advisable but that the data collected from these instruments will initially be inaccurate, and the work will need repeating as the UX'er becomes more familiar with the device.

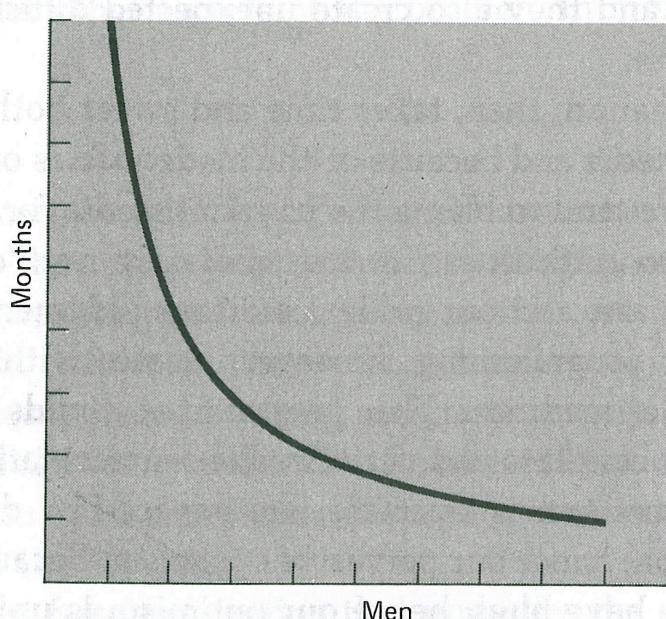


Figure: Perfectly Partitionable Task. Time Versus Number of Workers, Perfectly Partitionable Task. —Image Credit: The Mythical Man Month.

17.6.2 Collection

Data collection is another aspect of the evaluation methodology which may cause problems. As we have previously seen, different UX methodologies pre-suppose a different training because they come from specific single disciplinary domains. Therefore, understanding the data collection necessities one UX methodology, which may not be second nature to a UX'er, who is a highly trained expert in a methodology from a different domain. For instance it is often initially difficult for qualitative investigators from, say, the social sciences, to fully understand the rationale for data collection and the qualitative aspects of data collection from, say, anthropological ethnography. Likewise, in the UX domain we see a very high level of interdisciplinary evaluation methodologies being used, but because the training of the UX'ers has often occurred within a single domain it will take time for their mindset to alter such that they understand the new methodologies and the data collection requirements inherent within those methodologies.

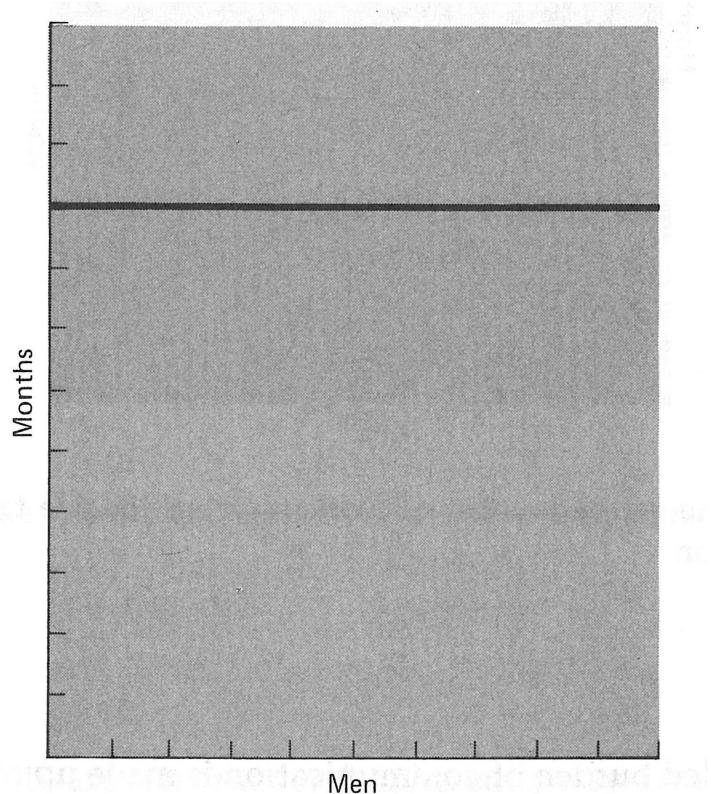


Figure: Unpartitionable Task. Time Versus Number of Workers, Unpartitionable Task. –Image Credit: The Mythical Man Month.

Because the training of the UX'ers has often occurred within a single domain it will take time for their mindset to alter such that they understand the new methodologies and the data collection requirements inherent within those methodologies.

17.6.3 Analysis

Finally, data analysis is one of the key and thorny issues which surrounds UX work. In many cases, the quality of the data analysis is key to understanding the outcomes of the work, whether that data be collected via complex instrumentation or simple pad, pen and observation. In either case, understanding how to analyse the data and what that analysis may tell us is key to an accurate outcome.

In most cases we can understand the qualitative aspects although for a more rigorous quantitative outcome we should also transcribe each script such that we can understand repeating concepts and terms which arise between participants pointing to a more general understanding of problems or solutions within the specific evaluation. However, the key aspect to analysis is statistics. In most cases statistical analysis used incorrectly can prove to be a tool for inaccuracy, miss-understandings, and miss-information. However, used correctly, statistics can tell a significant amount about the UX domain. Statistical analysis can run from the quite simplistic to the quite compli-

cated, however, as soon as the term ‘statistics’ is mentioned **most researchers become understandably wary**. This is mainly because the possibility for the introduction of errors are increased and the concepts which surround statistical analysis are perceived as being complex; and in some cases a ‘dark-art’.

As we have seen, basic statistical tasks are nothing to worry about; these range from statistical test which allows us to describe the internal consistency of the data themselves without making any generalisations over a wider group of users; to inferential statistics which allow us to extrapolate some meaning from a certain set of data and extend it over a larger population than the sample from which it was extracted. Choosing the right statistics for the job is key, and once that choice is made the automated software is sophisticated enough to do with the rest. As long as the UX'er understands those statistical outcomes there is nothing to fear and an accurate analysis, with a high degree of certainty, can be achieved.

We can see then, that if the evaluation designer does not take into account the experiences of each UX'er within the team, and their ability to contribute to different aspects of the skills required, then the evaluation can become badly flawed in terms of time and scientific outcome. By understanding the possible risks at the design phase, and by mitigating those risks usually by increasing the time allowed for the work, and in some cases adding a training element to the design, successful and accurate UX evaluation outcomes can be achieved.

17.7 Optimism

We can see then, that expecting uncertainty and variability within the plan is not only advisable but the key to maintaining good user experience outcomes. Even if the project plan and the other members of the team do not take this viewpoint, you – as a user experience professional – should. Indeed, there are many salutary lessons we can learn from other business and engineering domains. However, some of the best advice comes from Frederick Brooks' experience on the IBM 360, which he generalises in his book ‘The Mythical Man Month’ [**Brooks, 1995**]; and I believe centres mainly around optimism.

There are of course many opposing views to the generalisations Brooks sets out, as well as changes by the author after 20 years of retrospective work; however, in my opinion, the bones of the work are still valid and in some cases the retrospective reviews or incorrect¹.

This said, the Mythical Man Month is most famous for stating Brooks' Law which, as he says, is oversimplified but states that *‘Adding manpower to a late software project makes it later’*. However, while this is useful in the user experience domain especially when we are trying to address commissioning constraints (for constraints, read errors) such as the ‘Just-In-Time Constraint’ and the ‘Too-Little-Time Constraint’ it should not be our main focus. One key aspect that most commentators or experts failed to address

¹For instance, Brooks initially states that code should not be hidden from other developers on the same project. He suggests openness and open scrutiny increases code quality; thereby supporting the work of Harlan Mills and undermining the work of David Parnas. After 20 years he cites Object Oriented Programming and the concept of abstraction for why he thinks he was wrong – 20 years ago – and why Parnas was right. I'd disagree, or, at least, be more specific in that abstraction is good in use, but openness is good in an audit. We can see that open source software suffers fewer bugs and has increased code quality, mainly because of the oversight (our reputation to our peers - maybe) of other coders.

– and which Fred Brooks gets right – is that computer scientists, developers, software engineers, and user experience specialist related to these technical domains are mostly optimistic.

This optimism is one of their key strengths; they expect to code better, think most problems are soluble with enough technology and code, create systems which are often useful, and focus on bugs which they believe can all be removed. But this optimism can often be a hindrance when it comes to planning a project and understanding when aspects of the project will be completed; and to what level of maturity those answers, generated from evaluations, are at. In effect we want to believe the system is good because we're optimistic; even when the system is bad, or the process will over run, or all bugs will not be found, and not all problems have solutions amenable to code / device development.

Let's consider in a little more detail the central tenant of the Mythical Man Month, in that the addition of workers often increases the length of time the project takes. This is important concerning user experience because we can understand that adding human factors specialist may not be the best plan of action if it looks like we're going to be late. In some cases, there can be positive benefits such as those surrounding tasks which are perfectly partition-able. In the context of user experience we can think of this as running multiple and parallel data collection activities, interviews, surveys or laboratory work in which more experimenters will be able to gather more data in parallel (see [Figure: Perfectly Partitionable Task](#)). However, the problem comes when unpartitionable tasks occur. Here we may think of tasks such as the data analysis, statistical work, in some cases report writing, etc. whereby the number of workers will have no effect because the task is unpartitionable and reasonably linear (see [Figure: Unpartitionable Task](#)). We also need to think about whether there will be a training period required for each new user experience professional. This means that the project will become even later while this training period occurs because the training period will take the resources of both the experienced UX specialists who are providing the training, as well as those new workers who are receiving training. This means that even though a full project may include perfectly partition-able tasks and perfectly unpartitionable tasks – and the full spectrum in between – it will often be the case that the things that are partition-able do not outweigh those that are unpartitionable; and further, we do not know the amount of additional training which will be required to add new workers, which may outstrip the benefits of running the partition-able tasks themselves.

Another effect, of what I think is optimism, is that described as the 'Second System Effect'. Simply this means that the designer progresses slowly with the first system being careful because they do not know exactly what is required, they are careful to stick to the specifications that they have created. Even as embellishments occur to the designers and developers as they progress, these are stored away for the next version. However, when it comes to creating the second version the developers are buoyed by their expertise and mastery of the first system and try to cram in as many of the flourishes and embellishments from the first system as is possible. In effect making this makes the second system a terrible hodgepodge of work, and in our case a terrible user interface (or interactive design). Brooks tells us that this second system is the most dangerous one to be building and gives us an example the IBM 7090 the 'Stretch' computer, indeed one of the developers of the 'Stretch' tells us that:

"I get the impression that stretch is in some way the end of one line of development. Like some early computer programs it is immensely ingenious, immensely complicated, and extremely effective, but somehow at the same time crude, wasteful, and inelegant, and one feels that there must be a better way of doing things." [Strachey, 1962].

This fits well into our understanding of user experience, as the author suggests the system is extremely effective, however, it is also inelegant this directly contradicts our [desire as previously discussed](#)). Further, how [does this equate to the Xerox Star computer?](#) In reality, we can see that Star was developed over many iterations slowly and steadily with each iteration being used internally by the Xerox Parc infrastructure. It seems that this may be a way of mitigating the danger of adding unnecessary embellishments to a system through versions; indeed negating the second system effect may just be a simple case of eating-your-own-dog-food.

Optimism also makes predictions, both of the work required and the time it will take, inaccurate. This prediction inaccuracy can mean that what seems like short delays or inconsistencies in the work can expand into very major delays (see [Figure: Programming Rates](#)). Further, these delays can impact other aspects of the project such that the cumulative effect is far more difficult to address down the singular effect of the delay itself (see [Figure: Debugging Rates](#)). There is an old adage which asks 'how does a project become one year late?', The answer, of course, is 'one day at a time'. It is up to you to understand that our predictions are often optimistic and that we need to add in extra time, if at all possible because these predictions will go awry.

One of the most important lessons to learn is that of a throwaway project. This means that you should plan to throw away the first version of anything that you create, be it a user interface design or an engineered interaction. The first version often enables an understanding of the system and aspects of the system development, but it should not be confused with a real live deployable user experience. If you do not plan to throw away one of these systems you will have to do it in the end anyhow, the only question is whether you plan in advance to throw away the first version of whether this comes upon you as a shock after you've

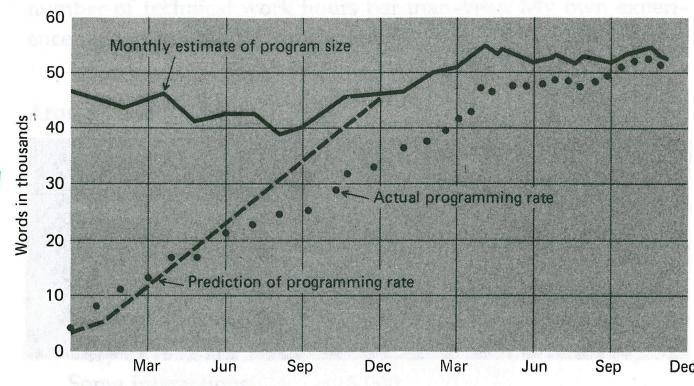


Figure: Programming Rates. ESS Predicted and Actual Programming Rates. —Image Credit: The Mythical Man Month.

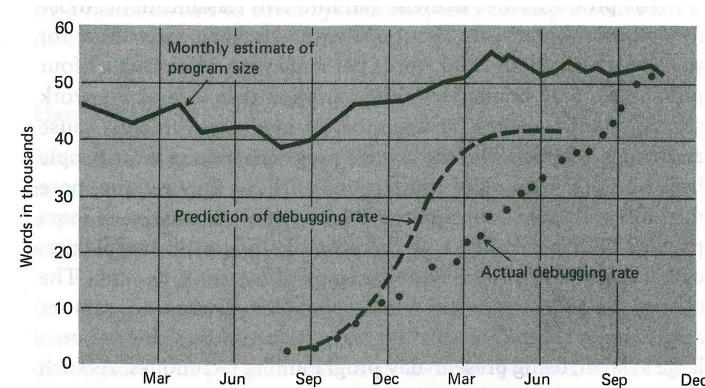


Figure: Debugging Rates. ESS Predicted and Actual Debugging Rates. —Image Credit: The Mythical Man Month.

If you do not plan to throw away one of these systems you will have to do it in the end anyhow, the only question is whether you plan in advance to throw away the first version of whether this comes upon you as a shock after you've

delivered it to customers.

Now that you've recognise that the first system is the pilot system, and a pilot system will be thrown away, so you may as well plan-it in the first place, you can start to become more accustomed to the fact that changes occur, and indeed, that the only consistency is change itself. We can see that [agile methods](#), and to some degree, the iterative design cycle makes accommodation for these constantly changing factors. But you should also realise that even though the project planner or manager may have agreed on a specification, that specification is unlikely to be the one delivered. Unfortunately, most of the interaction with customers and users will fall with you and so as the user experience specialist you need to understand that users change their minds when faced with both the system they are evaluating and in some cases the awakening concept that they can now add features that had not previously thought possible.

In this case you must think that change will occur at all levels: changes to the system, changing user requirements, changing technologies, and the changing views of the software engineers; these changes you must just take as a given. System and specification changes will occur; you must understand this, and plan for it while also enacting small steps in the UX design and the associated user interactions.

Since Brooks first suggested the concept of the throw one away project (which I subscribe to) new thought and developing techniques have suggested that this isn't the case. That the throw-one-away concept was predicated on the waterfall model, but now we have incremental and iterative refinement as well as agile methods. While we might be able to understand the logic of this, in real life it seems the idea of throwing a version way is still appropriate and is very useful specifically for time planning purposes. Indeed, even if we look at incremental build and agile SCRUM methods with inherent sprints, at the user experience level and (possibly because this is so close to the ebb and flow of user need, or perception) we still end up performing the same formative, summative, and development cycles again and again throwing away the older versions as the users suddenly realise that their concept of what was required is limited by their understanding of what is possible. As soon as they experience the pilot system, what is possible in their minds suddenly increases, and so this 'increased' system is now required².

One thing that has come out of the work of David Parnas, and in some cases may be [transferred back](#), is the ability to think of a software project as a family of related products; Parnas suggests this method anticipates enhanced sideways extensions and versioning.

It also seems that this is useful for the separation of concerns, and decoupling of the interface from the program logic. I would also suggest that this kind of view means that the individual applications become highly partitioned enabling user experience people to work on different interfaces for each member of this family of related projects; one failure of an individual does not directly affect the functioning of the others.

I would further suggest that moving away from the versioning tree and seeing each family member as enacting certain functionality which is specific to a particular user

²Further this way of thinking of disposable prototypes does not contradict the idea of the incremental build, rapid prototyping, or the build-every-night approach. These items can still occur because the build does not imply a finished product but just the absence of bugs in the program logic (No compiler errors). The misconception of user requirements and the need for full system changes – at least at the interface level – are still very present; indeed, the ebb and flow of the user dictates this is so.

enables us to enact openness, open APIs, and decoupled interfaces. We, therefore, move away from one single monolithic product, which is prone to failure and a logarithmic explosion of bugs (both within the program logic and at the interface level), to a set of smaller projects more tailored to users need with the data flow occurring between individuals of the product family.

In this way, we can see that software artefacts are more tailored to the needs of the user, and resemble the user roles and stakeholders. This division enables much more personalisation of the specific individual artefact, as opposed to a massive rebuild, which may be required of a monolithic system.

17.8 Summary

Goethe says ‘What we do not understand we do not possess.’, while Crabbe says ‘O give me commentators plain, who with no deep researchers vex the brain.’

In the real world of user experience, the first is relevant to the UX specialist while the second is relevant to the user of the system. As user experience professionals if we do not understand the users or the system in total then we do not possess an ability to affect change within that system or within the users who interact with it. Without this deep understanding there is no ability for us to create accurate software artefacts or models of the users.

On the other hand, the user requires the system to speak to them in a language they understand plainly, the need for a deep understanding of how the system works and how to use it should be should not be required. In general, if the design of the system suggests that the system is complicated because it is for experts – and it, therefore, can’t be simplified, or made more easy-to-use – then I would be sceptical of the system designers.

The concept of the system needing expert users is often because the interface engineering or software engineering has been performed shoddily. There are some cases – and as few exceptions which may prove this rule – which exist but instead of being optimistic about a statement that I’ve heard so many times, be pessimistic. If the system cannot be used by a novice, it is most likely this is because the system is at fault and not the novice.

It is often said that there is no silver bullet, and this is exactly true when it comes to real-world user experience development. There is no system that will solve all your problems, there is no book which will present all possible options, there is no focus group or user base which you can rely on to give you accurate information, software engineers may not see the importance of the interface engineering aspects, stakeholders may have conflicting requirements, and project managers may give you no time for the work to be undertaken [**Streiner and Sidani, 2010**]. As the UX professional it is up to you to expect this, and therefore, take account of all these factors, your optimism, and the inane nature of the system which changes minute by minute (as does the world in which we live, the system being simply a reflection of this in microcosm); and think back to Ted Nelson.

While there might be no silver bullet (indeed the silver bullet may take away some of the challenge, and, therefore, our fun), the best advice I have is to always keep in mind

the pessimistic view of an optimistic visionary “*most people are fools, most authority is malignant, God does not exist, and everything is wrong.*”.

17.8.1 Optional Further Reading

- [M. C. Alkin]. Evaluation essentials from A to Z. Guilford Press, New York, 2011.
- [F. P. Brooks]. The mythical man-month : essays on software engineering. Addison-Wesley Pub. Co., Reading, Mass., 1995.
- [J. A. Gliner], G. A. Morgan, and N. L. Leech. Research methods in applied settings : an integrated approach to design and analysis. Routledge, New York; London, 2009.
- [D. L. Streiner] and S. Sidani. When research goes off the rails : why it happens and what you can do about it. Guilford Press, New York, 2010.
- [W. P. Vogt], D. C. Gardner, and L. M. Haeffele. When to use what research design. Guilford Press, New York, 2012.

17.8.2 Self Assessment Questions

Try these without reference to the text:

1. You are suffering from the ‘Too–Little–Time’ constraint and need to get a formative evaluation with 20 people (employees of the factory commissioning your new production line software) underway very quickly. At this stage, you only need qualitative results – how would you go about getting this information in the fastest time possible, and why would you be cautious?
2. What are commissioning constraints?
3. What is real world work limited by?
4. Why is optimism often a bad mindset to have when it comes to planning UX work?
5. Describe the ‘Second System Effect’?

18. Final Thoughts

‘How do I explain what I do at a party? The short version is that I say I humanize technology.’

– Fred Beecher, Director of UX, The Nerdery

As we have seen, UX refers to the overall experience that a user has when interacting with a product, system, or service. It encompasses all aspects of the user’s interaction with the product, including their emotional response, ease of use, and satisfaction with the product or service.

UX design is the process of designing products, systems, or services with the user’s experience in mind. It involves understanding the user’s needs and preferences, and designing products that are intuitive, easy to use, and provide a positive experience. It’s also associated with visual design as opposed to the broader user experience.

Good UX can lead to increased user satisfaction, improved engagement, and higher conversion rates. It is a crucial consideration for any product or service, as it directly impacts the user’s perception of the product and can determine whether or not they continue to use it.

Overall, user experience is a critical consideration for any product or service, and UX design plays a crucial role in ensuring that products are effective, engaging, and user-friendly.

18.1 Design

Designing the User Experience (UX) refers to the process of creating products, systems, or services with a focus on the user’s experience. It is a user-centric approach to design that involves understanding the user’s needs, preferences, and behaviors, and designing products that meet those needs.

The UX design process typically involves several stages, including user research, information architecture, interaction design, visual design, and usability testing. User research involves gathering information about the user’s needs and preferences through interviews, surveys, and other methods. Information architecture involves organizing and structuring content and functionality in a way that is intuitive and easy to navigate. Interaction design involves designing the way that users interact with a product, such as through buttons, menus, and other interface elements. Visual design involves designing the overall look and feel of the product, including colors, typography, and other visual elements. Usability testing involves testing the product with users to identify any issues or areas for improvement.

Effective UX design can lead to improved user satisfaction, increased engagement, and higher conversion rates. It is a critical consideration for any product or service, as it

directly impacts the user's perception of the product and can determine whether or not they continue to use it.

Overall, designing the user experience is a user-centric approach to design that involves understanding the user's needs and preferences and designing products that meet those needs. It is a critical consideration for any product or service, and effective UX design can lead to improved user satisfaction and engagement.



Figure: Oblivion Touch Surfaces. Oblivion Natural User Interfaces (NUIs) —Image Credit: Universal Pictures

18.2 Development

UX implementation refers to the process of translating the user experience design into a functional digital product. It involves using various tools, technologies, and development practices to bring the user interface (UI) to life while ensuring that it aligns with the design vision and meets the needs of the target users. Here are some key aspects of UX implementation:

1. **Front-end Development:** Front-end development is responsible for implementing the UI design using web technologies such as HTML, CSS, and JavaScript. It involves coding the visual elements, layout, and interactions to create an interface that matches the design specifications.
2. **Responsiveness:** With the growing use of mobile devices, it is crucial to ensure that the user interface is responsive and adapts to different screen sizes and resolutions. Responsive design techniques, such as media queries and fluid layouts, are employed to create a seamless experience across devices.
3. **Accessibility:** Accessibility is an important consideration in UX implementation. Developers need to follow web accessibility guidelines, such as the Web Content Accessibility Guidelines (WCAG), to ensure that the product is usable by people with disabilities. This includes providing alternative text for images, using proper semantic markup, and implementing keyboard navigation support.

4. Interaction Design: Implementing the interactive elements of the user interface involves using JavaScript or other scripting languages. This includes creating animations, transitions, form validations, and other dynamic behaviors that enhance the user experience.
5. Integration with Back-end Systems: In many cases, the user interface needs to communicate with back-end systems or APIs to retrieve and display data or perform certain actions. Developers need to integrate the UI with the appropriate back-end technologies, such as databases, APIs, or server-side scripting languages, to enable these interactions.
6. Performance Optimization: UX implementation also involves optimizing the performance of the product. This includes techniques such as optimizing image sizes, minimizing file sizes, and leveraging caching mechanisms to ensure fast loading times and smooth interactions.
7. Cross-Browser and Cross-Device Compatibility: The user interface needs to work consistently across different web browsers and devices. Developers perform testing and apply necessary fixes to ensure compatibility and a consistent experience for users.
8. Continuous Integration and Deployment: UX implementation often follows agile development practices, where continuous integration and deployment processes are employed. This allows for regular updates, bug fixes, and improvements to be deployed efficiently, ensuring that the product remains up-to-date and responsive to user needs.

Throughout the implementation process, collaboration and communication between designers and developers are vital to maintain the integrity of the user experience and address any technical challenges that may arise. It is also common to conduct usability testing during and after implementation to identify any usability issues and make necessary refinements.

18.3 Validation

Validating the user experience is a crucial process in the field of product development and design. It involves gathering feedback and data from users to assess the effectiveness, usability, and satisfaction of a product or service. By validating the user experience, businesses can ensure that their offerings meet the needs and expectations of their target audience, resulting in improved customer satisfaction, engagement, and loyalty.

The process of validating the user experience typically involves various methods and techniques, including user testing, surveys, interviews, and analytics. Key aspects centre around:

1. User Testing: User testing involves observing and analyzing how real users interact with a product or prototype. It can be conducted in a controlled environment or remotely, depending on the nature of the product. By observing users' actions, listening to their feedback, and analyzing their behavior, businesses can identify usability issues, uncover pain points, and gain valuable insights for improvement.

2. Surveys and Interviews: Surveys and interviews provide an opportunity to collect feedback from a larger group of users. Structured questionnaires and interviews can help gather specific insights about user preferences, satisfaction levels, and areas for improvement. These methods can be conducted online or in person, depending on the target audience.
3. Analytics and Data Analysis: Leveraging analytics tools and data analysis allows businesses to gain quantitative insights into user behavior. By tracking user interactions, such as clicks, navigation paths, and time spent on different sections, businesses can identify patterns, detect bottlenecks, and measure the success of specific features or design changes.
4. Iterative Design: Validating the user experience is an iterative process. Feedback and insights gathered from users should be used to refine and improve the product or service continuously. This iterative approach allows businesses to address user concerns, enhance usability, and align the user experience with evolving user needs and expectations.
5. A/B Testing: A/B testing involves comparing two or more versions of a product or feature to determine which one performs better in terms of user experience metrics. By randomly assigning different users to each version and analyzing the results, businesses can make data-driven decisions about design choices, content placement, and feature implementation.

The ultimate goal of validating the user experience is to create products and services that are intuitive, enjoyable, and valuable to users. By actively involving users in the design and development process, businesses can make informed decisions, reduce the risk of costly mistakes, and deliver experiences that delight their target audience.

18.4 As Practically Applied

In real life, the user experience (UX) process typically involves several key steps that are followed to create and validate a successful user experience. But these steps are also prone to failure and chaos as opposed to the sanitised version that is taught in the textbooks. These steps broadly proceed as follows:

1. User Research: The UX process begins with user research, which involves gathering information about the target audience, their needs, behaviors, and preferences. This step may include conducting interviews, surveys, and observational studies to gain insights into user goals, pain points, and motivations.
2. User Personas: Based on the user research findings, user personas are created. Personas represent fictional characters that embody the characteristics and behaviors of different user types. They help the design team understand and empathize with the users, guiding design decisions throughout the process.
3. User Flows and Information Architecture: User flows and information architecture are created to define the structure and organization of the product or service. User flows outline the path users will take to accomplish their goals, while information architecture involves structuring and labeling content in a way that is intuitive and easy to navigate.

4. Wireframing and Prototyping: Wireframes and prototypes are low-fidelity representations of the product's interface and functionality. They are used to visualize and test different design concepts and interactions. Wireframes focus on layout and structure, while prototypes offer more interactivity and simulate the user experience.
5. Usability Testing: Usability testing involves observing real users as they interact with prototypes or the actual product. Test participants are given specific tasks to complete, while researchers observe their actions, listen to their feedback, and note any usability issues or areas for improvement. Usability testing helps uncover design flaws, improve usability, and validate design decisions.
6. Iterative Design and Feedback: Based on the insights gained from usability testing, the design team iterates on the design, making refinements and improvements. This iterative process involves incorporating user feedback, addressing usability issues, and continuously refining the user experience.
7. Visual Design: Once the structure and interactions are well-defined, the visual design phase begins. Visual design focuses on the aesthetics, branding, typography, color schemes, and overall visual appeal of the product. The visual design should align with the brand identity and enhance the overall user experience.
8. Development and Implementation: After the design phase, the development team takes the finalized designs and brings them to life, implementing the necessary functionalities and integrating the visual elements. Close collaboration between designers and developers is essential to ensure the design vision is accurately translated into the final product.
9. User Testing and Validation: Once the product is developed, it undergoes further testing and validation. This may include user acceptance testing, beta testing with a larger user base, or even conducting additional usability tests to ensure the product meets user expectations and performs as intended.
10. Post-Launch Evaluation: After the product is launched, the UX process doesn't end. Continuous evaluation and monitoring of user feedback, analytics, and performance metrics are crucial to identify areas for improvement and plan future enhancements or updates.

It's important to note that the UX process is not necessarily linear, and different projects may have variations in the order or intensity of these steps. However, the underlying principles of user-centered design, research, iterative development, and user validation remain consistent throughout the real-life UX process.

18.5 Final Thoughts

As we've seen, the UX domain is still very young and in the process of formation. But, it does bring together many already established areas within the human-computer interaction field. You may wish to think of UX as the practical application of research knowledge repurposed from other domains into the user-facing software engineering process. But by now you probably have your own view and concept of UX and UXD for that matter.

Your concept of UX is valid and that is one advantage of UX as a training domain in that it is a new, practical, cross-disciplinary subject. No one has yet trained on a bespoke UX

only degree programme, and so everyone has their own background and ‘slant’ to the area. There are many UX Industrial Departments/Companies, there are few UX courses. The combination of your technical Computer Science training coupled with this UX training will give you an advantage; a software focused UX professional.

Good luck in the future, and good luck in your continued UX work and study!

18.5.1 Optional Further Reading

- Accessible Player Experiences (APX)¹
- Everything is a Remix²
- Apple brand less ‘inspiring’, survey says³
- Being online aged 90 has made my old age less lonely. Others aren’t so lucky⁴
- Steve Whittaker⁵



The Encyclopedia of Human-Computer Interaction

The Encyclopedia of Human-Computer Interaction, is an online resource created by over 100 interaction experts and is a positive treasure trove of how to practically apply the tools and techniques presented in this text. I think the encyclopedia is definitive and I trust it more than other books. It was created by the [Interaction Design Foundation](#)⁶ and in 4,000+ pages cover the design of interactive products and services such as websites, household objects, smartphones, computer software, aircraft cockpits, and what have you. Name an item of design interest, and you’ll probably find it discussed inside.

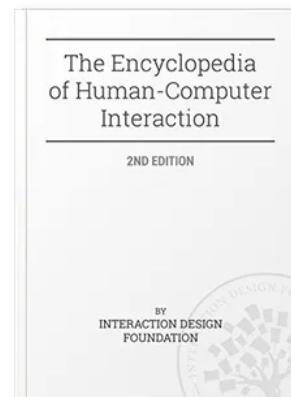


Figure: The Encyclopedia of Human-Computer Interaction, 2nd Ed.. Interaction Design Foundation — Image Credit: Interaction Design Foundation.

¹<http://www.includification.com/>

²<http://everythingisaremix.info/>

³<http://www.bbc.co.uk/news/business-21857393>

⁴<http://www.guardian.co.uk/commentisfree/2013/may/17/online-aged-90-less-lonely>

⁵<http://people.ucsc.edu/~swhittak/>

⁶<https://www.interaction-design.org/>

Appendices

Appendix: Zen and the Art of Motorcycle Maintenance in UX

As we have already discussed, Zen and the Art of Motorcycle Maintenance (ZAMM) [[Pirsig, 1974](#)] may seem like a strange text to use for a Computer Science based UX text; but it isn't. In reality, I am not interested in you remembering anything much within the general narrative of ZAMM. But ZAMM is not really about Zen or indeed motorcycle maintenance, it's about science, quality, and rhetoric. Actually it is much more than this and so I want to talk now about why the concepts contained within it are useful to cover in this text, and how these relate to your understanding and practice of UX within a professional setting.



As you read further, you will start to think this text sounds familiar — that is because it is! Think back to the '[Be Curious, Be Critical!](#)' chapter. Much of this text is recharacterised there - with the ZAMM parts removed. This is so you will realise that the arguments which seemed logical there, are actually much further reaching, that addition reach is important in UX because it is predominantly about accepting and leveraging what is going on in the real world.

The Collision of Two Opposing Ideologies

In the past we characterised practical human computer interaction in terms of usability and interaction engineering (in some cases accessibility was included but mainly as an afterthought). In this case, we decided if an interface was usable and the interaction design was good, based on tangible, measurable metrics such as task completion time. These kinds of metrics enabled us to understand the interactive experience in terms of time, theorising that the least time spent using the interface, the better; and this may have been, in some ways correct, as most computers were used in work situations.

As time passed our concept of the computer and the interface evolved such that computers were no more tied down to the desk but could be mobile or ubiquitous, and the interface was not solely confined to software but may also include aspects of hardware, moving computers from the workplace and into the consumer product domain. Our ways of measuring and valuing the goodness of these interfaces however remained the same, task completion time, errors and error rates, correction times, Fitts Law pointing predictions, etc. Indeed this was the scientific or classic view of technology. At this point intangibles were seen as being soft science, unmeasurable and too open to incorrect interpretation. Other aspects, which might also affect how interfaces were experienced, but which could not be directly measured, and which relied more on subjective views of the user was seen as being, at best inconsequential, and at worst just plain old bad-science.

This clash of ideologies runs large through the whole of Zen and the Art of Motorcycle Maintenance, its main theme being how to unify both the scientific and the romantic; the classic and the aesthetic; the tangible and the intangible; the measurable and the experiential. ZAMM tries to rhetorically unite these two opposing ideologies via a quality framework, indeed this rhetorical attempt eventually sends the author insane; happily this is not the case with practical HCI.

UX is our attempt to unite classic HCI with modern ideas of experience and perception in which accessibility, usability, and interaction engineering (tangible, scientific, measurable) are combined with aesthetic, emotional, fun, affective, collaborative, and gameplay (intangible, humane, difficult to measure).

By understanding the clash of worlds as discussed at length in ZAMM we can also understand the more successful combination of these two – seemingly opposing – ideologies into a unified and cohesive whole as practically applied in UX. By understanding the nuances as discussed in ZAMM, we can better understand what issue we will need to overcome in future UX design, build, and evaluation and also begin to value – and better understand – the subjective qualitative views of the interactive experiences of our users.

Perception of the User Experience

One of the other major themes of ZAMM is that of perception, and the differences which lie between people and their experience of the world. At its most trivial this can be seen in the realisation that all Chris has been able to see for hundreds of miles is the back of his father – until he stands up – and that this is in someway responsible for Chris' boredom and behaviour. However, more deeply the perception of reality for the different protagonists and how that reality is experienced is discussed in detail. This runs from how Chris experiences driving a car when his father is unable to function properly, through to the imposition of normality expected of the author by the society in which he lives, and culminating in the annihilation of his personality. Through to the perception of the authors friends in Bozeman who do not understand that the person they knew does not now exist.

These complicated experiences and perceptions of experience should be taken as warnings to anyone working in user experience. Our perceptions are complicated and incredibly difficult to categorise; what may seem to be obvious to one person, maybe obscure to another. User experience, as opposed to classical HCI, takes these different subjective perceptions into account in its desire to create practical pleasing experiences for each user.

Perception is something not normally measured or quantified in classic HCI and so only in the modern additions of emotion, fun and dynamic interaction can these intangibles be acknowledged. ZAMM shows us that experience can be massively divergent, and that outliers are as important as the general cases (if not more so). By intertwining stories of two different personalities in the same person, and understanding the perceptions – explicitly discussing these perceptions – of experience from different protagonists, ZAMM becomes a valuable teaching tool for UX. It could be thought of as one large ‘Agile Scenario’.

The Discussion and Framing of Science in the User Experience

In its comparison of subjective and objective paradigms – the classical and the romantic – ZAMM does an excellent job of conveying the nature of the scientific method and the work that stems from it. Indeed, in its discussion of empiricism, ZAMM also discusses objectivity and the belief systems that arise around both objectivity and subjectivity.

The discussion of science and its limitations is also pursued, in this case we can see that most of Pirsig's students share a common understanding of quality, in that they can tell quality when they see it, but quality is difficult to measure in any empirical or objective way; or describe with any degree of clarity. It seems in some ways an emergent property, or an umbrella term under which other more easily measured objective indicators can play a part. However, the richness of the description of quality is difficult to place only in such objective terms. So we can see that science cannot be the only measure of user experience, because science is mostly about generalisation, and because we do not have a full model of the universe; we therefore do not know all the variables which may arise to influence the user experience of a single individual. By nature we must conclude, in some regard, that objective, empirical science cannot give us all the answers at this time (until our model is complete), only the answer to testable questions.

This discussion of science is directly related to our discussions of the application of user experience, how we understand modern user experience, and how older styles of human computer interaction serve as an excellent base, but cannot provide the richness which is associated with the intangible, and often unquantifiable subjective, and emotional aspects which we would expect any user experience to comprise of.

We must, however, be cautious. By suggesting that subjective measures may not be testable means that we may be able to convince ourselves and others that a system is acceptable, and even assists or aids the user experience; while in reality there is no evidence, be it theoretical or experimental, which supports this argument. It may be that we are using rhetoric and argumentation to support subjective measures as a way of sidestepping the scientific process which may very well disprove our hypotheses as opposed to support it.

The Conceptualisation of Theoretical and Empirical User Experience

Notice, in the last section we discussed one fundamental of science, the fact that we can disprove or support a hypothesis, in empirical work we cannot prove one. We cannot prove a hypothesis because in the real world we are not able to test every single condition that may be applied to the hypothesis. In this case we can only say that our hypothesis is strong because we have tried to destroy it and have failed. But now notice that in ZAMM this is not the case.

Pirsig, is trained in rhetoric, in theoretical not empirical work, and so his conception

of science is different to ours. In theoretical science (the science Pirsig is familiar with) it is quite possible to prove or disprove the hypothesis. This is because the model of the world is known in full, all tests can be applied, all answers can be evaluated. This is especially the case with regard to mathematics or theoretical physics whereby the mathematical principles are the way the world is modelled, and this theoretical world works on known principles. However this is not the case in user experience, and it is not the case in empirical science whereby we are observing phenomena in the real world and testing our theories using experiments, which may be tightly controlled, but are often as naturalistic as possible. In this case it is not possible to prove a hypothesis, because our model of the world is not complete, because we do not know the extent of the world, or all possible variables, in complex combination, which are able to affect the outcome.

In real-world empirical work we only need one negative result to disprove our hypothesis, but we need to have tested all possibilities to prove our hypothesis correct; we just don't know when everything has been tested.

Rhetoric, Argumentation, and the User Experience

So, how can we satisfy ourselves that subjective, or intangible factors are taken into account in the design process and afterwards. ZAMM provides us with an answer in the form of rhetoric and argumentation. While we may not be able to measure the subjective outcomes or directly generalise them we are able to rationalise these aspects with logical argumentation; and rhetoric – the art of using language effectively so as to persuade or influence others – can obviously play a key role in this. However, you will notice that the problem with rhetoric is that while you may be able to persuade or influence others, especially with the aid of logical argumentation, your results and premise may still be incorrect. Indeed, these failures are also discussed within ZAMM whereby the author discusses rhetorical debates within his Chicago Ph.D. program, but which seem to have little concrete outcome even though the rhetoric is built upon seemingly solid logical and rhetorical foundation. Pirsig, at first fails to 'win' his rhetorical encounter with his supervisor - but then successfully argues the same point and does 'win'; notice that the point is the same, win or loose. If you haven't thought of or don't predict the [counter]arguments that will be made, and have your own convincing counter arguments you'll loose - you may be right, but if you are, your arguments and counter arguments should be complete; that's the point of rhetoric and rhetorical debate.

Remember though, that with the user experience it is not our job to win an argument just for the sake of argumentation or rhetoric itself. We are not there to prove our eloquence, but we are there to support our inductive and deductive reasoning, and our own expertise (Pirsig calls this 'feel' when he refers to it in the context of mechanical repair) when it comes to understanding the user experience within the subjective or intangible.

Further, Pirsig elucidates 'feel' by telling us that 'The difference between a good mechanic and a bad one, like the difference between a good mathematician and a bad one, is precisely this ability to select the good facts from the bad ones on the basis of quality. He has to care! This is an ability about which formal traditional scientific method has

nothing to say.'

Values, and the Intangible Nature of the User Experience

ZAMM's subtitle is 'An Inquiry into Values' and it is useful to remember this in the context of understanding the user experience. In reality, many of the intangibilities which arise in UX work stem from these often hidden values. Values which are expressed throughout the book, from when Pirsig describes his experiences with motorcycle mechanics who may be competent but do not seem to place a value on their work or their ability to fix, in this case, tappets, to Chris Sutherlands view of 'shims'.

More interestingly however, Pirsig discusses the values which are related to the world-views of the people he is with; the Sutherland's. Indeed he uses the Sutherland's to play the counterpart to his mechanical, logical, functional view; the Sutherland's being romantic, aesthetic, and emotional. Again, denoted by the discussion surrounding John's view of the 'shim' created from a can as opposed to that created for the specific purpose and so therefore more aesthetically appealing.

The point here that Pirsig, and myself for that matter, are trying to make is that people bring their own values based on previous experience and their emotional state of equilibrium to any experience. These aspects are intangible and can be difficult to spot especially with regard to understanding the user experience. However, we can also use these values and world-views (if we have some idea of them) to positively influence users emotional response to an interface or interaction. Remember, we have already discussed that the expectation or perception of an experience, be it good or bad, will influence to a large degree the perception of that actual experience once enacted.

The ZAMM Narrative Enhances the User Experience

Finally, let us consider 'Zen and the Art of Motorcycle Maintenance' in more broader terms. I would imagine that if you have read the book you would have immediately found the principles and concepts now that they've been pointed out to you more digestible than those within the main teaching text. It is often very difficult to make textbooks as engaging as a good story, especially when it's overarching ideas and viewpoints you're trying to convey.

Placing these ideas into a more digestible form (such as my use of ZAMM, or Pirsig's use of Chautauquas) is like slipping broccoli into a big Mac, or a vitamin supplement into Coco-hoops. We already know that stories and narrative are a key aspect of usability or 'efficient experience' because they enhanced learnability. This said, key aspects may be lost in the general narrative if they're not signposted or pointed out. This appendix does just that.

Appendix: Defining UX

There are many definitions for user experience⁷ below are a pool of definitions found from the literature and on the Web:

1. All the aspects of how people use an interactive product: the way it feels in their hands, how well they understand how it works, how they feel about it while they're using it, how well it serves their purposes, and how well it fits into the entire context in which they are using it – Alben (1996);
2. All aspects of the end-user's interaction with the company, its services, and its products. The first requirement for an exemplary user experience is to meet the exact needs of the customer, without fuss or bother. Next comes simplicity and elegance that produce products that are a joy to own, a joy to use. True user experience goes far beyond giving customers what they say they want, or providing checklist features. In order to achieve high-quality user experience in a company's offerings there must be a seamless merging of the services of multiple disciplines, including engineering, marketing, graphical and industrial design, and interface design – Nielsen-Norman Group;
3. The overall experience, in general or specifics, a user, customer, or audience member has with a product, service, or event. In the Usability field, this experience is usually defined in terms of ease-of-use. However, the experience encompasses more than merely function and flow, but the understanding compiled through all of the senses – Shadroff;
4. Every aspect of the user's interaction with a product, service, or company that make up the user's perceptions of the whole. User experience design as a discipline is concerned with all the elements that together make up that interface, including layout, visual design, text, brand, sound, and interaction. UE works to coordinate these elements to allow for the best possible interaction by users – UPA;
5. User eXperience (UX) is about how a person feels about using a system. User experience highlights the experiential, affective, meaningful and valuable aspects of human-computer interaction (HCI) and product ownership, but it also covers a person's perceptions of the practical aspects such as utility, ease of use and efficiency of the system. User experience is subjective in nature, because it is about an individual's performance, feelings and thoughts about the system. User experience is dynamic, because it changes over time as the circumstances change – Wikipedia;
6. The overall experience and satisfaction a user has when using a product or system – Old Wikipedia definition, still used e.g. at BitPipe.com.
7. User Experience (abbreviated: UX) is the quality of experience a person has when interacting with a specific design – UXnet.org and Interaction-Design.org;
8. A result of motivated action in a certain context. User's previous experiences and expectations influence the present experience; this present experience leads to more experiences and modified expectations – Mäkelä & Fulton Suri (2001);

⁷AllAboutUX provides information about user experience (UX). The information on the site is collected from the UX community and is shared and maintained by volunteers: Virpi Roto; Ming Lee; Kari Pihkala; Brenda Castro; Arnold Vermeeren; Effie Law; Kaisa Väänänen-Vainio-Mattila; Jettie Hoonhout; and Marianna Obrist. **I'd like to thank everyone who contributes to allaboutux.org, it is a great community resource.**

9. A consequence of a user's internal state (predispositions, expectations, needs, motivation, mood, etc.), the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc.) and the context (or the environment) within which the interaction occurs (e.g. organisational/social setting, meaningfulness of the activity, voluntariness of use, etc.) – Hassenzahl & Tractinsky (2006);
10. The value derived from interaction(s) [or anticipated interaction(s)] with a product or service and the supporting cast in the context of use (e.g., time, location, and user disposition) – Sward & MacArthur (2007).
11. The user experience considers the wider relationship between the product and the user in order to investigate the individual's personal experience of using it – McNamara & Kirakowski (2006);
12. Users' perceptions of interaction that constitute qualities of use – Colbert (2005);
13. An activity of encounter by a computer user with the auditory and visual presentation of a collection of computer programs. It is important to note that this includes only what the user perceives and not all that is presented – Microsoft;
14. An umbrella term used to describe all the factors that contribute to a site user's overall perception of a system. Is it easy to use, attractive and appropriate? Does it meet user needs? – Public Life;
15. The entire set of affects that is elicited by the interaction between a user and a product, including the degree to which all our senses are gratified (aesthetic experience), the meanings we attach to the product (experience of meaning), and the feelings and emotions that are elicited (emotional experience) – Hekkert (2006);
16. UX is a momentary, primarily evaluative feeling (good-bad) while interacting with a product or service – Hassenzahl (2008);
17. A person's perceptions and responses that result from the use or anticipated use of a product, system or service – ISO 9241-210 (2010);
18. A set of material rendered by a user agent which may be perceived by a user and with which interaction may be possible – W3C;
19. Encompasses all aspects of a digital product that users experience directly-and perceive, learn, and use-including its form, behaviour, and content. Learnability, usability, usefulness, and aesthetic appeal are key factors in users' experience of a product – UXmatters;
20. The design of user interaction with a system, product or service considering the usability, the enjoyment and the fit to the way users think – TicToc;
21. The user experience, mostly called 'customer experience' when referring to e-commerce websites; the totality of the experience of a user when visiting a website. Their impressions and feelings. Whether they're successful. Whether they enjoy themselves. Whether they feel like coming back again. The extent to which they encounter problems, confusions, and bugs – UsabilityFirst.com;
22. User experience = Convenience + Design – Cost. Convenience is the king. What makes a product convenient is quite often what makes it usable. It might also relate to the availability of the product. It might also have something to do with laziness and productivity. Defining 'convenience' is by no means an easy task. As is with everything else in this chart, convenience is subjective. Design is what makes a product liked and attractive, even before it has been used. Design is what makes you want the product. It is beauty, the touch of a famous designer, a likeable company, character-pretty much what brand value is thought to be – Nyman (2005);

23. The user experience is the totality of end-users' perceptions as they interact with a product or service. These perceptions include effectiveness (how good is the result?), efficiency (how fast or cheap is it?), emotional satisfaction (how good does it feel?), and the quality of the relationship with the entity that created the product or service (what expectations does it create for subsequent interactions?) – Kuniavsky (2010);
24. The overall perception and comprehensive interaction an individual has with a company, service or product. A positive user experience is an end-user's successful and streamlined completion of a desired task – Goto (2004);
25. UX = the sum of a series of interactions User experience (UX) represents the perception left in someone's mind following a series of interactions between people, devices, and events – or any combination thereof – Fatdux.com;
26. User experience stands for the quality of a global experience as perceived by a person (user) interacting with a system – use-design.com; and
27. Users' judgement of product quality arising from their experience of interaction, and the product qualities which engender effective use and pleasure – Sutcliffe (2010).

Appendix: Defining Accessibility

There are many definitions for accessibility⁸. Below are a pool of definitions found from the literature and on the Web:

1. Web accessibility means that people with disabilities can use the Web. More specifically, Web accessibility means that people with disabilities can perceive, understand, navigate, and interact with the Web, and that they can contribute to the Web. WAI / Thatcher, M. Burks, C. Heilmann, S. Henry, A. Kirkpatrick, P. Lauke, B. Lawson, B. Regan, R. Rutter, M. Urban, and C. Waddell. *Web Accessibility: Web Standards and Regulatory Compliance*, 2006. <http://www.w3.org/WAI/intro/accessibility.php>⁹.
2. Technology is accessible if it can be used as effectively by people with disabilities as by those without. Jim Thatcher <http://www.jimthatcher.com/webcourse1.htm>¹⁰.
3. Individuals with disabilities, who are members of the public seeking information or services from a Federal agency, have access to and use of information and data that is comparable to that provided to the public who are not individuals with disabilities, unless an undue burden would be imposed on the agency. Section 508 http://www.access-board.gov/sec508/standards.htm#Subpart_a¹¹.
4. Accessibility is the word used to describe whether a product (for example, a website, mobile site, digital TV interface or application) can be used by people of all abilities and disabilities. BBC http://www.bbc.co.uk/accessibility/best_practice/what_is.shtml¹².
5. Web accessibility is about making your website accessible to all Internet users (both disabled and non-disabled), regardless of what browsing technology they're using. Webcredible <http://www.webcredible.co.uk/user-friendly-resources/web-accessibility/basics.shtml>¹³.
6. Web site accessibility is the ease with which people—all types of people—can use your site, regardless of situational or physical limitations. W3Access <http://www.w3access.com/>
7. Accessible web design is the practice of designing and developing websites that are usable by everyone. Knowbility <http://wiki.knowbility.org/2010/03/02/web-accessibility-guidelines/>¹⁵.
8. Web Accessibility is about the inclusion and participation of people with disabilities using the web. Imperial College <http://www3.imperial.ac.uk/ict/services/teachingandresearchservices/elearning/webaccessibility>
9. The principle that all web users should have access to information available on the internet. AccessibilIT <http://www.accessibilit.com/public/content/glossary.html#W>¹⁷.

⁸These definitions were compiled for a project involving Yeliz Yesilada, Giorgio Brajnik, Markel Vigo, and myself. Thanks mainly go to Yeliz Yesilada, at the Middle Eastern Technical University (Cyprus Campus), for compiling this list.

⁹<http://www.w3.org/WAI/intro/accessibility.php>

¹⁰<http://www.jimthatcher.com/webcourse1.htm>

¹¹http://www.access-board.gov/sec508/standards.htm#Subpart_a

¹²http://www.bbc.co.uk/accessibility/best_practice/what_is.shtml

¹³<http://www.webcredible.co.uk/user-friendly-resources/web-accessibility/basics.shtml>

¹⁴<http://www.w3access.com/access.htm>

¹⁵<http://wiki.knowbility.org/2010/03/02/web-accessibility-guidelines/>

¹⁶<http://www3.imperial.ac.uk/ict/services/teachingandresearchservices/elearning/webaccessibility>

¹⁷<http://www.accessibilit.com/public/content/glossary.html#W>

10. Users with disabilities can only utilize a web site if it is designed to be compatible with the various assistive technologies. A web site that is sufficiently flexible to be used by all of these assistive technologies is called an accessible web site. Lazar et al. [http://dx.doi.org/10.1016/j.chb.2003.10.018¹⁸](http://dx.doi.org/10.1016/j.chb.2003.10.018).
11. Universal design is the design of products and environments to be usable by all people, to the greatest extend possible, without the need for adaptation of specialised design. Universal design for Web applications by Chisholm and May
12. The extend to which a product/website can be used by specified users with specified disabilities to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. ISO 9241 - quoted from Petrie and Kheir [http://doi.acm.org/10.1145/1240624.1240688¹⁹](http://doi.acm.org/10.1145/1240624.1240688).
13. The problem is that most websites have accessibility barriers that make it difficult or impossible for many people with disabilities to use them. Web accessibility is about removing those barriers so that people with disabilities can use and contribute to the web. Just Ask: Integrating Accessibility Throughout Design by Shawn Lawton Henry
14. Accessible websites allow all users including elderly people and people with disabilities to use the Web content. Web content should be accessed by everyone regardless of disability. 'What is Web accessibility' by Ueki Makoto, Bulletin of the Japan Special Libraries Association, No: 218, pp 28-33 [http://scielinks.jp/j-east/article/200618/000020061806A0638159.php²⁰](http://scielinks.jp/j-east/article/200618/000020061806A0638159.php).
15. A Website is accessible if it deploys services and information so that they can be exploited with no discrimination also by disabled persons. Italian Parliament [http://dx.doi.org/10.1007/978-3-540-85200-1_9²¹](http://dx.doi.org/10.1007/978-3-540-85200-1_9).
16. A website is accessible if it is effective, efficient and satisfactory for more people in more situations. J. Thatcher, C. Waddell, S. Henry, S. Swierenga, M. Urban, M. Burks, B. Regan, and P. Bohman. Constructing Accessible Web Sites. Glasshouse, 2002. [http://dx.doi.org/10.1007/978-3-540-85200-1_9²²](http://dx.doi.org/10.1007/978-3-540-85200-1_9).
17. The removal of all technical barriers to effective interaction. Simon Harper
18. There are no accessible pages. There are pages that are accessible to a person, in an environment (physical and technical), carrying out a task. In fact that is the only way you can determine that something is "accessible" in the absolute. That same page may be in accessible to the same person in a different environment (physical or accessible).

¹⁸<http://dx.doi.org/10.1016/j.chb.2003.10.018>

¹⁹<http://doi.acm.org/10.1145/1240624.1240688>

²⁰<http://scielinks.jp/j-east/article/200618/000020061806A0638159.php>

²¹http://dx.doi.org/10.1007/978-3-540-85200-1_9

²²http://dx.doi.org/10.1007/978-3-540-85200-1_9

Appendix: Ethics Annex

Each set of evaluation standards proposes a slightly different set of principles by which ethical evaluation is to be conducted. However, there are a number of commonalities which are covered throughout all of these standards. It is these commonalities that make up the seven specific principles which will be outlined in more detail in this section. These principles are mainly culled from the American Psychological Association's ethical guidelines along with the ethical guidelines proposed by the Beaumont Report. This is mainly because they are repeated, and in some cases expanded upon, in other more recent work, however, the core principles remain the same.

You Must Be Competent

This first principle relates not so much to the methodology that will be used, or the direct ethical considerations of how that methodology will affect the participants. Rather, it relates directly to the competency of the UX specialist designing the evaluation and with those who will deliver that methodology and analyse the resultant data (in the case of double-blind trials). This places the onus on the UX specialist to make sure that they do not overstep the bounds of their competency, and indeed, that they understand the domains under which, and principles to which, they were trained. If there is an area for which you are not fully trained or that does not have recognised professional standards, you should exercise careful judgement and take appropriate precautions, in some cases be overly cautious, to protect the welfare of those with whom you work and the participants with whom you will be conducting the evaluation. You should also make sure that you maintain your knowledge of the relevant scientific evaluation area and information relating to your UX specialism. This may be through membership of professional and governing bodies; in the United Kingdom this may be through the British Computer Society (BCS), in the United States the Association of Computer Machinery (ACM) or the Institute of Electrical and Electronic Engineers (IEEE), or in Europe via one of the chartered engineering Institutes. Indeed, each country and region normally have their own governing body for computer science; and human computer interaction as a sub discipline. However, in the absence of such a governing body guidance from one of the various experimental psychology associations, most notably the American Psychological Association, may be useful.

You Must Have Integrity

It is often easy in UX evaluations, or scientific work in general, to have a preconceived notion of the desired outcome. Indeed, I use the term 'desired' deliberately in this context, to show how easy it is to influence work in a way which supports the investigators preferred outcome (there should be no desired outcome). Removal of this desire is one of the main reasons why the 'blind method' is used when conducting trials. Blinding is a basic tool to prevent a UX'ers cognitive bias causing them to unconsciously influence the participants of an experiment. It is a significant threat to a study's internal

validity, and is therefore typically controlled using a double-blind experimental design. A double-blind trial describes an especially stringent way of conducting an experiment in an attempt to eliminate subjective bias on the part of both experimental subjects and the experimenters. This use of controls recognises two things: firstly, that there may be an implicit bias imposed by the UX specialist; but also, that the UX specialist, or indeed the company for which they work, may require a certain outcome. It is this final point which needs to be moderated with regard to the ethical procedures of the work, and this is why integrity is such an important part of the process; and a key component of ethically sound methodologies. Striving for personal evaluation integrity enables you to understand: (1) that you must not be concerned with the resultant outcome of the work; (2) that both a seemingly positive, or negative, outcome with regard to an evaluation is valid and tells us more about the interface or the system than we knew before; (3) that by trying to create a valid experiment, free of bias in all aspects of possible influence, enables your work to actually have meaning; and that (4) being driven only by an incorrect desire to validate a bad system or interface will, in the end, not address the needs of the user, or the long-term business case of the organisation conducting the trials.

Conform to Scientific Principles

Belief can alter observations; those with a particular belief will often see things as reinforcing their belief, even if to another observer they would appear not to do so. The scientific method is a body of techniques for investigation and knowledge acquisition which removes the need to believe through the application of empiricism. To be scientific, a method of inquiry must be based on the gathering of observable, empirical, measurable, and refutable evidence, and be subject to specific principles of reasoning. The essential elements of the scientific method are observations, definitions, and measurements of the subject of inquiry; theoretical, hypothetical explanations of observations and measurements of the subject; and reasoning including logical deduction from the hypothesis or theory; the principle means of validation for all of these elements is testing by experimentation which produces observable results. This means that by conforming to scientific principles the evaluation methodology can be validated within a well understood framework. The scientific principles discussed, ensure that the evaluation and testing of the interface and system is performed to maximise the validity and generalisability of the results. Without these safeguards, built into the scientific method, there can be no reliable test that the resultant outcomes represent a truthful understanding of the users interactive behaviour. Therefore, if you do not conduct your evaluations in a scientific manner then your results may be incorrect, and the stress, which you have placed your participants under, will have been for nothing.

Respect Your Participants

At all times you should show a high degree of respect for your human participants within the evaluation setting, including treating them with dignity and respecting their autonomous choices. You must not presume that you know better or try to elicit responses from your participants with which they do not agree. Not every participant will be capable of self-determination or self autonomy. They may exhibit a diminished

capacity in some regard, for instance they may be very young, or have some cognitive or learning difficulties. In this case you should still treat them with the utmost respect. In addition, if you have any suggestion that your participant has some form of diminished autonomy, or that they do not fully understand any aspect of the study, or that they cannot give proper consent, you must look for other participants or the agreement of their guardian. As an UX specialist you must accord appropriate respect for the rights, dignity and worth, of all participants, this includes an individual's right to privacy confidentiality self-determination and autonomy being mindful that legal and other obligations may lead to inconsistency and conflict within sight of these rights. You should also be aware of differences being age, gender, race, and ethnicity and be specifically aware of any cultural aspects which may influence, or affect a person's dignity, or rights, or where culture may exert an implicit force when undertaking experimental evaluation. As part of this respect you should also make sure that your participants have an appropriate understanding of the purpose of the evaluation and why their participation is useful and beneficial. Additionally, you should make them aware of any monitoring or recording devices, within the experimental environment, and how the data collected from those devices will be used and stored. Finally, you should not apply any external force in the recruitment of participants and their participation within the study should be entirely voluntary. This means that at all times you must be more concerned with your participants welfare, and the welfare of the UX'ers undertaking the experimental work.

Maximise Benefits

The maximum for increasing benefits and reducing possible harms is that first proposed by the medical profession: 'do no harm'. Further, the Hippocratic Oath requires physicians to benefit their patients according to their best judgement. Maximising benefits and minimising possible harms is one of the most important aspects of evaluation within the human sciences and more specifically within the UX domain. At all times the experimenter must keep in mind the possible harm's that could occur within the experimental setting. In this case, most ethical committees require a risk assessment to be undertaken before the experiment can proceed. The main problem with understanding benefits is that they can seldom be estimated accurately in advance and in most evaluation the benefit is to science as opposed directly to the individual, or community from which they are drawn. Because the cost to participants and the benefits to science cannot easily be understood, especially at the outset of the evaluation, it falls upon the experimenter the designer of the evaluation study to take a morally responsible decision. Luckily, the evaluation ethics committee can also provide a secondary check regarding this, sometimes tricky, judgement. Finally, another factor to consider when assessing benefits and harms are not those as applied to the individual, but to the society or institution from which individuals are drawn. For example, with regard to studies involving some kinds of religious institutions; there is a very delicate balance between understanding the benefits to science, weighed against the harm that may come to the celebrants, of the religion, or the society which surrounds it, if through this evaluation the institution is diminished or destroyed.

Ensure Justice

Threats to justice often occur due to be inherent power differential between experimenter and evaluation participant. In this case, it is the duty of the experimenter (the UX specialist in this case) to make sure that this is not the case and that participant selection, proceeds were possible, within the populations or communities which will directly benefit from the proposed evaluation. As has been discussed previously, evaluation was often undertaken with participants who would not directly benefit from the results of that evaluation. This means that participants may undergo a number of experimental methodological procedures, but as either individuals or as a subgroup, would never be able to reap the benefits of the resultant understandings gained from those experiments. We have already seen that one of the guiding motivations behind empirical work with human subjects is that the work should do more good than harm, or that more people should benefit than suffer. There are a number of accepted just ways of disputing burdens or benefits: (1) to each person an equal share; (2) to each person according to individual need; (3) to each person according to individual effort; (4) to each person according to societal contribution; and (5) to each person according to merit. Of course these distributions of justice may be difficult in the real world especially when participant selection is by random probabilistic methods. One solution would be to bound the sample frame, or the population under investigation, such that the population from which the representative random sample is drawn receives the benefits and burdens equally. In addition, there should be appropriate procedures to ensure that experimenters, assistants, and participants alike have adequate access mechanisms to address any possible concerns regarding the evaluation itself. So then, in UX, the application of justice is markedly easier than in other subjects. This is mainly because the participants within the study are drawn directly from the population to which that study is applicable; there are normally, no invasive procedures, or procedures which might cause psychological distress or harm and therefore the benefits often outweigh the burdens. The only real burden within an UX evaluation is that of time and effort, and these could be suitably compensated with a financial incentive.

Maintain Trust

UX'ers must maintain a high degree of trust between themselves, the participants, and the experimenters who will be enacting the experimentation. This trust is based upon the agreement about what will, or will not, be involved within the experimentation, and how the data collected from that experiment will, or will not, be used. Aspects of confidentiality and privacy should be addressed from the outset and this should include the anonymising of individual user data and the protection of the identity of the participants. This protection must also be applied over data which may, in combination, be used to identify of the participant. For instance, data which suggests a male, over 85, being treated for a heart condition at Manchester Royal Infirmary, first admitted in June 2006, may enable the identity of that user to be derived. While some UX specialists may not see the importance of maintaining privacy and confidentiality within the user data itself it is important to understand that the use to which this data may be put, by unscrupulous individuals, is often not self-evident. As well as privacy and confidentiality, trust must be maintained throughout the process so that the participants fully understand their roles and responsibilities and those of the experimenter. This enables them to feel safe and comfortable within the evaluation setting so that a more accurate

experiment can be undertaken. We have already seen that we need to minimise the number of confounding variables if you wish to create an accurate study. If a certain level of trust is not maintained then the participant may become agitated or stressed and these two factors may become a confounding variable within the experimental evaluation.

Social Responsibility

Social responsibility is final principle discussed here and in some respects is not directly relevant to the practice of human computer interaction. However, the UX specially should be aware of the professional and scientific responsibilities to both the community, and the society, in which they work and live. This includes not just the local organisation or institution of which they belong, but the wider population within the area and also the more distributed community of practice of which they are part. Indeed, if the UX practitioner brings disrepute onto the UX community by unethical conduct, then the cause of UX, in general, is diminished. As we have already seen, professional ethics, competency, and integrity is key to the practice of UX, and such is its importance that the professional bodies representing the UX specialist take professional conduct very seriously. More importantly we have, as UX specialists, a responsibility to the society in which we live and work. Again, when undertaking scientific evaluation we must strive to advance human welfare and the science of human computer interaction, we must strive to avoid the misuse of our work, or the misunderstanding of the results of that work, we must comply with the law, if the moral and ethical compass of the practitioner agrees. And we must encourage the development of the law and social policies that serve the interests of humans interacting with computers from a software, systems, or ergonomic perspective.

Thanks...

Thanks for proof reading this text, catching typos, suggesting changes, and the like, go to:

- Nima Ara;
- Sean Bechhofer;
- Arina Belova;
- Natasha Birch;
- Andrew Brown;
- Daniel Buckle;
- Adam Cook;
- Tabitha Day;
- Christopher Densham;
- Nic Garner;
- Phoebe Harris;
- Veselin Karaganev;
- Dylan Lewis;
- Aitor Apaolaza Llorente;
- Tom Macpherson-Pope;
- Bijan Parsia;
- Kaiser PÃ©ter;
- Daskiran Phagura;
- James Rowlands;
- Isabella Shaw;
- Ibrahim Sowunmi;
- Jake Thornton;
- Simeon Tsvetankov;
- Lifeng Qiu Lin;
- Markel Vigo; and
- Countless COMP33512 students who didn't provide their name.



If you'd like your name above then send me typos and suggest changes.

References



If you think there are elements missed from these references (which would be best included here) please let me know.

- [Abbott, 2004]** Abbott, Andrew Delano. Methods of discovery: Heuristics for the social sciences (contemporary societies). WW Norton & Company, 2004.
- [Adams, 2019]** Adams, James L. Conceptual blockbusting: A guide to better ideas. Basic Books, 2019.
- [Alkin, 2011]** Alkin, M. C. (2011). Evaluation essentials from A to Z. Guilford Press, New York.
- [Anderson, 2011]** Anderson, S. P. (2011). Seductive interaction design: creating playful, fun, and effective user experiences. New Riders, Berkeley, CA.
- [Ambler, 2005]** Ambler, S. W. (2005). The elements of UML 2.0 style. Cambridge University Press, Cambridge.
- [Association, 2003]** Association, A. P. (2003). Ethical principles of psychologists and code of conduct. American Psychological Association.
- [Ashcraft, 1998]** Ashcraft, M. H. (1998). Fundamentals of cognition. Longman, New York.
- [Baggini and Fosl, 2010]** Baggini, J. and Fosl, P. S. (2010). The philosopher's toolkit: a compendium of philosophical concepts and methods. Wiley-Blackwell, Oxford, 2nd ed edition.
- [Bear et al., 2007]** Bear, M. F., Connors, B. W., and Paradiso, M. A. (2007). Neuroscience: exploring the brain. Lippincott Williams & Wilkins, Philadelphia, PA, 3rd ed edition.
- [Blythe, 2003]** Blythe, M. A. (2003). Funology: from usability to enjoyment, volume v. 3. Kluwer Academic Publishers, Dordrecht.
- [Bowles and Box, 2011]** Bowles, C. and Box, J. (2011). Undercover user experience: learn how to do great UX work with tiny budgets, no time, and limited support. Voices that matter. New Riders, Berkeley, CA.
- [Brand, 1988]** Brand, S. (1988). The Media Lab: inventing the future at MIT. Penguin Books, New York, N.Y., U.S.A.
- [Brewster et al., 2006]** Brewster, S., McGookin, D., and Miller, C. (2006). Olfoto: designing a smell-based interaction. In Proceedings of the SIGCHI conference on Human Factors in computing systems, CHI '06, pages 653–662, New York, NY, USA. ACM.
- [Brooks, 1995]** Brooks, F. P. (1995). The mythical man-month: essays on software engineering. Addison-Wesley Pub. Co., Reading, Mass., anniversary ed edition.
- [Brooker, 2013]** Brooker, C. (2013). How Videogames Changed the World - Wikipedia,

the free encyclopedia. Wikipedia. Retrieved from the Wikipedia on November 17, 2014.²³

[Bryman, 2008] Bryman, A. (2008). Social research methods. Oxford University Press, Oxford, 3rd ed edition.

[Burton, 2003] Burton, H. i. (2003). Visual cortex activity in early and late blind people. Journal of Neuroscience, 23(10):4005–4011.

[Burgess, 2003] Burgess, Robert G. In the field: An introduction to field research. Routledge, 2002.

[Buxton, 2010] Buxton, Bill. Sketching user experiences: getting the design right and the right design. Morgan kaufmann, 2010.

[Card et al., 1983] Card, S. K., Moran, T. P., and Newell, A. (1983). The psychology of human-computer interaction. L. Erlbaum Associates, Hillsdale, N.J.

[Cato, 2001] Cato, J. (2001). User Centered Web Design. Addison Wesley.

[Cockburn, 2001] Cockburn, A. (2001). Writing effective use cases. Addison-Wesley, Boston.

[Cockburn, 2002] Cockburn, A. (2002). Agile software development. Addison-Wesley, Boston.

[Cohn, 2004] Cohn, M. (2004). User stories applied: for agile software development. Addison-Wesley, Boston.

[Cooper, 1999] Cooper, A. (1999). The inmates are running the asylum. Sams, Indianapolis, IN.

[Chisholm, 2008] Chisholm, W. (2008). Universal design for Web applications. O'Reilly Media Inc., Se- bastopol, CA.

[Csikszentmihalyi, 1990] Csikszentmihalyi, M. (1990). Flow: the psychology of optimal experience. Harper & Row, New York, 1st ed edition.

[Dix, 2010] Dix, A. (2010). Human–computer interaction: A stable discipline, a nascent science, and the growth of the long tail. Interacting with Computers, 22(1):13 – 27.

[Fairweather, 2008] Fairweather, P. G. (2008). How older and younger adults differ in their approach to prob- lem solving on a complex website. In Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility, Assets '08, pages 67–72, New York, NY, USA. ACM.

[Faulkner, 2003] Faulkner, L. (2003). Beyond the Five-User Assumption: Benefits of Increased Sample Sizes in Usability Testing. Behavior Research Methods, Instruments, & Computers, 35(3).

[Fishwick, 2006] Fishwick, P. A. (2006). Aesthetic computing. Leonardo. MIT Press, Cambridge, Mass.

[Floridi, 2004] Floridi, L. (2004). The Blackwell guide to the philosophy of computing and information. Blackwell Pub., Malden, MA.

[Fogg, 2003] Fogg, B. J. (2003). Persuasive technology: using computers to change what

²³https://en.wikipedia.org/wiki/How_Videogames_Changed_the_World

we think and do. Morgan Kaufmann Publishers, Amsterdam.

[Garrett, 2003] Garrett, J. J. (2003). The elements of user experience: user-centered design for the Web. New Riders, Indianapolis, Ind., 1st ed edition.

[Gibson, 1977] Gibson, J. J. (1977). The theory of affordances. In Shaw, R., Bransford, J., and of Minnesota. Center for Research in Human Learning, U., editors, Perceiving, acting, and knowing: toward an ecological psychology. Lawrence Erlbaum Associates.

[Gliner et al., 2009] Gliner, J. A., Morgan, G. A., and Leech, N. L. (2009). Research methods in applied settings : an integrated approach to design and analysis. Routledge, New York; London.

[Graziano and Raulin, 2010] Graziano, A. M. and Raulin, M. L. (2010). Research methods: a process of inquiry. Allyn and Bacon, Boston, 7th ed edition.

[Harper, 2007] Harper, S. (2007). Is there design-for-all? Universal Access in the Information Society, 6:111–113.10.1007/s10209-007-0071-2.

[Harper and Yesilada, 2008] Harper, S. and Yesilada, Y. (2008). Web Accessibility: A Foundation for Research, volume 1 of Human-Computer Interaction Series. Springer, London, 1st edition.

[Jordan, 2003] Jordan, P. W. (2003). Designing pleasurable products: an introduction to the new human factors. Taylor & Francis e-Library, London.

[Khaslavsky and Shedroff, 1999] Khaslavsky, J. and Shedroff, N. (1999). Understanding the seductive experience. Commun. ACM, 42:45–49.

[Kirkpatrick et al., 2006] Kirkpatrick, A., Rutter, R., Heilmann, C., Thatcher, J., and Waddell, C. (2006). Web Accessibility: Web Standards and Regulatory Compliance. friends of ED.

[Krug, 2000] Krug, S. (2000). Don't make me think!: a common sense approach to Web usability. Que, Indianapolis, Ind.

[Krum, 2013] Krum, Randy. Cool infographics: Effective communication with data visualization and design. John Wiley & Sons, 2013.

[Lavie and Tractinsky, 2004] Lavie and Tractinsky, N. (2004). Assessing dimensions of perceived visual aesthetics of web sites. International Journal of Human-Computer Studies, 60(3):269–298.

[Law et al., 2009] Law, E. L.-C., Roto, V., Hassenzahl, M., Vermeeren, A. P., and Kort, J. (2009). Under-standing, scoping and defining user experience: a survey approach. In Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, pages 719–728, New York, NY, USA. ACM.

[Lazar et al., 2010] Lazar, J., Feng, J. H., and Hochheiser, H. (2010). Research methods in human-computer interaction. Wiley, Chichester, West Sussex, U.K.

[Licklider, 1960] Licklider, Joseph CR. "Man-computer symbiosis." IRE transactions on human factors in electronics 1 (1960): 4-11.

[Loucopoulos and Karakostas, 1995] Loucopoulos, P. and Karakostas, V. (1995). System requirements engi-neering. International Software Engineering Series. McGraw-

Hill.

[Lund, 2011] Lund, A. (2011). User experience management: essential skills for leading effective UX teams. Morgan Kaufmann, Burlington, MA.

[Lunn and Harper, 2011] Lunn, D. and Harper, S. (2011). Providing assistance to older users of dynamic web content. *Comput. Hum. Behav.*, 27:2098–2107.

[Martin, 2011] Martin, R. C. (2011). The clean coder: a code of conduct for professional programmers. Prentice Hall, Upper Saddle River, NJ.

[Miller, 1956] Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63(2):81–97.

[Narumi et al., 2011] Narumi, T., Nishizaka, S., Kajinami, T., Tanikawa, T., and Hirose, M. (2011). Augmented reality flavors: gustatory display based on edible marker and cross-modal interaction. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, pages 93–102, New York, NY, USA. ACM.

[Neustaedter and Sengers, 2012] Neustaedter, C. and Sengers, P. (2012). Autobiographical design in HCI research: designing and learning through use-it-yourself. In Proceedings of the Designing Interactive Systems Conference, DIS '12, pages 514–523, New York, NY, USA. ACM.

[Neustaedter and Sengers, 2012a] Neustaedter, C. and Sengers, P. (2012). Autobiographical design: what you can learn from designing for yourself. *interactions*, 19(6):28–33.

[Nielsen, 1993] Nielsen, J. (1993). Usability engineering. Academic Press, Boston.

[Nielsen, 2000] Nielsen, J. (2000). Why You Only Need to Test With 5 Users.

[Nielsen, 2013] Nielsen, L. (2013). Personas. The Encyclopaedia of Human-Computer Interaction, 2nd Ed.

[Norman, 1988] Norman, D. A. (1988). The design of everyday things. Basic Books, New York, 1st basic paperback edition.

[Norman, 2004] Norman, D. A. (2004). Emotional design: why we love (or hate) everyday things. Basic Books, New York.

[Oram and Wilson, 2007] Oram, A. and Wilson, G. (2007). Beautiful code. O'Reilly, Beijing, 1st. ed edition.

[Pelachaud, 2012] Pelachaud, C. (2012). Emotion-oriented systems. ISTE, London.

[Picard, 1997] Picard, R. W. (1997). Affective computing. MIT Press, Cambridge, Mass.

[Pirsig, 1974] Pirsig, R. M. (1974). Zen and the art of motorcycle maintenance: an inquiry into values. Morrow, New York.

[Raskin, 2000] Raskin, J. (2000). The humane interface: new directions for designing interactive systems. Addison Wesley, Reading, Mass.

[Raymond, 1998] Raymond, E. S. (1998). Homesteading the Noosphere. *First Monday*, 3(10).

[Raymond, 2001] Raymond, E. S. (2001). The cathedral and the bazaar: musings on

- Linux and Open Source by an accidental revolutionary. O'Reilly, Beijing, rev. ed edition.
- [Reeves and Nass, 1996]** Reeves, B. and Nass, C. I. (1996). The media equation: how people treat computers, television, and new media like real people and places. CSLI Publications, Stanford, Calif.
- [Rettig, 1992]** Rettig, M. (1992). Hat racks for understanding. *Commun. ACM*, 35:21–24.
- [Robertson, 2012]** Robertson, J. i. (2012). Likert-type scales, statistical methods, and effect sizes. *Commun. ACM*, 55(5):6–7.
- [Rodden et al., 2010]** Rodden, K., Hutchinson, H., and Fu, X. (2010). Measuring the user experience on a large scale: user-centered metrics for web applications. In Proceedings of the 28th international conference on Human factors in computing systems, CHI '10, pages 2395–2398, New York, NY, USA. ACM.
- [Rosenberg, 2012]** Rosenberg, A. (2012). Philosophy of science: a contemporary introduction. Routledge contemporary introductions to philosophy. Routledge, New York, 3rd ed edition.
- [Roto et al., 2011]** Roto, V., Law, E., Vermeeren, A., and Hoonhout, J. (2011). USER EXPERIENCE WHITE PAPER: Bringing clarity to the concept of user experience. Technical report, AllAbouUX - <http://www.allaboutux.org/uxwhitepaper>.
- [Sales and Folkman, 2000]** Sales, B. D. and Folkman, S. (2000). Ethics in research with human participants. American Psychological Association, Washington, D.C.
- [Salen and Zimmerman, 2006]** Salen, K. and Zimmerman, E. (2006). The game design reader: a Rules of play anthology. MIT Press, Cambridge, Mass.
- [Schmettow, 2012]** Schmettow, M. (2012). Sample size in usability studies. *Commun. ACM*, 55(4):64–70.
- [Shneiderman, 2001]** Shneiderman, B. (2001). Design: CUU: bridging the digital divide with universal usability. *interactions*, 8(2):11–15.
- [Smith et al., 1982]** Smith, D. C., Irby, C., Kimball, R., Verplank, B., and Harslem, E. (1982). Designing the star user interface. *BYTE*, 7(4):242–282.
- [Shneiderman, 1987]** Shneiderman, B. (1987). Designing the user interface: strategies for effective human-computer interaction. Addison-Wesley, Reading, Mass.
- [Sommerville, 2011]** Sommerville, I. (2011). Software engineering. Pearson, Boston, 9th ed edition.
- [Strachey, 1962]** Strachey, C. (1962). Project stretch. In Planning a Computer System, page 322.
- [Streiner and Sidani, 2010]** Streiner, D. L. and Sidani, S. (2010). When research goes off the rails : why it happens and what you can do about it. Guilford Press, New York.
- [Tullis and Albert, 2008]** Tullis, T. and Albert, B. (2008). Measuring the user experience: collecting, analyzing, and presenting usability metrics. The Morgan Kaufmann interactive technologies series. Elsevier/Morgan Kaufmann, Amsterdam.
- [Unger and Chandler, 2009]** Unger, R. and Chandler, C. (2009). A project guide to UX design: for user experience designers in the field or in the making. Voices that matter.

New Riders, Berkeley, CA.

[Vega, 2008] Vega, S. (2008, September 23). Tom's essay. The New York Times. Retrieved August 5, 2022, from <https://archive.nytimes.com/opinionator.blogs.nytimes.com/2008/09/23/essay/>

[Vanderheiden, 2000] Vanderheiden, G. (2000). Fundamental principles and priority setting for universal usability. In Proceedings on the 2000 conference on Universal Usability, CUU '00, pages 32–37, New York, NY, USA. ACM.

[Van Maanen, 2011] Van Maanen, J. (2011). Tales of the field: on writing ethnography. Chicago guides to writing, editing, and publishing. University of Chicago Press, Chicago, 2nd ed edition.

[Various, 2014a] Various. (2014a). Kolibree Gamified Toothbrush. Kolibree. [Retrieved from the Web Archive on November 17, 2014.](#)²⁴

[Various, 2014b] Various. (2014b). The Basics of APA Style. [Http://www.apastyle.org](http://www.apastyle.org). American Psychological Association. [Retrieved from the Web Archive on November 17, 2014.](#)²⁵

[Veritasium, 2013] Veritasium. (2013). Can You Solve This? [Retrieved from YouTube on November 17, 2014.](#)²⁶

[Vogt et al., 2012] Vogt, W. P., Gardner, D. C., and Haeffele, L. M. (2012). When to use what research design. Guilford Press, New York.

[Watts et al., 1996] Watts, J. C., Woods, D. D., Corban, J. M., Patterson, E. S., Kerr, R. L., and Hicks, L. C. (1996). Voice loops as cooperative aids in space shuttle mission control. In Proceedings of the 1996 ACM conference on Computer supported cooperative work, CSCW '96, pages 48–56, New York, NY, USA. ACM.

[Webb, 1966] Webb, E. J. (1966). Unobtrusive measures; nonreactive research in the social sciences. Rand McNally sociology series. Rand McNally, Chicago.

[Weinschenk, 2011] Weinschenk, S. (2011). 100 things every designer needs to know about people. Voices that matter. New Riders, Berkeley, CA.

[Yesilada and Harper, 2019] Yesilada, Y. and Harper, S. (2019). Web Accessibility: A Foundation for Research, volume 2 of Human-Computer Interaction Series. Springer, London, 2nd edition.

[Zichermann and Cunningham, 2011] Zichermann, G. and Cunningham, C. (2011). Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps. O'Reilly Media Inc.

[Zobel, 2013] Zobel, G. (BBC W. S.) (2013). The light inventor who is poor but proud. BBC News. BBC World Service. [Retrieved from the BBC on November 17, 2014.](#)²⁷

[9241-110:2006, 2006] ISO/TR 9241-110:2006 (2006). ergonomics of human-system interaction – part 110: Dialogue principles. TC/SC: TC 159/SC 4 ICS 13.180, International Organisation for Standardisation (ISO), Geneva, Switzerland.

²⁴<https://web.archive.org/web/20141117164736/http://www.kolibree.com/en/>

²⁵<https://web.archive.org/web/20141117164335/http://www.apastyle.org/learn/tutorials/basics-tutorial.aspx>

²⁶<https://www.youtube.com/watch?v=vKA4w2O61Xo>

²⁷<http://www.bbc.co.uk/news/magazine-23536914>

[9241-20:2008, 2008] ISO 9241-20:2008 (2008). ergonomics of human-system interaction – part 20: Accessibility guidelines for information/communication technology (ict) equipment and service. TC/SC: TC 159/SC 4 ICS: 35.180; 13.180 / Stage: 90.60 (2011-06-17), International Organisation for Standardisation (ISO), Geneva, Switzerland.

[9241-129:2010, 2010] ISO 9241-129:2010 (2010). ergonomics of human-system interaction – part 129: Guidance on software individualisation. TC/SC: TC 159/SC 4 ICS: 35.180; 13.180 / Stage: 60.60 (2010- 11-08), International Organisation for Standardisation (ISO), Geneva, Switzerland.

[9241-171:2008, 2010] ISO 9241-171:2008 (2010). ergonomics of human-system interaction – part 171: Guidance on software accessibility. TC/SC: TC 159/SC 4 ICS: 13.180 / Stage: 90.20 (2011-07-15), International Organisation for Standardisation (ISO), Geneva, Switzerland.

[9241-210:2010, 2010] 9241-210:2010, I. (2010). ergonomics of human-system interaction – part 210: Human- centred design for interactive systems. TC/SC: TC 159/SC 4 ICS: 35.180; 13.180 / Stage: 60.60 (2010- 03-03), International Organisation for Standardisation (ISO), Geneva, Switzerland.

[9241-100:2011, 2011] 9241-100:2011, I. (2011). ergonomics of human-system interaction – part 100: Introduction to standards related to software ergonomics. TC/SC: TC 159/SC 4 ICS 13.180; 35.180, International Organisation for Standardisation (ISO), Geneva, Switzerland.

Glossary of Terms



If you think there are elements missed from the glossary (which would be best defined here) please let me know.

Ted Nelson: Coined the term ‘hypertext.’ He is also seen as something of a radical figure, opposing authority and tradition. He has been called ‘one of the most influential contrarians in the history of the information age.’ — iBiblio.

Vannevar Bush: Consider by many to be the Godfather of the World Wide Web, often making reference to his 1945 essay, ‘As We May Think.’ — iBiblio.

Doug Englebart: Wanted to use technology to augment human intellect. He saw technology, especially computers, as the answers to the problem of dealing with the ever more complex modern world and has dedicated his life to the pursuit of developing technology to augment human intellect — iBiblio.

User Experience: An umbrella term used to describe all the factors that contribute to the quality of experience a person has when interacting with a specific software artefact, or system. It focuses on the practice of user centred: design, creation, and testing, whereby the outcomes can be qualitatively tested using small numbers of users.

User Experience Engineer: We use the term ‘engineer’ to imply a software engineering type role, whereby the UX specialist is also required to contribute code, or at least build working software interface prototypes.

Skunkworks: A skunkworks project is one typically developed by a small and loosely structured group of people who research and develop a project primarily for the sake of radical innovation. The term typically refers to technology projects, and originated with Skunk Works, an official alias for the Lockheed Martin Advanced Development Programs (formerly Lockheed Advanced Development Projects). The reference is to the comic strip Li'l Abner and the job no one wanted: to be the inside man at the Skunk Works. — Wikipedia.

Cognetic: Raskin expanded the meaning of the term cognetics in his book The Humane Interface to mean ‘the ergonomics of the mind’. Raskin discouraged using the informal term intuitive in user interface design, claiming that easy to use interfaces are often due to exposure to previous, similar systems, thus the term ‘familiar’ should be preferred. Aiming for ‘intuitive’ interfaces (based on reusing existing skills with interaction systems) could lead designers to discard a better design solution only because it would require a novel approach — Various Sources.

Jef Raskin: Raskin was an American human-computer interface expert best known for starting the Macintosh project for Apple in the late 1970s. He left Apple in 1982 and formed Information Appliance, Inc. to implement the concepts of his original Macintosh concept. The first product was the SwyftCard, a firmware card for the Apple

II containing an integrated application suite, also released on a disk as SwyftWare — Wikipedia.

Earcons: An Earcon is a brief, distinctive sound used to represent a specific event or convey other information. Taken as originally intended, Earcons are not a common feature of computer operating systems and applications, and while some include general systems error beeps etc, the original intention of Earcons is to convey more complex information such as that present in the graphic of an icon.

Reductionist: Reductionism can mean either (a) an approach to understanding the nature of complex things by reducing them to the interactions of their parts, or to simpler or more fundamental things or (b) a philosophical position that a complex system is nothing but the sum of its parts, and that an account of it can be reduced to accounts of individual constituents. This can be said of objects, phenomena, explanations, theories, and meanings. Reductionism strongly reflects a certain perspective on causality. In a reductionist framework, phenomena that can be explained completely in terms of relations between other more fundamental phenomena, are called epiphenomena. Often there is an implication that the epiphenomenon exerts no causal agency on the fundamental phenomena that explain it — Various Sources (including Wikipedia).

Affective Computing: Affective computing, coined by Rosalind Picard (circa 1995) is the study and development of systems and devices that can recognise, interpret, process, and simulate human emotional changes.

Coding: Coding is a process for both categorising qualitative data and for describing the implications and details of these categories. Initially one does open coding, considering the data in minute detail while developing some initial categories. Later, one moves to more selective coding where one systematically codes with respect to a core concept — Social Methods Knowledge Base.

Memoing: Memoing is a process for recording the thoughts and ideas of the researcher as they evolve throughout the study. You might think of memoing as extensive marginal notes and comments. Again, early in the process these memos tend to be very open while later on they tend to increasingly focus in on the core concept — Social Methods Knowledge Base.

Group-Think: Groupthink is a psychological phenomenon that occurs within groups of people. It is the mode of thinking that happens when the desire for harmony in a decision-making group overrides a realistic appraisal of alternatives. Group members try to minimise conflict and reach a consensus decision without critical evaluation of alternative ideas or viewpoints. Antecedent factors such as group cohesiveness, structural faults, and situational context play into the likelihood of whether or not groupthink will impact the decision-making process. The primary socially negative cost of groupthink is the loss of individual creativity, uniqueness, and independent thinking — Wikipedia.

JAWS: ‘Job Access With Speech’ is a computer screen reader program in Microsoft Windows that allows blind and visually impaired users to read the screen either with a text-to-speech output or by a Refreshable Braille display. JAWS is produced by the Blind and Low Vision Group of Freedom Scientific, St. Petersburg, Florida, USA. — Wikipedia.

PET: Positron emission tomography (PET) is nuclear medicine imaging technique that produces a three-dimensional image or picture of functional processes in the

body. The system detects pairs of gamma rays emitted indirectly by a positron-emitting radionuclide (tracer), which is introduced into the body on a biologically active molecule. Three-dimensional images of tracer concentration within the body are then constructed by computer analysis. In modern scanners, three dimensional imaging is often accomplished with the aid of a CT X-ray scan performed on the patient during the same session, in the same machine — Wikipedia.

Screen Reader: A screen reader is a software application that attempts to identify and interpret what is being displayed on the screen (or, more accurately, sent to standard output, whether a video monitor is present or not). This interpretation is then re-presented to the user with text-to-speech, sound icons, or a Braille output device. Screen readers are a form of assistive technology (AT) potentially useful to people who are blind, visually impaired, illiterate or learning disabled, often in combination with other AT, such as screen magnifiers — Wikipedia.

Task Completion Time: (TCT) is a measure of the time it takes a user to perform a task (from start to finish). This is a typical metric in usability evaluation — [Usability First²⁸](#).

PARC: Palo Alto Research Center, or Xerox PARC, was founded in 1970 as a division of Xerox Corporation, PARC has been responsible for such well known and important developments as laser printing, Ethernet, the modern personal computer, graphical user interface (GUI), object-oriented programming, ubiquitous computing, amorphous silicon (a-Si) applications, and advancing very-large-scale-integration (VLSI) for semiconductors. Xerox PARC (and now PARC) is a mecca human facing thinkers and included three Turing Award winners — Various Sources.

Eat Their Own Dog Food: More correctly ‘Eating your own dog food’, also called dogfooding, is when a company (usually, a software company) uses the products that it makes~~~Wikipedia.

Heuristics: Heuristics are strategies using readily accessible, though loosely applicable, information to control problem solving in human beings and machines. Heuristic (meaning to find or discover) refers to experience-based techniques for problem solving, learning, and discovery. Heuristic methods are used to speed up the process of finding a satisfactory solution, where an exhaustive search is impractical. Examples of this method include using a ‘rule of thumb’, an educated guess, an intuitive judgment, or common sense — Wikipedia.

Creatives: A creative person, a person whose job involves creative work; a person who carries out creative work on an advertising campaign say, esp. a copywriter, art director, or designer. (of a person) having good imagination or original ideas — OED.

Mechanical Turk: Was a fake chess-playing machine constructed in the late 18th century. From 1770 until its destruction by fire in 1854, it was exhibited by various owners as an automaton, though it was exposed in the early 1820s as an elaborate hoax — Wikipedia.

Sample Frame: In statistics, a sampling frame is the source material or device from which a sample is drawn. It is a list of all those within a population who can be sampled, and may include individuals, households or institutions — Wikipedia.

Probabilistic: Probability sampling is a sampling technique wherein the samples are gathered in a process that gives all the individuals in the population equal chances of

²⁸<http://www.usabilityfirst.com/>

being selected — <http://www.experiment-resources.com>²⁹.

Non-Probabilistic: Non-probability sampling is a sampling technique where the samples are gathered in a process that does not give all the individuals in the population equal chances of being selected — <http://www.experiment-resources.com>³⁰.

Internal Validity: The key question in internal validity is whether observed changes can be attributed to your changes (i.e., the cause) and not to other possible causes (sometimes described as ‘alternative explanations’ for the outcome) — <http://www.socialresearchmethods.net>

External Validity: External validity refers to the approximate truth of conclusions the involve generalizations. Put in more pedestrian terms, external validity is the degree to which the conclusions in your study would hold for other persons in other places and at other times — <http://www.socialresearchmethods.net>³².

Confounding Factors: Confounding variables (or factors) are variables that the researcher failed to control, or eliminate, damaging the internal validity of an experiment — <http://www.experiment-resources.com>³³.

Power Analysis: Power analysis allows us to make sure that we have looked hard enough to find it, if there is enough of it there to bother us. The size of the thing we are looking for is known as the ‘effect size’. Several methods exist for deciding what effect size we would be interested in. Different statistical tests have different effect sizes developed for them, however the general principle is the same — www.jeremytaylor.co.uk³⁴.

Statistical Validity: In science and statistics, validity has no single agreed definition but generally refers to the extent to which a concept, conclusion or measurement is well-founded and corresponds accurately to the real world — Wikipedia.

²⁹<http://www.experiment-resources.com>

³⁰<http://www.experiment-resources.com>

³¹<http://www.socialresearchmethods.net>

³²<http://www.socialresearchmethods.net>

³³<http://www.experiment-resources.com>

³⁴<http://www.jeremytaylor.co.uk>