

Human-Centred Interaction Design

Scoping a problem with PACT

The aim of human-centred interaction design is to harmonise the PACT elements in a particular domain. Designers want to get the right mix of technologies to support the activities being undertaken by people in different contexts. A PACT analysis is useful for both analysis and design activities; understanding the current situation, seeing where possible improvements can be made or envisioning future situations. To do a PACT analysis the designer simply scopes out the variety of P, A, C and Ts that are possible. This can be done using brainstorming and other envisionment techniques (e.g., draw pictures, sketches, cartoons, cut out pictures from magazines and stick them on a board).and by working with people through observations, interviews and workshops.

The results can be written up as detailed concrete ‘**scenarios** of use’. Scenarios are stories about people undertaking activities using technologies in contexts. Develop conceptual scenarios that cover the main activities that the technology has to support. Develop concrete versions of these for specific designs of the technology. For example - a conceptual scenario might say ‘Pete logs onto the computer’, and a concrete version might be ‘Pete clicks on the “log on” icon’

The designer should look for trade-offs between combinations of PACT and think about how these might effect design.

For people, designers need to think about the physical, psychological and social differences and how those differences change in different circumstances and over time. It is most important that designers consider all the various stakeholders in a project, not simply the ‘end users’. There are often groups of people who have an interest, or stake, in a project who will never use the system; managers, administrators, customers can all be affected by changed systems that other people are using.

Developing ‘Personas’ can be useful here: A **persona** is a profile of a typical user; it is a description of an archetypal user synthesized from a series of interviews with real people and includes a name, a social history, and a set of goals that drive the design of the product or web site. By closely adhering to the goals of a specific persona, the designers satisfy the needs of the many users who have goals similar to those of the persona. The process is even more effective when designers design for several personas simultaneously, as they can satisfy an even larger number of users.

For activities designers need to think about the complexity of the activity (focused or vague, simple or difficult, few steps or many), the temporal features (frequency, peaks and troughs, continuous or interruptible), co-operative features and the nature of the data. For contexts they think about the physical, social and organisational setting and for technologies they concentrate on input, output, communication and content.

The Process of Human-Centred Interaction Design

Design is a creative process concerned with bringing about something new. It is a social activity with social consequences. It is about conscious change and communication between designer and user. Different design disciplines have different methods and techniques for helping with this process. Approaches to, and philosophies of design change over time.

In mature disciplines, examples of good design are built up and people can study and reflect upon what makes a certain design great, good or awful. Different design disciplines have different constraints such as whether the designed object is ‘stand alone’ or whether it has to fit in and live with legacy systems or conform to standards.

Activities in Design

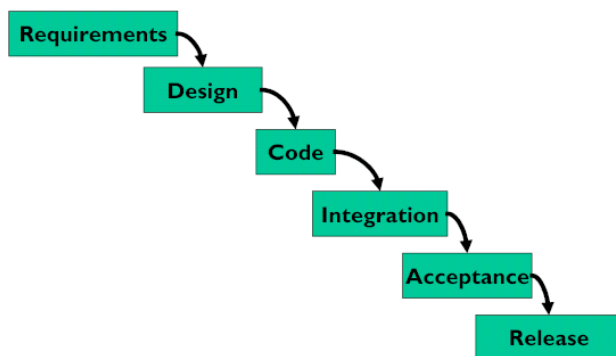
There are many different ways of characterising the activities involved in the design process. For David Kelley, founder of the product design company IDEO:

‘Design has three activities: understand, observe and visualize.’

He says

‘Remember, design is messy; designers try to understand this mess. They observe how their products will be used; design is about users and use. They visualize which is the act of deciding what it is’.

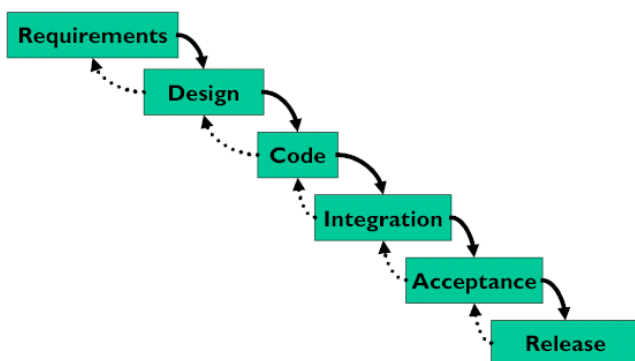
Yet the classic software engineering design life cycle is the waterfall model:



It models the design process as a sequence of stages. Each stage results in a concrete product – a requirements document, a design, a set of coded modules – that feeds into the next stage. Each stage also includes its own validation: the design is validated against the requirements, the code is validated (unit-tested) against the design, etc.

The biggest improvement of the waterfall model over previous (chaotic) approaches to software development is the discipline it puts on developers to think first, and code second. Requirements and designs generally precede the first line of code.

Validation is not always sufficient; though. Sometimes problems are missed until the next stage.



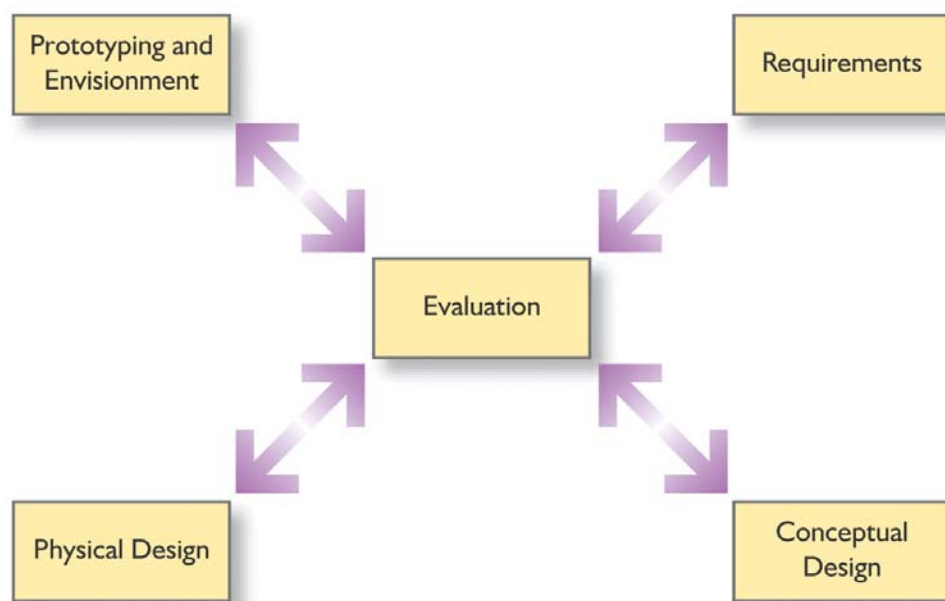
Trying to code the design may reveal flaws in the design – e.g., that it can’t be implemented in a way that meets the performance requirements. Trying to integrate may reveal bugs in the code that weren’t exposed by unit tests.

So the waterfall model implicitly needs feedback between stages.

The danger arises when a mistake in an early stage – such as a missing requirement – isn't discovered until a very late stage – like acceptance testing. Mistakes like this can force costly rework of the intervening stages.

Although the waterfall model is useful for some kinds of software development, it's very poorly suited to user interface development. Firstly, UI development is inherently risky. UI design is hard and we don't (yet) have an easy way to predict whether a UI design will succeed. Secondly, in the usual way that the waterfall model is applied, users appear in the process in only two places: requirements analysis and acceptance testing. Hopefully we asked the users what they needed at the beginning (requirements analysis), but then we code happily away and don't check back with the users until we're ready to present them with a finished system. So if we screwed up the design, the waterfall process won't tell us until the end. Thirdly, when UI problems arise, they often require dramatic fixes new requirements or new design..

A more appropriate design process for interactive systems is the Star Life cycle:



Hix and Hartson developed this user-centred and highly iterative model for the software design process after studying real designers at work. The key features of this process are:

- Evaluation is central to designing interactive systems. Everything gets evaluated at every step of the process
- The process can start at any point – sometimes there is a conceptual design in place, sometimes we start with a prototype, sometimes we start with requirements
- The activities can happen in any order, for example, requirements might be evaluated and a prototype built and evaluated and some aspect of a physical design might then be identified

Requirements

Requirements are concerned with what the system has to do, what it has to be like, how it has to fit in with other things. There are both functional and non-functional requirements to consider. Functional requirements are concerned with what the system should be able to do and with the functional constraints of a system. It is important for the designer to think about the whole human-computer system in an abstract way. Deciding who does what, when something should be displayed or the sequence in which actions are undertaken should come later in the design process. A good analysis of an activity will strive to be as independent of current practice as is possible. Of course there are always functional constraints – the realities of what is technically possible - which render certain ordering, sequencing and allocation of function inevitable. There are also logical and organisational constraints which may make particular designs infeasible.

Requirements are generated through discussions with future clients or users of the system, observations of existing systems and what people do. Requirements can be generated through working with people in focus groups, design workshops and so on. The aim is to collect and analyse the stories people have to tell. Requirements is essentially about understanding.

Conceptual design

Conceptual design is about designing a system in the abstract, about considering what information and functions are needed for the system to achieve its purpose. It is about deciding what someone will have to know to use the system. It is about finding a clear conceptualisation of a design solution and how that conceptualisation will be communicated to people (so that users will quickly develop a clear mental model).

There are a number of techniques to help with conceptual design. Software Engineers prefer modelling possible solutions with objects and relationships. Entity-relationship models are another popular conceptual modelling tool. Flow can be represented using dataflow diagrams and structure can be shown with structure charts. The conceptual design of a web site, for example, will include a site map and a navigation structure.

One way to conceptualise the main features of a system is to use a ‘Rich Picture’. (*Please see the accompanying article by Monk and Howard*). A rich picture captures the main conceptual relationships between the main conceptual entities in a system – a model of the structure of a situation. Peter Checkland, who originated the soft systems approach, also emphasises focusing on the key transformation of a system. This is the conceptual model of processing. The principle stakeholders - the customers, actors, system owners – should be identified. The designer should also consider the perspective from which an activity is being viewed as a system (the *Weltanschauung*) and the environment in which the activities take place. Most importantly the rich picture identifies the issues, or concerns of the stakeholders, thus helping to focus attention of problems or potential design solutions.

The key feature about conceptual design is to keep things abstract – focus on the ‘what’ rather than the ‘how’ - and to avoid making assumptions about how functions and information will be distributed. There is not a clear cut distinction between conceptual and physical design, but rather there are degrees of conceptuality.

Physical design

Physical design is concerned with how things are going to work and with detailing the look and feel of the product. Physical design is about structuring interactions into logical sequences and about clarifying and presenting the allocation of functions and knowledge between people and devices. The distinction between conceptual and physical design is very important. The conceptual design relates to the overall purpose of the whole human-computer system. Between the people and the technologies there has to be enough knowledge and ability to achieve the purpose. Physical design is concerned with taking this abstract representation and translating it into concrete designs. On one side this means requirements for hardware and software and on the other side it defines the knowledge, tasks and activities that people will be required to do. There are three components to physical design; operational design, representational design and interaction design.

- **Operational design** is concerned with specifying how everything works and how content is structured and stored. Taking a functional view of an activity means focusing on processes and on the movement, or flow, of things through a system. *Events* are occurrences that cause, or trigger, some other functions to be undertaken. Sometimes these arise from outside the system under consideration and sometimes they arise as a result of doing something else. For example, some activity might be triggered on a particular day, or at a particular time, another might be triggered by the arrival of a person, or document.
- **Representational design** is concerned with fixing on colours, shapes, sizes and information layout. It is concerned with style and aesthetics and is particularly important for issues such as the attitudes and feelings of people, but also for the efficient retrieval of information.

Style concerns the overall 'look and feel' of the system. Does it appear old and 'clunky' or is slick, smooth and modern? What mood and feelings does the design engender? For example most Microsoft products engender an 'office' and 'work' mood, serious rather than playful. Many other systems aim to make the interaction engaging, some aim to make it challenging and others entertaining. In multimedia and games applications this is particularly important.

- **Interaction design** is concerned with the allocation of functions to human or to technology and with the structuring and sequencing of the interactions. Allocation of function has a significant impact on the usability of the system. For example, consider the activity of making a phone call. Certain functions are necessary; indicate a desire to make a phone call, connect to the network, enter the phone number, make connection. Years ago a telephone exchange was staffed by people and it was these people who made connections by physically putting wires into connectors. In the days of wired phones, picking up the receiver indicated the desire to make a call, the full number had to be dialled in and then the telephone exchange would automatically make the connections. Nowadays a person has to press the connect button on a cellular phone, choose someone's name from the phone's address book and the technology does the rest.

Prototyping and envisionment

Designs need to be visualised both to help designers clarify their own ideas and to enable people to evaluate them. Prototyping and envisionment is concerned with finding appropriate media in which to render to design ideas. The medium needs to be appropriate for the stage of the process, the audience, the resources available and the questions that the prototype is helping to answer.

Techniques for prototyping and envisionment include any way in which abstract ideas can be brought to life. Sketches 'on the back of an envelope', fully functioning prototypes, cardboard mock ups are just some of the methods used.

Evaluation

Evaluation is tightly coupled with envisionment because the nature of the representation used will affect what can be evaluated. The evaluation criteria will also depend on who is able to use the representation. Any of the other design activities will be followed by an evaluation. Sometimes this is simply the designer checking through to make sure something is complete and correct. It could be a list of requirements or a high level design brief that is sent to a client, an abstract conceptual model that is discussed with a colleague, or it may be a formal evaluation of a functional prototype by the future system users.

Techniques for evaluation are many and various depending once again on the circumstances. The important thing to keep in mind is that the technique used must be appropriate for the nature of the representation, the questions being asked and the people involved in the evaluation.

Implementation

Ultimately things have to be engineered and software has to be written and tested. Databases have to be designed and populated and programs have to be validated. The whole system needs to be checked to ensure that it meets the requirements until finally the system can be formally 'launched' and signed off as finished.

Implementation can account for a significant portion of total development costs. Clients will often want extra features when they see a system nearing completion, but these will have to be costed and paid for. On the other hand the developers need to ensure that their system really does meet the specification and does not contain any 'bugs'.

Doing design

To illustrate the design process, let us return to the example of the laboratory access system. We use letters to illustrate which activity is being undertaken.

R = requirements,
C = conceptual design,
P = prototyping and envisionment,
D = physical design,
E = evaluation.

Notice the iterative nature of the design and how some details get fixed that then affect other possibilities. Notice also how the problem and solution evolve together, for example, through clarification of requirements.

The PACT analysis has resulted from an initial brief (R) and discussions between the clients (the university department) and our designers (R). Some requirements have been written down (P) and discussed further by clients and designers (E). As a result some points of clarification with the university department has taken place (R). We can now speculate about possible designs; three conceptual designs have been suggested (C)

1. Each user of the laboratory could be issued with a card that gets swiped through a card reader at the door.
2. Instead of having a card people could type in a number on a key pad.
3. Other technologies such as iris recognition could be used.

These conceptual designs can now be evaluated (E) by discussing with colleagues and client. For example, the advantages with 1 are that most people already have some form of card, but visitors would have to be issued with a temporary card. The activity would be quite quick if the technology works OK, but sometimes cards do not swipe properly. Even with a total time of only 2 seconds this might take a long time at the start of a lab session. It has the advantage that different laboratories can allow access to different groups, but it may be confusing for people if they cannot get into a particular lab when they want to. Good signage is needed to go with such a system. With option 2 it would take longer than swiping a card, but it is easier to issue people with number than it is to issue cards. Technologies for 3 are still in their infancy. Besides the same problems with visitors arise.

As result of these discussions some further technical research would probably be undertaken so that technical opportunities and constraints can be clarified (R). In terms of physical design (D) there is not much opportunity in this case as most of the technology will be bought in, but it may be that concrete prototypes (P) would need to be developed and evaluated (E) to verify that the system would work when under time pressure.