UNIT-II

Interaction Design Basics

What is Design?

Design - achieving goals within constraints

Goals - what is the purpose of the design we are intending to produce? who is it for? why do they want it?

For eg, if we are designing a wireless personal movie player, we may think about young affluent users. wanting to watch the ~~taste~~ latest movies ~~or~~ at the same time on the move & download free copies.

Constraints - what materials must we use? what standard must we adopt? How much can it cost? How much time do we ~~want~~ have to develop it? Are there health & safety issues?

In the eg, of movie player, does it withstand to rain? Must we use existing video standards to download movies?

The golden rule of design is i.e understand your materials.

For HCI the materials are

- understand computers : limitations, capacities, tools, platforms.
- understand people : psychological, social aspects, human error

The Process of Design

We will look in detail at the software development process & how HCI fits within it.
Here we will take a simplified view of four main phases plus an iteration loop, focused on the design of interaction.

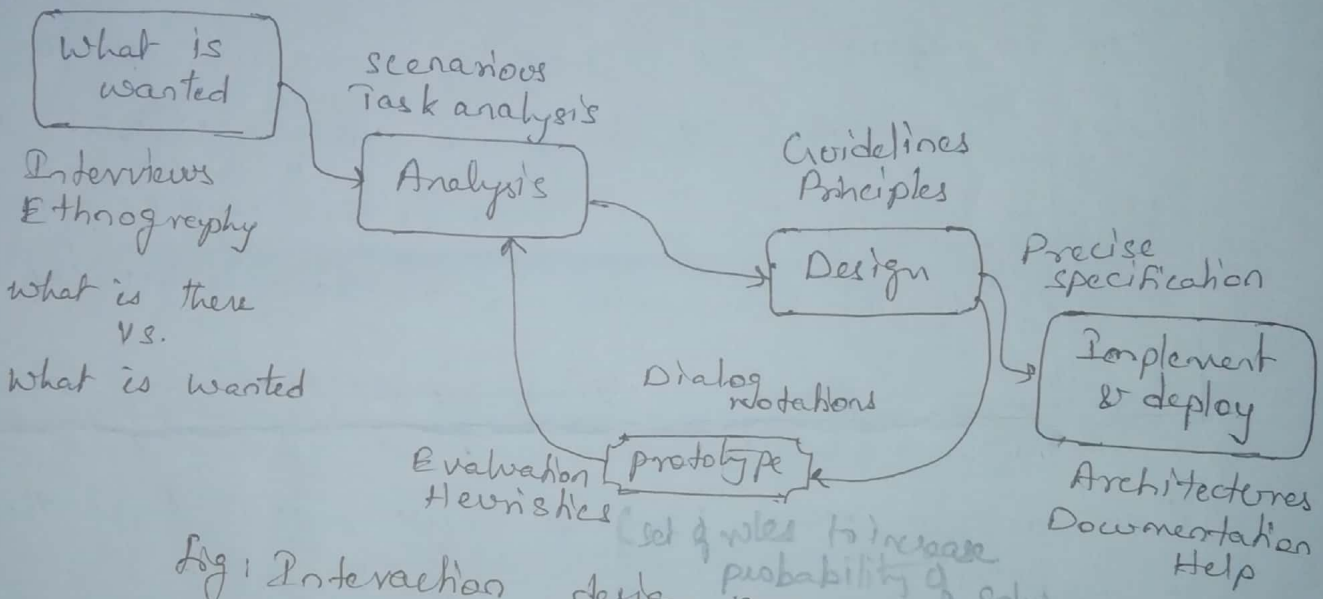


Fig: Interaction design process

Requirements - what is wanted - The first stage is establishing what exactly is needed. As a precursor to this it is usually necessary to find out what is currently happening.

There are a number of techniques used for this in HCI: interviewing people, videotaping them, looking at the documents & objects that they work with, observing them directly.

Ethnography, a form of observation deriving from anthropology.

Analysis - The results of observation & interview need to be ordered in some way to bring out key issues & communicate with later stages of design.

- We will look at scenarios, rich stories of interaction, which can be used in conjunction with a method like task analysis or on their own to record & make vivid actual interaction. These techniques can be used both to represent the situation as it is and also the desired situation.

Design - There are numerous rules, guidelines and design principles that can be used to help in design.

- We need to record our design choices in some way and there are various notations and methods to do this, including those used to record the existing situation.

Iteration & prototyping - Humans are complex & we cannot expect to get designs right first time. We therefore need to evaluate a design to see how well it is working and where there can be improvements.

It is hard to get real feedback on paper without trying it out.

Most user interface design therefore involves some form of prototyping, producing early versions of systems to try out with real users.

Implementation & deployment - Finally, when we are happy with our design, we need to create it and deploy it. This will involve writing code, perhaps perhaps making hardware, writing documentation and manuals - everything that goes into a real system that can be given to others.

Scenarios

Scenarios are stories for design: rich stories of interaction.

They are perhaps the simplest design representation, but one of the most flexible & powerful.

Some scenarios are quite short: the user intends to press the 'save' button, but accidentally presses the 'quit' button so loses his work.

Scenarios can be augmented by sketches, simulated screen shots, etc. These sketches & pictures are called story boards.

Where the design includes physical artifacts, the scenarios can be used as a script to act out potential patterns of use.

In addition scenarios can be used for to:

① communicate with others - other designers, clients or users

3) Validate other models - A detailed scenario can be 'played' against various more formal representations such as task models or dialog & navigation models.

3) Express dynamics - Individual screen shots & pictures give you a sense of what a system would look like, but not how it behaves.

Navigation Design

The object of design is not just a computer system or device, but the socio-technical intervention as a whole.

Imagine yourself using a word processor. You will be doing this in some particular social & physical setting, for a purpose.

But now we are focussing on the computer system itself. You interact at several levels:

widgets. The appropriate choice of widgets and wording in menus & buttons will help you know how to use them for a particular selection or action.

screens or windows - You need to find things on the screen, understand the logical grouping of buttons.

Levels of Interaction

PC application	website	Physical device
- widgets	- form elements, tags & links.	- Buttons, dials, lights, displays
- Screen design	- Page design	- Physical layout
- Navigation design	- Site structure	- Main modes of device
- Other apps & operating system	- The web, browser, external links	- The real world

- Navigation within the application - You need to be able to understand what will happen when a button is pressed, to understand where you are in the interaction.

- Environment - The word processor has to read documents from disk, perhaps some are on remote networks. You swap between applications, perhaps cut & paste.

The above table shows, there are differences, for eg., in the web we have less control of how people enter a site & on a physical device we have the same layout of buttons & displays no matter what the internal state

The place to start when considering the structure of an application is to think about actual use :

- who is going to use the application?
- how do they think about it?
- what will they do with it?

In the Navigation design, we will look mainly on, i.e the main screens or modes within a system & how they interconnect. & also how this interacts with wider environment.

In the navigation design, we will consider two main kinds of issue:

1) Local structure - looking from one screen or page out.
2) Global structure - structure of site, movement between screens.

Local structure:

- Much of interaction involves goal-seeking behaviour.
- Users have some idea of what they are after & a partial model of the system.

To do this goal seeking, each state of the system or each screen needs to give the user enough knowledge of what to do to get closer to their goal.
To get you started, here are four things to look for when looking at a single web page, screen or state of a device.

- knowing where you are
- knowing what you can do
- knowing where you are going - or what will happen.
- knowing where ~~you~~ you've been - or what you have done.

The screen, web page or device displays should make clear where you are in terms of the interaction or state of the system.

It is also important to know what you can do — what can be pressed or clicked to go somewhere or do something. Some web pages are particularly bad in that it is ~~evel~~ unclear which images are pure decoration and which are links to take you somewhere.

You then need to know where you are going when you click a button or what will happen of course you can ~~click~~ try clicking the button to see.

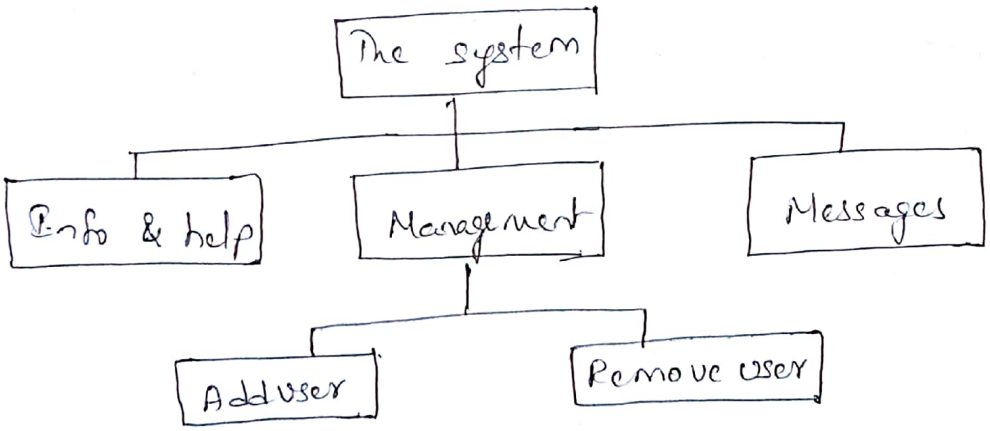
Finally, if you have just done some major action you also want some sort of confirmation of what you have done. If you are faultless and have perfect knowledge, of course you will be sure that you have hit the right key & know exactly what will happen.

Global structure - hierarchical organization

We will now look at the overall structure of an application. This is the way the various screens, pages or device states link to one another.

One way to organize a system is in some form of hierarchy. This is typically organized along functional boundaries (i.e different kinds of things), but may be organized by roles, user type or some more esoteric breakdown such as modules in an educational system.

The hierarchy links screens, pages or states in logical groupings.



Application functional hierarchy

The above fig. gives a high-level breakdown of some sort of messaging system. This sort of hierarchy can be used purely to help during design, but can also be used to structure the actual system. For eg. this may reflect the menu structure of a PC application or the site structure on the web.

Global structure - dialog

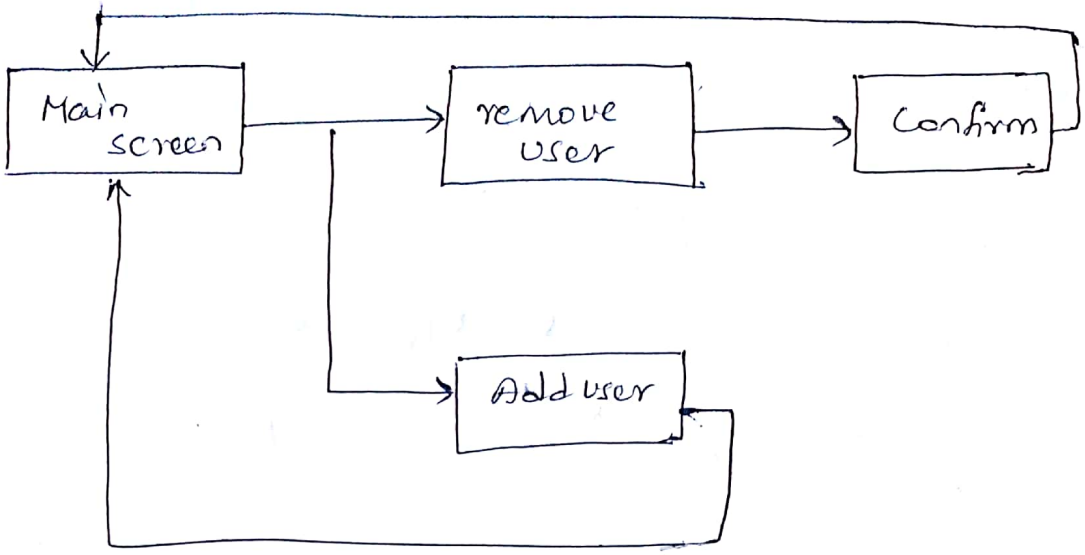
In a pure information system or static ~~web~~ website it may be sufficient to have a fully hierarchical structure, perhaps with next/previous links between items in the same group.

For eg, in a stock control system there may be a way of going from a stock item to all orders outstanding on that item & then from an order to the purchase record for the customer who placed the order. These would each be in a very different part of a hierarchical view of the application, yet directly accessible from one another.

In HCI the word 'dialog' is used to refer to the pattern of interactions between the user & a system.

A simple way is to use a network diagram showing the principal states or screens linked together with arrows. This can:

- show what lead to what
- show what happens when
- include branches & loops
- be more task oriented than a hierarchy



Network of screens / states

The above fig shows a network diagram illustrating the main screens for adding or deleting a user from the messaging system.

The arrows show the general flow between the states. We can see that from main screen we can get to either the 'remove user' screen or the 'add user screen', by selecting buttons or links. We can also see that from the 'add user' screen the system always returns to the main screen, but after the 'remove user' screen there is a further confirmation screen.

Screen design and Layout:

The basic principles at the screen level reflect those in other areas of interaction design:

Ask : What is the user doing?

Think : what information is required? what comparisons may the user need to make? In what order are things likely to be needed?

Design : Form follows function: let the required interactions drive the layout.

Tools for Layout

We have a number of visual tools available to help us suggest to the user appropriate ways to read and interact with a screen or device.

1) Grouping & structure:

If things logically belong together, then we should normally group them together. This may involve multiple levels of structure.

eg: Order:

Administrative Information
Billing details
Delivery details

Order Information
order line 1
order line 2

2) Order of groups & items:

In general we need to think: what is the natural order for the user? This should normally match the order on screen. For data entry forms or dialog boxes should also set up the order in which the tab key moves between fields.

eg:

Billing details:
Name:
Address:
Credit card no:

Delivery details:
Name:
Address:
Delivery time:

Order details:

Item	quantity	cost/item	cost
size 10 screws	7	3.71	25.97
⋮	⋮	⋮	⋮

3) Decoration:

Decorative features like font style, & text or background colours can be used to emphasize groupings.

eg: Consider Microwave control panel, see how the buttons differ in using the foreground and background colors (green & gold) so that groups are associated with one another.

4) Alignment:

Alignment of lists is also very important. For users who read text from left to right, lists of text items should be aligned to the left, numbers, however, should be aligned to the right or at the decimal point.

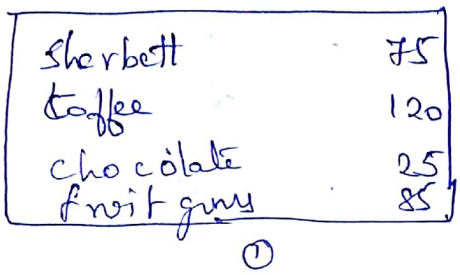
Multiple columns lists require more care. Text columns ~~has require~~ have to be wide enough for the largest item., which means you can get large between columns.

There are several visual ways to deal with this including:

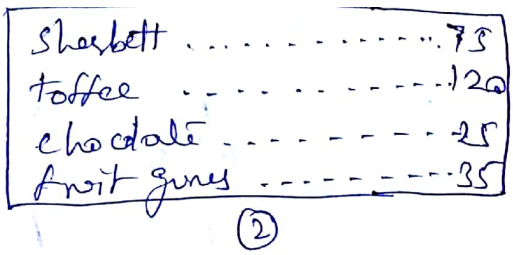
(1) Leaders - lines of dots linking the columns.

(2) using soft tone grays or ~~action~~ colors behind rows or columns.

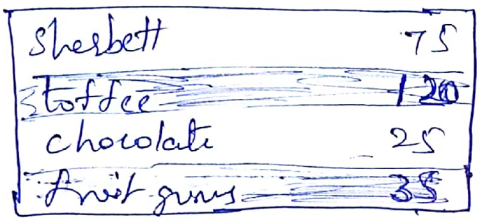
(3) This last alternative might be a good solution if you were frequently scanning the numbers & only occasionally scanning the names of items,



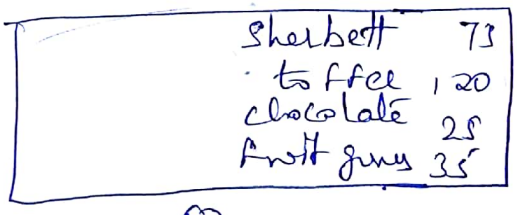
①



②



③



④

5) White space:

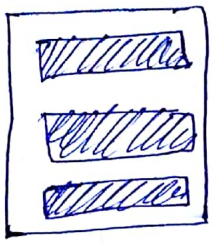
Space can be used in several ways. Some of these are shown below.

The colored areas represent continuous areas of text or graphics.

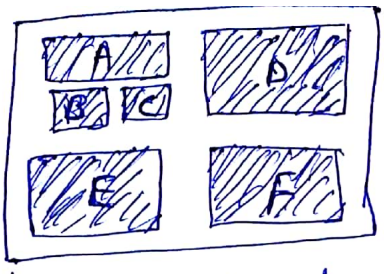
In (1) we can see space used to separate blocks as you often see in gaps between paragraphs or space between sections in a report.

In (2) there are clearly four main areas: A B C, D, E & F. Within one of these are three further areas, A, B, C which themselves are grouped as A on its own, followed by B & C together.

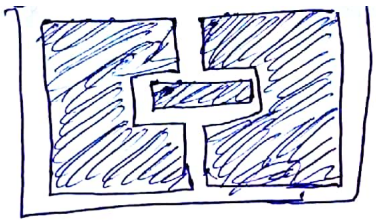
(3) We can see space used to highlight. This is a technique used frequently in magazine to highlight a quote or graphic.



(1) space to separate



(2) space to structure



(3) space to highlights

User Action and Control

① Entering Information:

Some of the most complicated & difficult screen layouts are found in form-based interfaces & dialog boxes.

In each case the screen consists not only of information presented to the user, but also of places for the user to enter information or select options.

Actually many of the ~~user~~ same layout issues for data presentation also apply to fields for data entry. Alignment is still important.

This is an occasion where right-justified text for the field labels may be best.

To encounter items in an appropriate order for the task, users are likely to ~~too~~ read from left to right

and top to bottom that a screen can be designed in an appropriate order for the task.

② Knowing what to do:

Some elements of a screen are passive, simply giving you information, others are active, expecting you to fill them in, or do something to them.

This is one of reasons for platforms and company style guides.

If everyone designs buttons to look the same & menus to look the same, then users will be able to recognize them when they see them.

It is important that the labels & icons on menus are also clear.

Again, standards can help for common actions such as save, delete or print.

for more system-specific actions, one needs to follow broader principles.

Eg: a button says "bold" does this represent the current state of a system or the action that will be performed if the button is pressed??

③ Affordances:

The psychological idea of affordance says that things may suggest by their shape and

other attributes what you can do to them: a handle affords pulling or lifting; a button affords pushing.

Note also that affordances are not intrinsic, but depend on the background & culture of users. Most computer-literate users will click on an icon. This is not because they go around pushing pictures in art galleries, but because they have learned that this is an affordance of such objects in a computer domain.

Appropriate appearance

① Presenting information

The way of presenting information on screen depends on the kind of information: text, numbers, maps, tables; on the technology available to present it: character display, line drawing, graphics, virtual reality

Different purposes require different representations. For more complex numerical data, we may be considering scatter graphs, histograms or 3D surfaces; for hierarchical structures, we may consider outlines or organization diagrams.

② Aesthetics and utility

Remember that a pretty interface is not necessarily a good interface. An interface should be aesthetically pleasing.

Indeed, good graphic design & attractive displays can increase users satisfaction & there improve productivity.

The beauty & utility may sometimes be at odds. The conflict between aesthetics & utility can also be seen in many 'well-designed' posters and multimedia systems.

The backdrop behind text must have low contrast in order to leave the text readable, this is often not the case & graphic designers may include excessively complex & strong backgrounds because they look good.

The results are impressive, perhaps even award winning, but completely unusable!

③ Making a mess of it: color and 3D:

One of the worst features in many interfaces is their appalling use of color.

This is partly because many monitors only support a limited range of primary colors and

partly because, as with the overuse of different fonts in word processors, the designer got carried ~~out~~ away.

An overuse of color can be distracting.

The increasing use of 3D effects in interfaces has posed a whole new set of problems for text and numerical information.

3D effects are excellent for presenting physical information and certain sorts of graphs, text presented in perspective can be very difficult to read.

(4) Localization / Internationalization:

The process of making software suitable for different languages & cultures is called localization or internationalization.

If you are working in a different country, you might see a document being would processed where the text of the document & the file names are in the local language, but all the menus & & the instructions are still in English.

Iteration and Prototyping

Because human situations are complex and designers are not infallible it is likely that our first design will not be perfect!

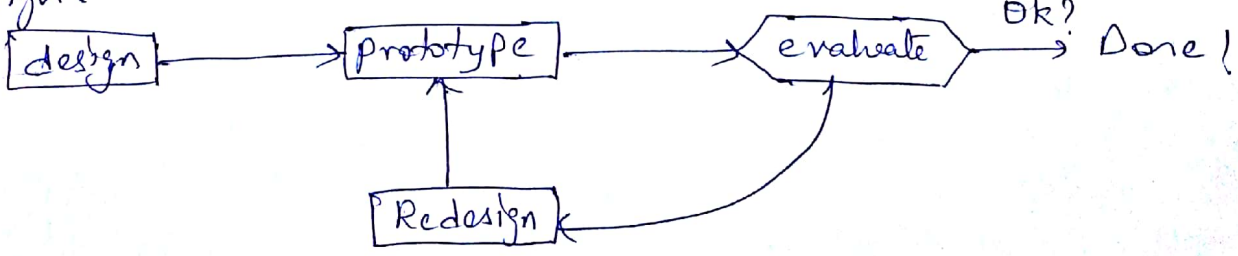
For this reason, almost all interaction design includes some form of iteration ideas.

Any of the prototypes, whether paper-based or purely software, can then be evaluated to see whether they are acceptable & whether there is improvement.

This sort of evaluation, intended to improve designs is called formative ~~cost~~ evaluation.

The result of evaluating the system will usually be a list of faults or problems & this is followed by a redesign exercise, which is then prototyped, evaluated.

So iteration & prototyping are the universally 'accepted 'best practice' approach for interaction design.



Role of prototyping

Prototyping is an example of what is known as a hill-climbing approach.

eg: Imagine you are standing somewhere in the open countryside. You walk uphill & keep going uphill as steeply as possible.

Eventually you will find yourself at a hill top. This is exactly how iterative prototyping works, you start somewhere, evaluate it to see how to make it better, change it to make it better & then keep on doing this until it can't get any better.

There are two things you need in order for prototyping methods to work:

(1) To understand what is wrong & how to improve

(2) A good start point.

HCI in Software Process

Software life cycle

A fundamental feature of software engineering, is that it provides the structure for applying & techniques to develop software systems.

The software life cycle is an attempt to identify the activities that occur in software development. In the development of a software product, we consider two main parties

1. the customer - who requires the use of the product.
2. the designer - are groups of people and some people can be both customer & designer.

Activities in the lifecycle

The activities of waterfall model of the software life cycle are

1. Requirements specification
2. Architectural design
3. Detailed design
4. Coding & unit testing
5. Integration & testing
6. Operation & maintenance

Activities in waterfall model of s/w life cycle

① Requirements specification

- In requirements specification, the designer & customer try to capture a description of what the eventual system will be expected to provide.
- Requirements specification involves eliciting information from the customer about the work environment, or domain, in which the final product will function.
- Requirements specification begins at the start of the product development.
  Though ~~they~~ the reqs are from the customer's perspective, if they are to be met by the software product they must be formulated in a language suitable for implementation.
- Requirements are usually initially expressed in the native language of the customer.

② Architectural design:

- The requirement specification concentrates on what the system is supposed to do.
- Architecture design concentrates on how the system provides the services expected from it.
- An architecture design performs the high-level decomposition of the system.
- It is not only concerned with the functional decomposition of the system but also it describes the interdependencies between separate components.

③ Detailed design :

The architectural design provides a decomposition of the system description that allows for isolated development of separate components which will later be integrated.

For those components that are not already available for immediate integration, the designer must provide a sufficiently detailed description so that they may be implemented in some programming language.

The detailed design is a refinement of the component descriptions provided by the architectural design.

④ Coding & unit testing

The detailed design for a component of the system should be in such a form that it is possible to implement it in some executable programming language.

After coding the component can be tested to verify that it performs correctly, according to some test criteria that were determined in earlier activities.

⑤ Integration & testing

Once enough components have been implemented and individually tested, they must be integrated as described in the architectural design.

Further, testing is done to ensure correct behavior and ~~acceptance testing~~ acceptable use of any shared resources.

It is only after acceptance of the integrated system that the product is finally released to the customer.

It may also be necessary to certify the final system according to requirements imposed by some outside authority.

The health & safety regulations and ISO-9241 provide ~~implement~~ impetus for designers to take seriously the HCI implications of their design

⑥ Maintenance

After product release, all work on the system is considered under the category of maintenance, until such time as a new version of the product demands a total redesign or the product is phased out entirely.

Maintenance involves the correction of errors in the system which are discovered after release

and the provision of the system services to satisfy requirements that were not realized during previous development.
Therefore, maintenance provides feedback to all of the other activities in the lifecycle.

Validation and verification

- Throughout the lifecycle, the design must be checked to ensure that it both satisfies the high-level requirements agreed with the customer and is also complete and internally consistent.
- These checks are referred to as validation and verification.
- Validation is designing "the right thing"
- Verification is designing "the ~~proda~~ thing right"
- Verification of a design will most often occur within a single life-cycle activity or between two adjacent activities.
- Validation of a design demonstrates that within the various activities the customer's requirements are satisfied.

Usability Engineering

One approach to user-centered design has been the introduction of explicit usability engineering goals into the design process.

The emphasis for usability engineering is in knowing exactly what criteria will be used to judge a product for its usability.

The ultimate test of a product's usability is based on measurement of users experience with it.

In relation to the software life cycle, important feature of usability engineering is usability specification forming part of the requirements specification.

For each attribute, 6 items are defined to form the usability specification of that attribute.

Fig: Sample usability specification for undo with a vcr

Attribute	
Measuring concept:	Undo an erroneous programming sequence
Measuring method:	Number of explicit user actions to undo current program.
Now level:	No current product allows such an undo
Worst ~~case~~ level:	As many actions as it takes to program in mistake.

Planned level : A maximum of two explicit user actions

Best case : One explicit cancel action.

The backward recoverability attribute is defined in terms of a measuring concept which makes the abstract attribute more concrete by describing it in terms of the actual product.

The measuring method states how the attribute will be measured.

The now level indicates the value for the measurement with the existing system.

The Worst case value is the lowest acceptable measurement for the task

The planned level is the target for the design and the best case is the level which is agreed to be the best possible measurement given the current state of development tools & technology.

Problems with usability engineering

- Usability specification requires level of detail that may not be specified.
- possible early in design satisfying a usability specification.
- does not necessarily satisfy usability.

Design Rationale

Design rationale is information that explains why a computer system is the way it is including its structural or architectural description & its functional or behavioral description.

Benefits of design rationale:

- Communication throughout life cycle
- reuse of design knowledge across products
- enforces design discipline
- presents arguments for design trade-offs
- organizes potentially large design space.
- capturing contextual information.

Types of design rationale

1) Process-oriented design rationale
2) structure-oriented design rationale

Process-oriented design rationale:

- Process-oriented preserves order of deliberation and decision-making.
- Much of the work on design rationale is based on Rittel's "issue-based information system (IBIS), a style for representing design & planning
- & IBIS is a graphical version. (Graphical IBIS)

Main elements are

① issue - hierarchical structure with one 'root' issue

② position - potential resolutions of an issue

③ arguments - modify the relationship between positions & issues.

A root issue is identified which represents the main problem or question that the argument is addressing.

Various positions ~~are put~~ as potential resolutions for the root issue, and are depicted as descendants in the IBIS hierarchy directly connected to the root issue.

arguments which modify the relationship between issue and position.

The hierarchy grows as secondary issues are raised which modify the root issue in some way. Each of these secondary-issues is in turn expanded by positions and arguments, further sub-issues, & so on.

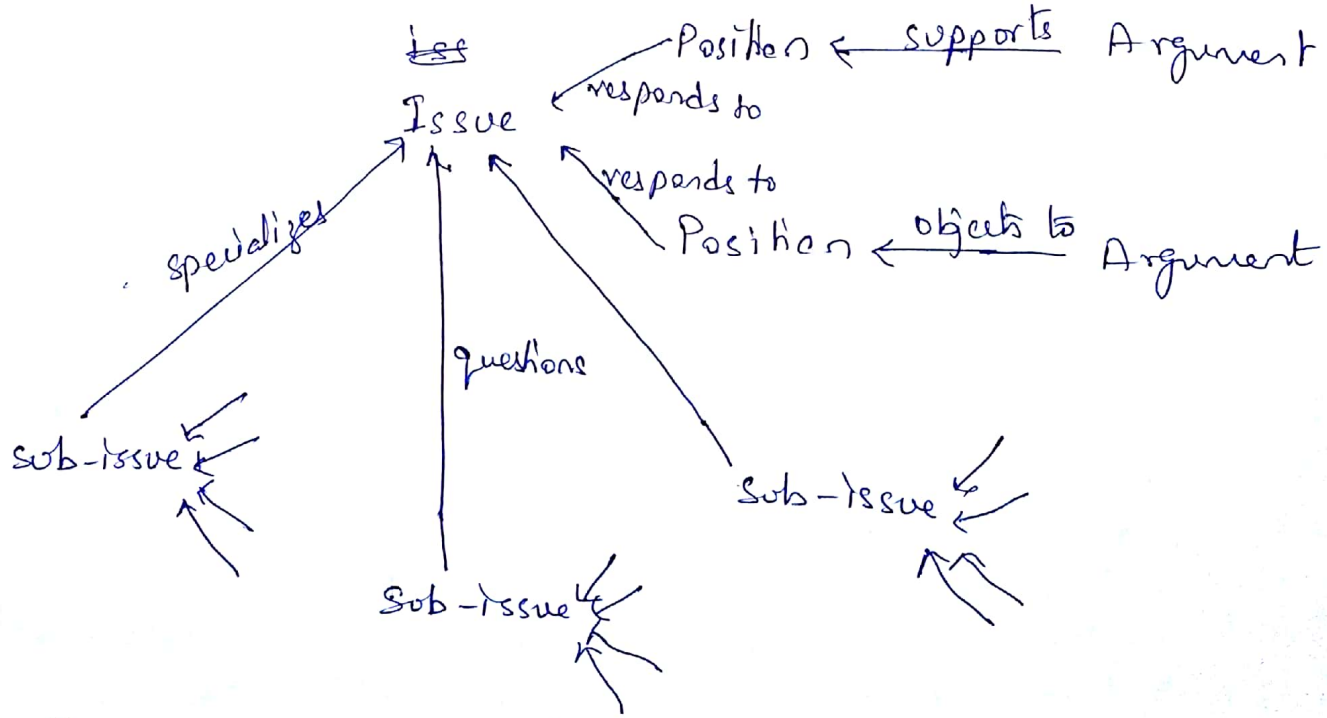


fig: The structure of a gIBIS design rationale

Structure-oriented design rationale (eg: Design space analysis).

Design space analysis emphasizes a post hoc structuring of the space of design alternatives that have been considered in a design project.

Design space analysis approach, embodied (will be) in the Questions, options & criteria (QOC) notation.

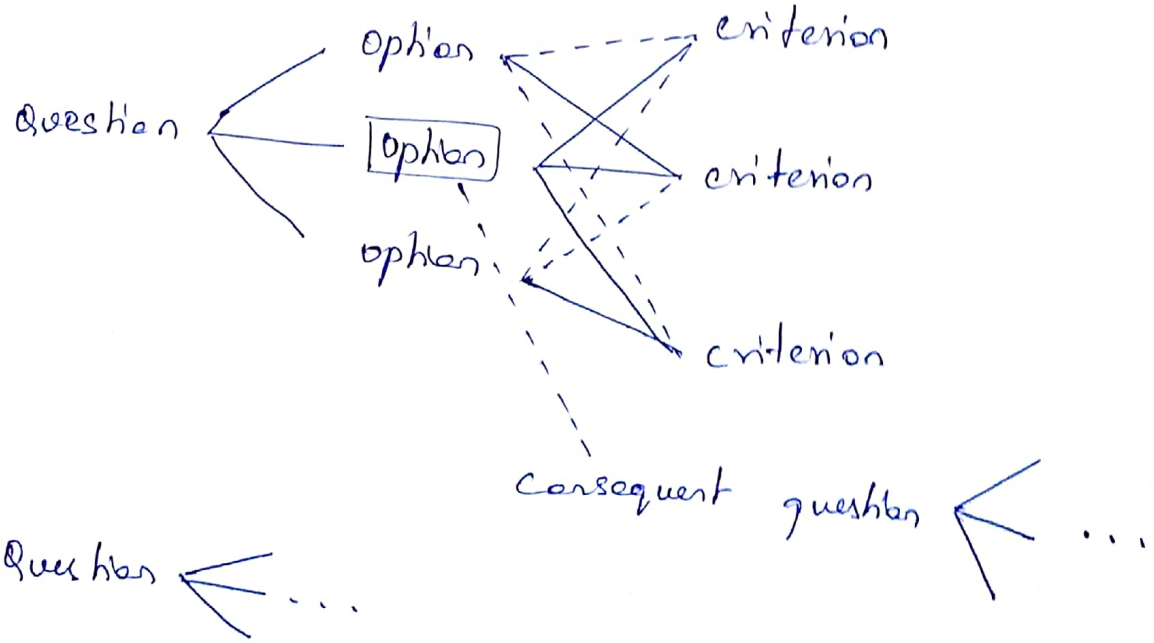
QOC notation is

Questions (& sub-questions) - represents major issues of a design

options - provide alternative solutions to the questions

criteria - the means to assess the options in order to make a choice.

fig: The QOC notation



- The design space is initially structured by a set of questions representing the major issues of the design.
- Since design space analysis is structure oriented, it is not so important that the questions recorded are the actual questions asked during design meetings.
- Questions in a design space analysis are similar to issues in IBIS except in the way they are captured.
- Options provide alternative solutions to the question. They are assessed according to some criteria in order to determine the most favorable option.
- An option which is favorably assessed in terms of a criterion is linked with a solid line, whereas negative links have a dashed line.
- The most favorable option is boxed in the diagram

Psychological design rationale

- The purpose of psychological design rationale is to support task-artifact cycle of design activity.
- Aims to make explicit consequences of design for users
- designers identify tasks system will support.
- scenarios are suggested to test task.
- users are observed on system.
- psychological claims of system made explicit.
- negative aspects of design can be used to improve next iteration of design.

Design Rules

Design rules, which are rules a designer can follow in order to increase the usability of the eventual software product.

- Rules classified into two dimensions, based on the rule's authority and generality.

Different types of design rules

1) Principles - are abstract design rules, with high generality and low authority.
2) Standards - are specific design rules, high in authority and limited in application.
3) guidelines - ~~the~~ tend to be lower in authority and more general in applications.

- Authority: an indication of whether or not the rule must be followed in design or whether it is only suggested.

generality: means whether the rule can be applied to many design situations or whether it is focussed on a more limited application situation.

Principles to support usability

Principles are divided into 3 main categories:

1) Learnability - the ease with which new users can begin effective interaction & achieve ~~maximum~~ Maximal performance.
2) Flexibility - the multiplicity of ways in which the user and system exchange information.

3) Robustness - the level of support provided to the user in determining successful achievement and assessment of goals.

Learnability

Learnability concerns the features of the interactive system that allow novice users to understand how to use it initially and then how to attain a maximal level of performance.

principles affecting learnability are

① Predictability - support for the user to determine the effect of future action based on past interaction history.

② Synthesizability - support for the user to assess the effect of past operations on the current state.

③ Familiarity - The extent to which a user's knowledge and experience in other real-world or computer-based domains can be applied when interacting with a new system.

④ Generalizability - Support for the user to extend knowledge of specific interaction ~~what~~ within and across applications to other similar situations.

⑤ Consistency - Likeness in input-output behavior arising from similar situations or similar task or objectives.

Flexibility

Flexibility refers to the multiplicity of ways in which the end-user and the system exchange information.

Principles affecting flexibility are

(1) Dialog initiative - allowing the user freedom from artificial constraints on the input dialog imposed by the system.

(2) Multi-threading - Ability of the system to support user interaction pertaining to more than one task at a time.

(3) Task migratability - The ability to pass control for the execution of a given task so that it becomes either internalized by the user or the system or shared between them.

(4) Substitutivity - Allowing equivalent values of input and output to be arbitrarily substituted for each other.

(5) Customizability - Modifiability of the user interface by the user or the system.

Robustness

In a work or task domain, a user is engaged with a computer in order to achieve some set of goals.

The robustness of that interaction covers features that support the successful achievement and assessment of the goals.

Principles affecting robustness are

① Observability - Ability of the user to evaluate the internal state of the system from its perceivable representation.

② Recoverability - Ability of the user to take corrective action once an error has been recognized.

③ Responsiveness - how the user perceives the rate of communication with the system.

④ Task Conformance - The degree to which the system services support all of the tasks the user wishes to perform and in the way that the user understands them.

2. Standards

Standards for interactive system design are usually set by national or international bodies to ensure compliance with a set of design rules by a large community.

Standards can apply specifically to either the hardware or the software used to build the interactive system.

ISO standard 9241 defines the following usability specifications:

① Usability - The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments.

② Effectiveness - The accuracy & completeness with which specified users can achieve specified goals in particular environments.

③ Efficiency - The resources expended in relation to the accuracy & completeness of goals achieved.

④ Satisfaction - The comfort & acceptability of the work system to its users and other people affected by its use.

3. Guidelines

The majority of design rules for interactive systems are suggestive and more general guidelines.

The basic categories of the Smith & Mosier guidelines are

1) Data entry
2) Data display
3) Sequence control
4) User guidance
5) Data transmission
6) Data protection