INTERACTION DESIGN BASICS

- Design
- The design process
- Users
- Scenarios
- Navigation
- Iteration and prototypes

UNIT WISE SYLLABUS - HCI

Unit – III	Design Process:	9
Interaction design basics: Introduction – The process of design – User focus – Scenarios – Navigation design – Screen design and layout – Iteration and Prototyping – HCI in the software process: Introduction – The software life cycle – Usability engineering – Iterative design and prototyping – Design rationale –Golden rules and heuristics – HCI patterns.		

INTERACTIONS AND INTERVENTIONS

design interactions not just interfaces

not just the immediate interaction

designing interventions not just artefacts

not just the system, but also …

- documentation, manuals, tutorials
- what we say and do as well as what we make

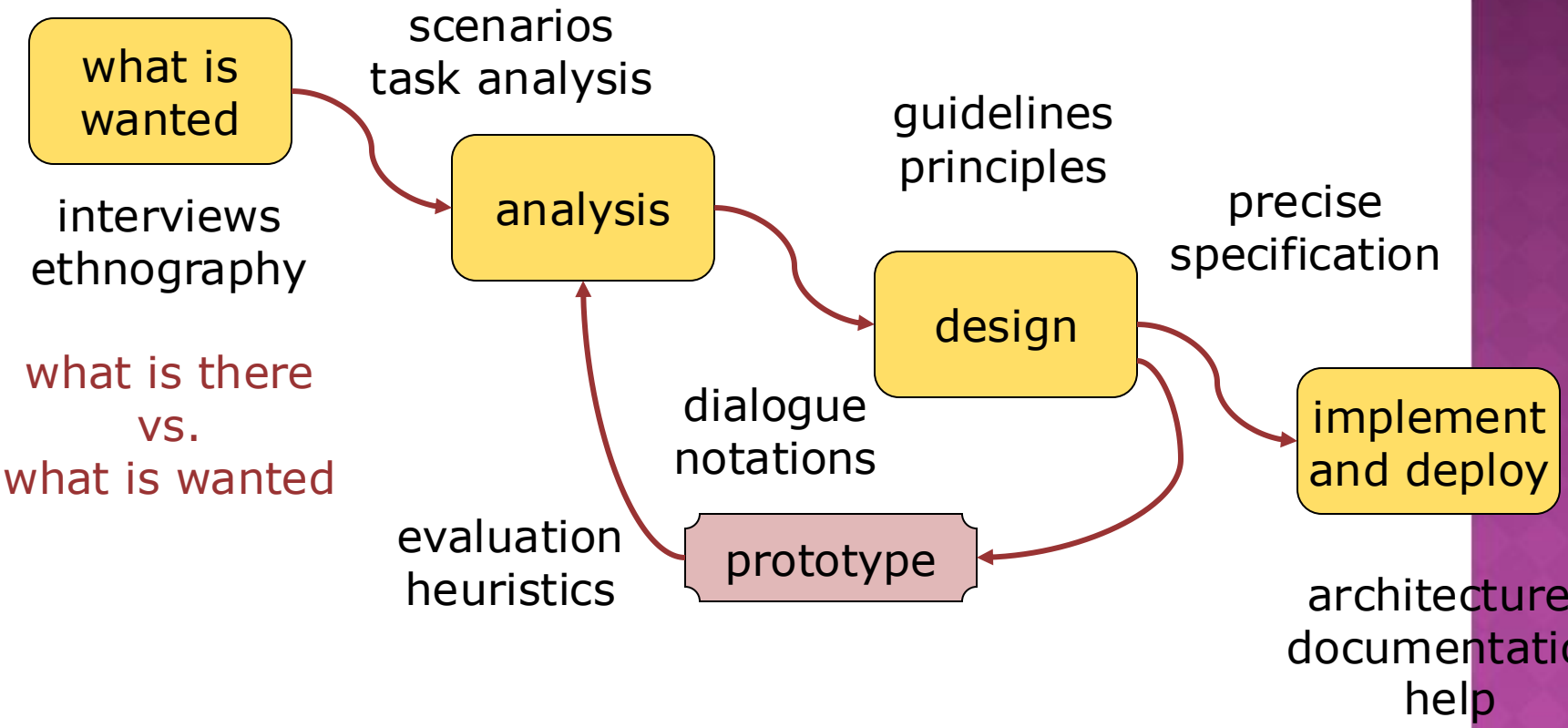
THE GOLDEN RULE OF DESIGN

- Golden Rule of Design: "**understand your materials**"
- For **Human-Computer Interaction** the obvious materials are the **human** and the **computer**.
  - understand ***computers***
    - limitations, capacities, tools, platforms
  - understand ***people***
    - psychological, social aspects, human error.

CORE OF INTERACTION DESIGN

- The **core of interaction design** is:
  - **put the user first**
  - **keep the user in the center** and
  - **remember the user at the end**

THE PROCESS OF DESIGN



STEPS

- PHASE 1:requirements
  - what is there and what is wanted …
- PHASE 2:analysis
  - ordering and understanding
- PHASE 3:design
  - what to do and how to decide
- PHASE 4:iteration and prototyping
  - finding what is really needed!
- PHASE 5:implementation and deployment
  - making it and getting it out there

PHASE 1: REQUIREMENTS - WHAT IS WANTED

- The first stage is establishing **what exactly is needed.**
- 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 and objects that they work with,**
  - **observing them directly.**
  - **ethnography, (field work)**

PHASE 2: ANALYSIS

- The results of observation and interview **need to be ordered in some way** to bring out **key issues** and communicate with later stages of design.
- **Techniques** used
  - **Scenarios**
  - **rich stories of interaction**

PHASE 3: DESIGN

- This is a central stage when you move from **what you want**, to **how to do it.**
- **Techniques** used
  - **Rules**
  - **Guidelines** and
  - **Design principles**

LOOP: ITERATION AND PROTOTYPING

- **We need to evaluate a design** to see how well it is working and where there can be improvements.
- Evaluation **Techniques** used
  - **Cognitive walkthrough**
  - **Heuristic evaluation**
  - **Review based**
  - **Model based**
  - **Experiment**
  - **Interviews**
  - **Questionnaire**
  - **Think aloud**
  - **Protocol analysis**
  - **Post-task walkthrough**
  - **Eye tracking**
  - **Physiological measurement**

LOOP: ITERATION AND PROTOTYPING

- **We need to evaluate a design** to see how well it is working and where there can be improvements.
- Evaluation **Techniques** used
  - **Cognitive walkthrough - user registration**
  - **Heuristic evaluation - experts review**
  - **Review based - existing analysis**
  - **Model based - predictive model**
  - **Experiment - checking on 2 version**
  - **Interviews - analyse satisfaction**
  - **Questionnaire - survey report**
  - **Think aloud - think of requirement during process**
  - **Protocol analysis - changes after think aloud**
  - **Post-task walkthrough - discuss risk factors**
  - **Eye tracking - record analysis**
  - **Physiological measurement - health related concern**

PHASE 4: IMPLEMENTATION AND DEPLOYMENT

- Finally, **create it** and **deploy it.**
- This will involve
  - writing code,
  - making hardware,
  - writing documentation
  - manuals –
- Everything that goes into a real system that can be given to others.
- **Techniques** used
  - **software architectures**
  - **web interfaces**
- **Trade offs between length of design period and quality of final design**

USER FOCUS

- **Initial state of interaction design-intended users**
- **Know your users**
- **Single user** and **Multi-user**.
- It is important to be aware that there is **rarely one user of a system.**
- Example:
  - Think about a **stock control system.**
  - The **warehouse manager** queries the system to find out **how many six-inch nails are in stock**

MULTI-USER

- Perhaps a **salesperson** has been asked to deliver 100,000 six-inch nails within a fortnight and wants to know if the company is able to fulfill the order in time.
- The **auditors** want to produce a valuation of company assets including stock in hand.
- the **assistant warehouse manager** needs to update the stock levels while his boss is on holiday.
- Over time **many people** are affected directly or indirectly by a system.
- The people involved in the system are called **stakeholders**.

- *A system will be successful if its :*
  - *Usable*
  - *Useful*
  - *used*

USER FOCUS

USER FOCUS-SCENARIOS

- Expectations were too high.
- It was a product not a solution.
- No clear need or target market.
- It was an invention rather than an innovation.

USER FOCUS

who are they

probably not similar to us

talk to them

watch them

use your imagination

USER FOCUS-HOW TO GET TO KNOW THE USERS

- who are they
  - Young/old
  - Experienced/freshers
  - Ex:stock, word processor users

USERS VIEW

- probably not similar to us
  - Assumed design of self interest and abilities
  - Ex: contrast between male and female developers

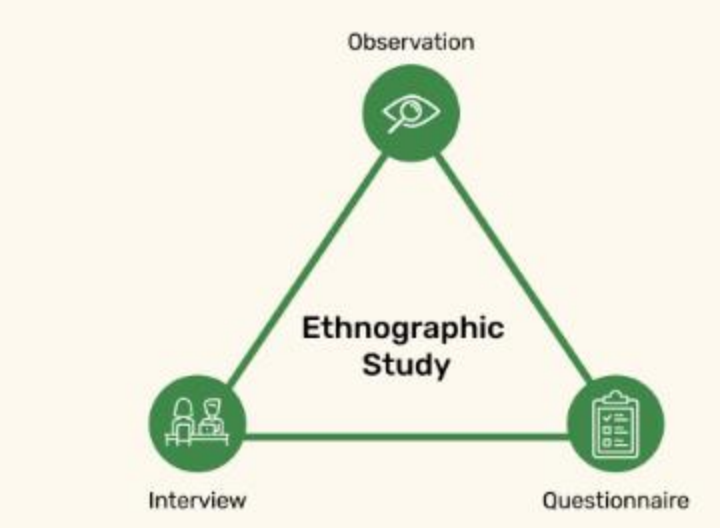
TALK TO THEM

- Get to know about the users
- Structured interviews
- Open ended discusions
- Participatory design-users for design process

HOW TO KNOW THE USERS

- Professional & Trainer-can do, can train
- Different kinds of knowledge
- Watch them.
  - **Watch the people** as well as **hear their view.**
  - **Like,**
    - **sitting** and **taking notes** of their routine,
    - **watching particular activities**,
    - using a **video camera** or **tape recorder**.
  - **Informal manner**
  - Developed method
    - **ethnography**
    - **contextual inquiry**.

ETHANOGRAPHY AND CULTURAL PROBES



ETHANOGRAPHY AND CULTURAL PROBES

- Watching people and being present

- Cultural probes
  - No intrusion
  - Added method of ethanography
  - Designed to record comments
- It serves as a means of **gathering inspirational data about people's lives, values and thoughts**.
- The probes are **small packages** that can include any sort of **artifact** (like a **map, postcard, camera or diary**) along with evocative tasks, which are given to participants to allow them to record specific events, feelings or interactions.

IMAGINATION

- Many users throughout design exercise
- Costly + hard + too many
- Kind of method acting
- Useful method to produce user focused design – persona
- Persona- rich picture of imaginary person in core group

USE YOUR IMAGINATION

- Use your imagination
  - **if you cannot involve actual users** you can at least **try to imagine their experiences**.
  - One method that has been quite successful in helping design teams produce user focused designs is the **persona**.
  - A persona is a **rich picture of an imaginary person who represents your core user group**.

„Time is the soul of business"

Kuzey Saydam

Business Owner, Manufacturer

Age: 42

SMART CREATIVE SOCIAL FOCUSED

Location Istanbul, Turkey

Biography

Into business almost his whole life, Kuzey started to manufacture the first products when he was 14. He values transparency and strong relationships, wastes no time, travels a lot to build new partnerships to grow his business.
He is frustrated by the fact that no matter how valuable a potential partner is, he spends hours looking for new partners. He also wants to present his products online cheaply and conveniently.

Values

- Time
- Reliability
- Relationships
- Innovation
- Balance
- Responsibility

Personality

Introvert	Extrovert
Analytical	Creative
Loyal	Fickle
Passive	Active

Goals

- Grow his business
- Sell more
- Build new relationships

Pain points

- No online space to sell his products
- Spends too much time looking for new partners

EXAMPLE TO CREATE USER PERSONA

Betty is 37 years old. She has been Warehouse Manager for five years and has worked for Simpkins Brothers Engineering for 12 years. She didn't go to university, but has studied in her evenings for a business diploma. She has two children aged 15 and 7 and does not like to work late. She did part of an introductory in-house computer course some years ago, but it was interrupted when she was promoted and could no longer afford to take the time. Her vision is perfect, but her right-hand movement is slightly restricted following an industrial accident three years ago. She is enthusiastic about her work and is happy to delegate responsibility and take suggestions from her staff. However, she does feel threatened by the introduction of yet another new computer system (the third in her time at SBE).

Figure 5.3 Persona – a rich description of Betty the Warehouse Manager

SCENARIOS

SCENARIOS

- Scenarios are **stories** for design: **rich stories** of interaction.
- Simplest design representation
- Most flexible and powerful
- Some scenarios are **quite short**:
  - 'the user intends to press the "save" button, but accidentally presses the "quit" button so loses his work'.

EXAMPLE SCENARIO FOR PROPOSED MOVIE PLAYER

Brian would like to see the new film *Moments of Significance* and wants to invite Alison, but he knows she doesn't like 'arty' films. He decides to take a look at it to see if she would like it and so connects to one of the movie-sharing networks. He uses his work machine as it has a higher bandwidth connection, but feels a bit guilty. He knows he will be getting an illegal copy of the film, but decides it is OK as he is intending to go to the cinema to watch it. After it downloads to his machine he takes out his new personal movie player. He presses the 'menu' button and on the small LCD screen he scrolls using the arrow keys to 'bluetooth connect' and presses the 'select' button. On his computer the movie download program now has an icon showing that it has recognized a compatible device and he drags the icon of the film over the icon for the player. On the player the LCD screen says 'downloading now', with a per cent done indicator and small whirling icon.

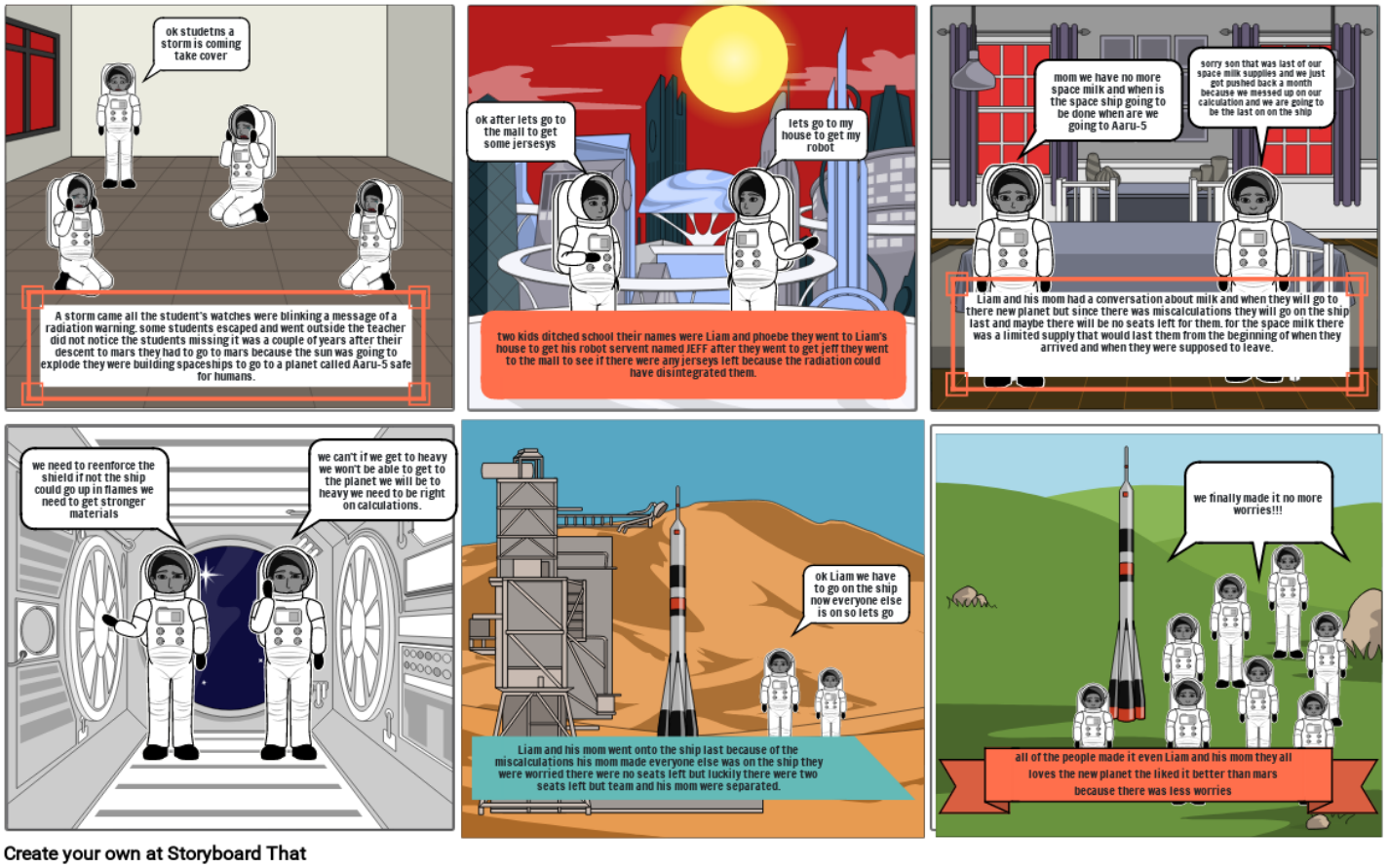
During lunchtime Brian takes out his movie player, plugs in his earphones and starts to watch. He uses the arrow keys to skip between portions of the film and decides that, yes, Alison would like it. Then he feels a tap on his shoulder. He turns round. It is Alison. He had been so absorbed he hadn't noticed her. 'What are you watching', she says. 'Here, listen', he says and flicks a small switch. The built-in directional speaker is loud enough for both Brian and Alison to hear, but not loud enough to disturb other people in the canteen. Alison recognizes the film from trailers, 'surprised this is out yet' she says. 'Well actually . . .', Brian confesses, 'you'd better come with me to see it and make an honest man of me'. 'I'll think about it', she replies.

SCENARIO

- More detailed-events seems real
- Instead of Plain text
  - Can have sketches,simulated screenshots
- Sketches and pictures-storyboards
- Scenarios act as script – design+physical artifacts

STORYBOARDS

- **sketches** + **pictures** – ***storyboards***
- similar to the techniques used in film making to envisage plot-lines.



EXAMPLE - DIGITAL SWISS ARMY KNIFE-DESIGN + PHYSICAL ARTIFACTS

- LCD screen + toothpick as stylus
- Connects to internet
- Tip:
  - open the stone remover
  - now push the blade..
- Refinement :Voice instruction could be better

- Scenario benefits:
  - Make us to think about design in detail
  - Notice potential problems before
  - Verify design + proposed architecture implementation

SCENARIOS

- scenarios can also be used to:
  - **Communicate with others**
    - **Designers**
    - **users**
  - **Validate other models**
    - **Task models/ dialog and navigation models**
  - **Express dynamics**

    Screenshots- picture of system looks

    but not the behaviour

Patterns of interaction

complex with networks / hierarchies

LINEARITY

Scenarios are linear – one linear single path through system among all potential interaction

Pros:

- time is linear→ life and time were linear
- easy to understand -stories and narrative are natural
- concrete -errors less likely

Cons:

- No alter-no choice, no branches, no special conditions
- miss the unintended

- Solution:
  - use several scenario
  - use several methods

SCENARIOS

- **Scenarios** are a **resource** that can be **used** and **reused** throughout the **design process.**
- Scenarios are
  - Helping- **what is wanted**,
  - suggesting -**how users will deal** with the potential design,
  - checking - **proposed implementations** will work
  - Generating- **test cases** for final evaluation.

NAVIGATION DESIGN

NAVIGATION DESIGN

- consider,
  - navigate the **main screens** or **modes**
  - **interconnect**
  - interactsionwith the **wider environment**?
- Interaction at several levels.
  - Widgets -how to use them for a particular selection or action
  - Screens or windows-logical grouping of buttons
  - Navigation within the application -when a button pressed
  - Environment

Table 5.1 Levels of interaction

PC application	Website	Physical device
Widgets	Form elements, tags and links	Buttons, dials, lights, displays
Screen design	Page design	Physical layout
Navigation design	Site structure	Main modes of device
Other apps and operating system	The web, browser, external links	The real world!

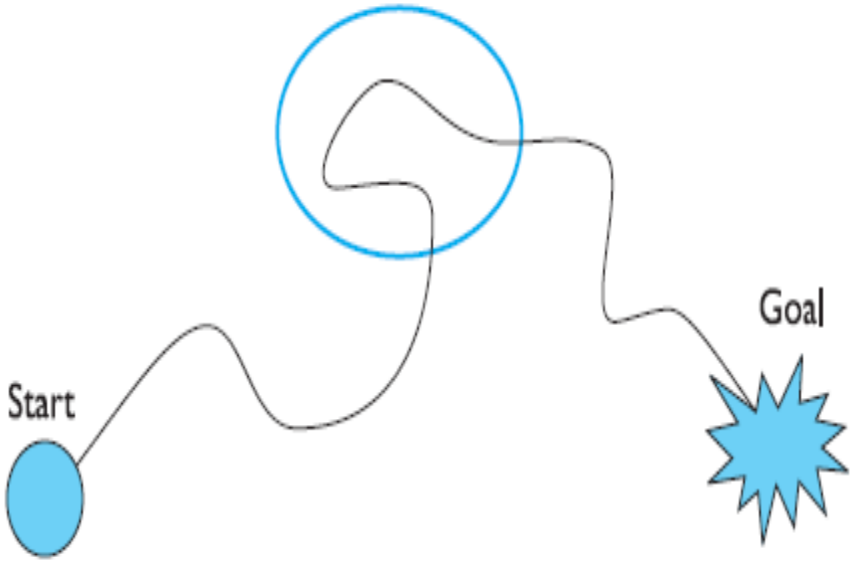
NAVIGATION DESIGN

- **Structure of application**
  - Think of -Who/how/what will they do
- **two main kinds of issue:**
  - **local structure**
    - looking from one **screen** or **page** out
  - **global structure**
    - **structure of site, movement between screens.**

local structure – single screen
global structure – whole site

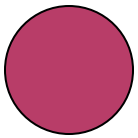
LOCAL STRUCTURE

- from one screen looking out
- Pressing all buttons-partial knowledge
- Deviate from efficient short route

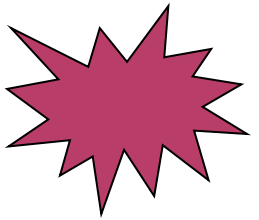


GOAL SEEKING

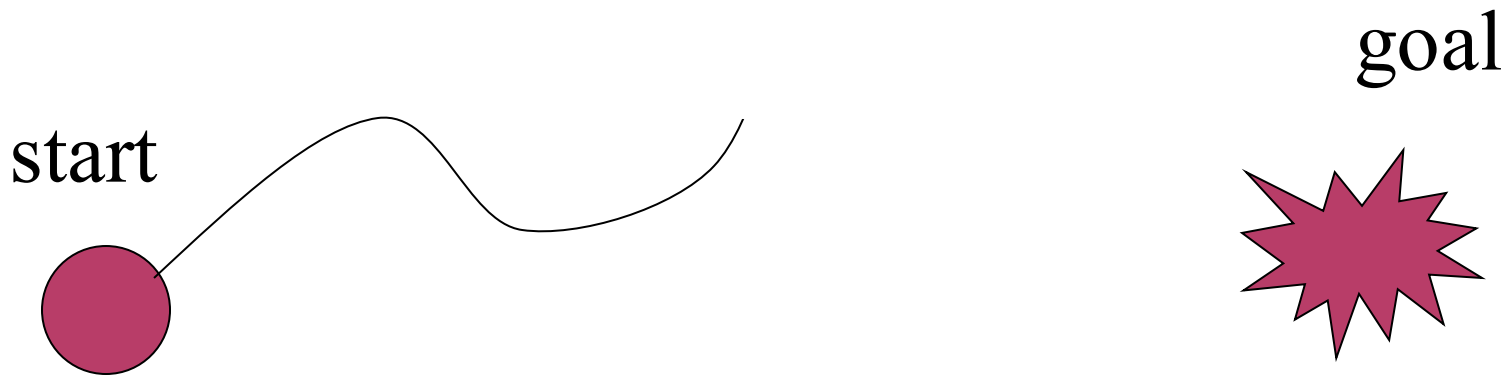
start



goal

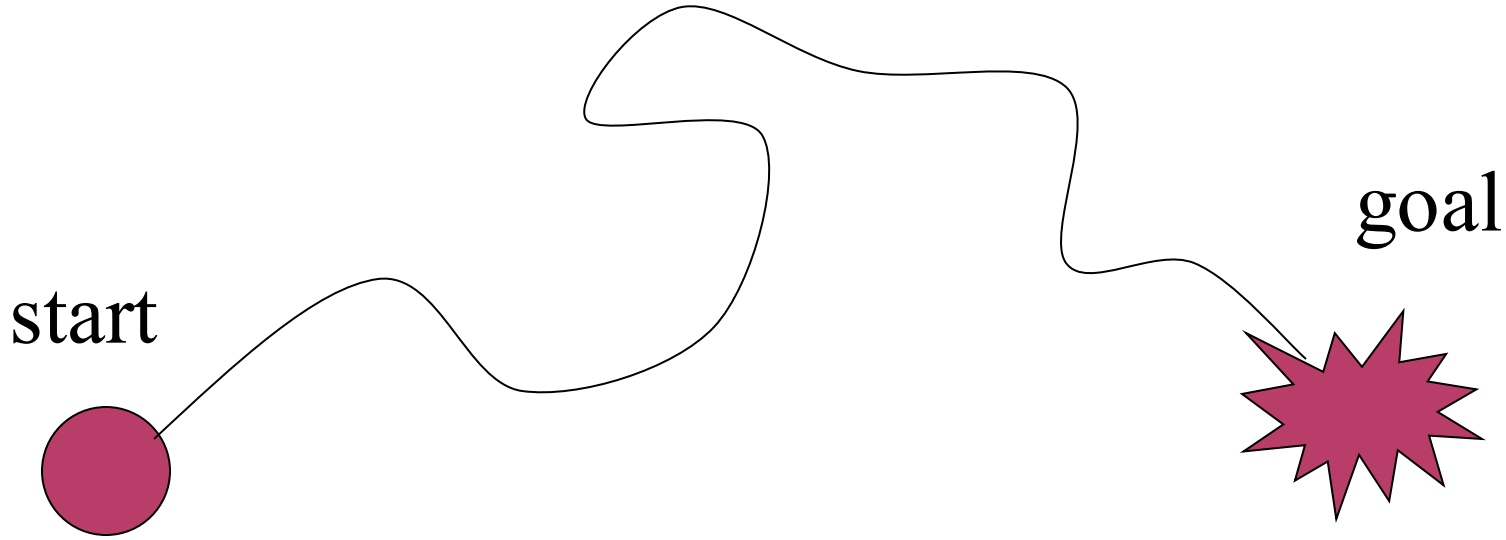


GOAL SEEKING



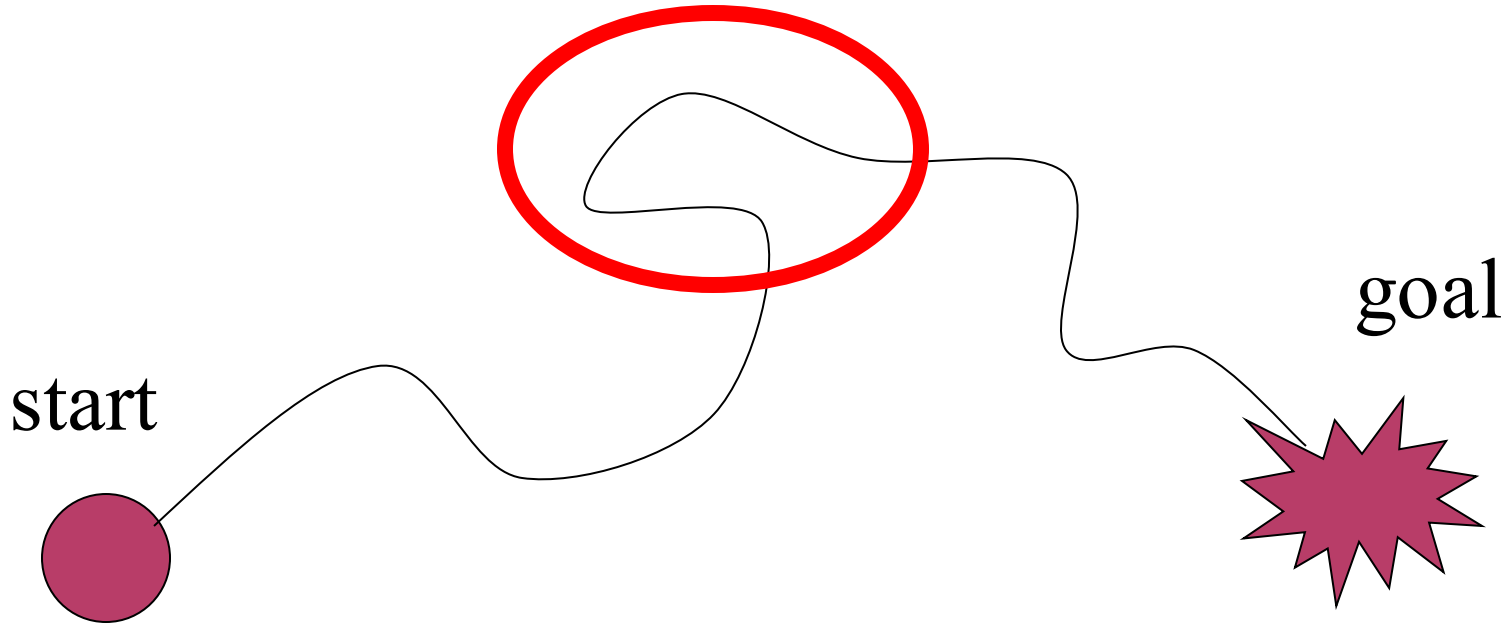
progress with local knowledge only ...

GOAL SEEKING



… but can get to the goal

GOAL SEEKING



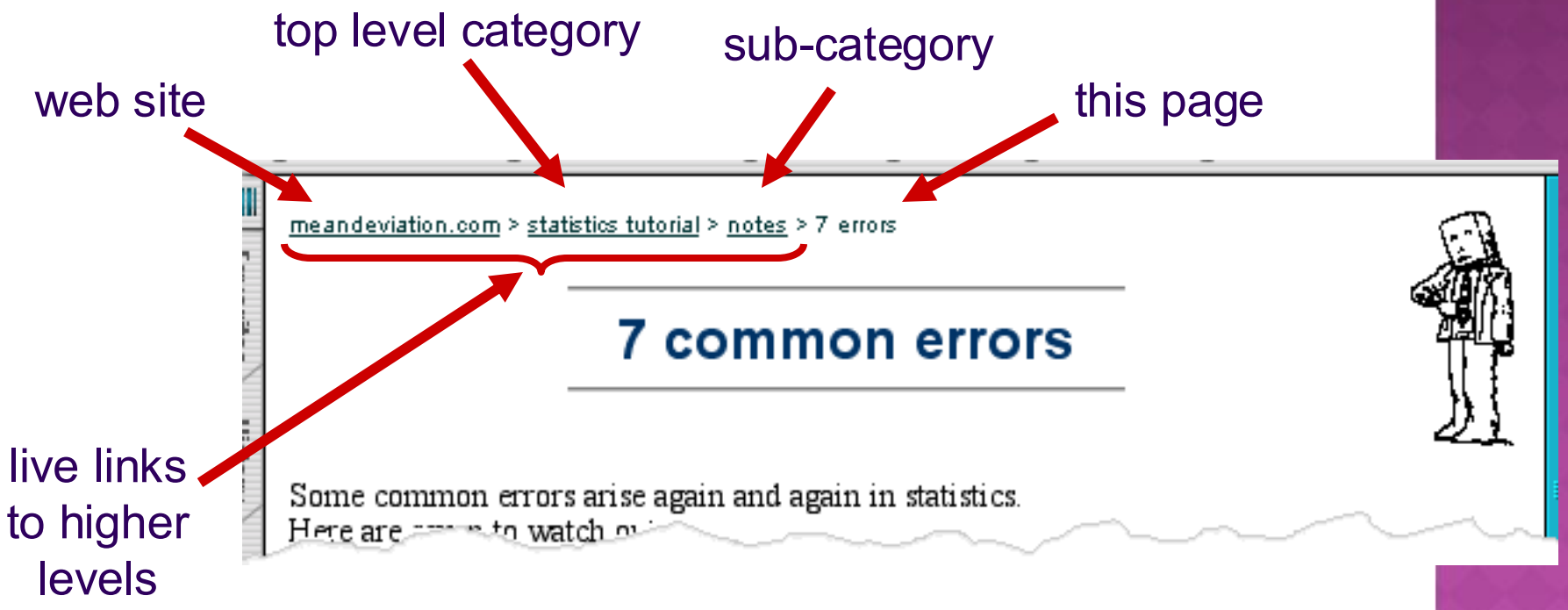
… try to avoid these bits!

FOUR GOLDEN RULES

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

WHERE YOU ARE - BREADCRUMBS

shows path through web site hierarchy



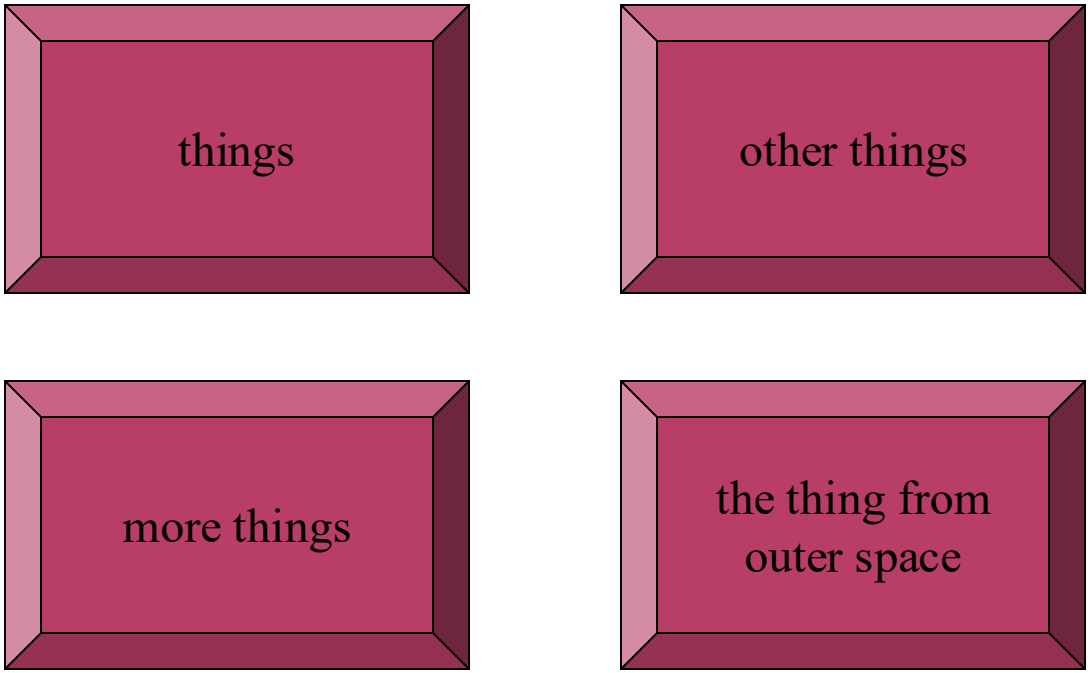
WHAT YOU CAN DO - LINKS

- **what can be pressed** or clicked to go somewhere or do something.
- Trade off-appearance and ease of use

WHERE YOU ARE GOING - BUTTONS PRESS

- when you click a button or ***what will happen***.

EXAMPLE - BEWARE THE BIG BUTTON TRAP



- where do they go?
  - lots of room for extra text!
  - Public info. System-wrong screeen navigation

MODES

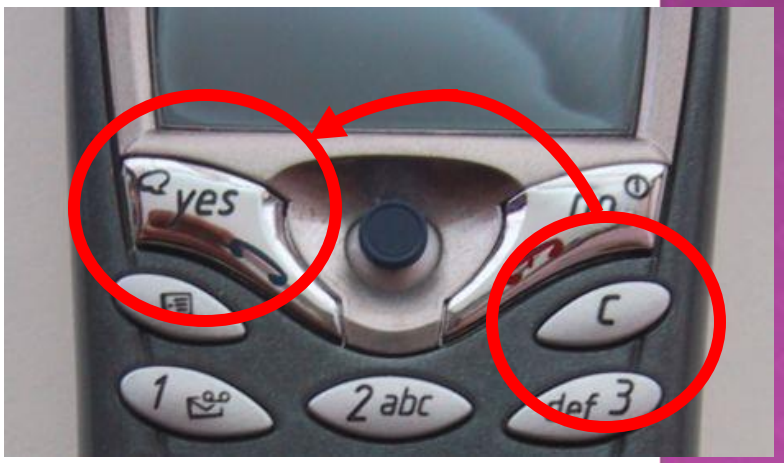
- Different context change interpretation of commands called modes
- X
  - enter me into text-normal typing mode
  - Exit mode
- Modes have less problem in windowed system

EXAMPLE - MODES

- lock to prevent accidental use …
  - remove lock - **‘c’ + ‘yes’** to confirm
  - frequent practiced action
- if lock forgotten
  - in pocket ‘yes’ gets pressed
  - goes to phone book
  - in phone book …

    ‘c’ - delete entry

    ‘yes’ - confirm



WHAT YOU'VE DONE - FEEDBACK AND HISTORY

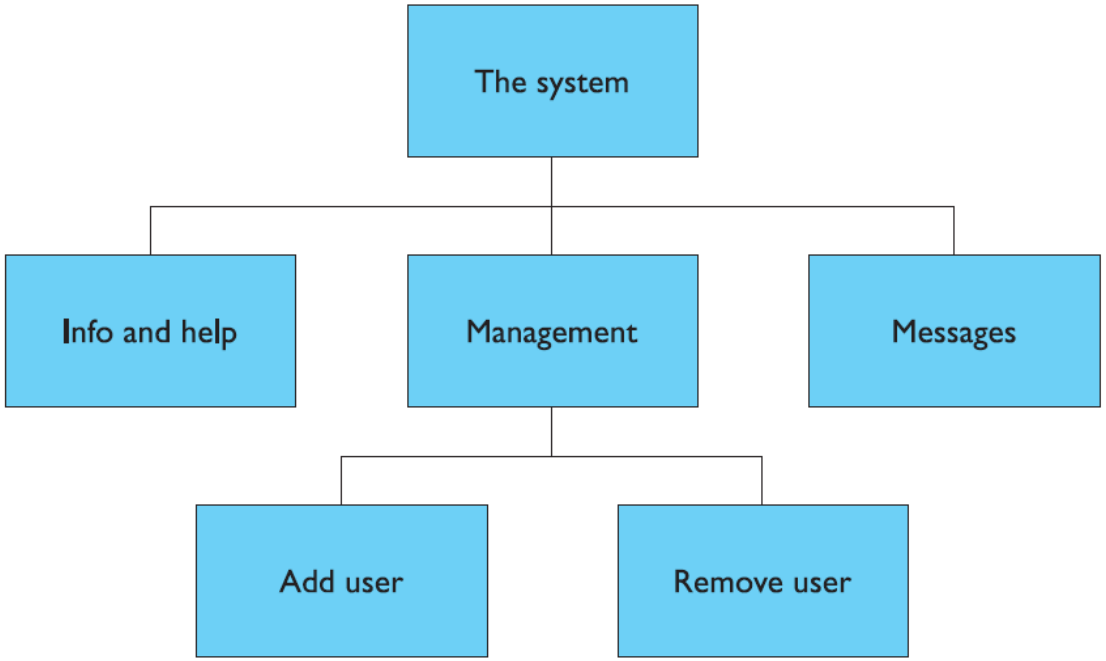
- The **system** needs to give some ***feedback***

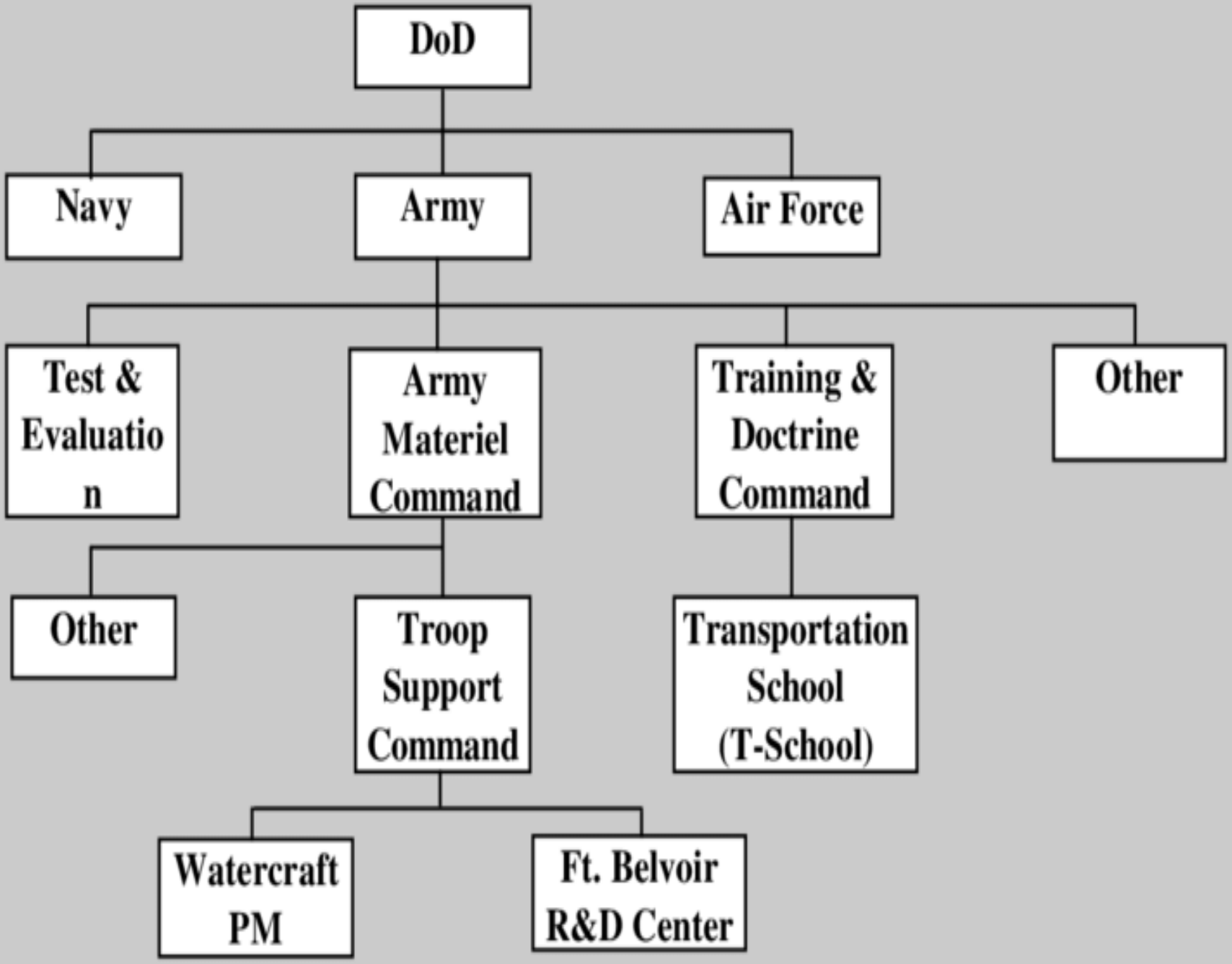
- **web browsers**
  - offer a **history** system
  - '**back**' button
    - that keeps a list of recently visited pages.

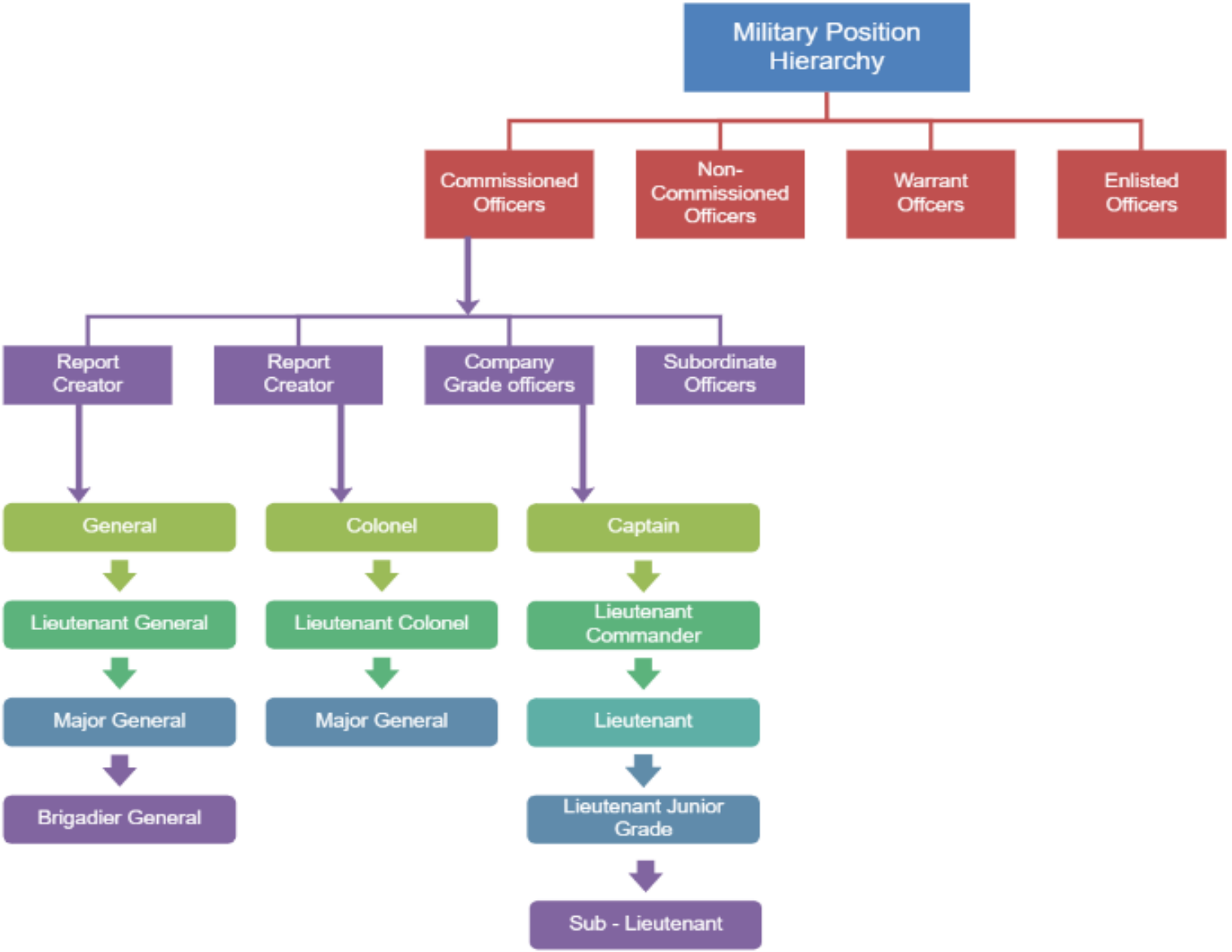
GLOBAL STRUCTURE

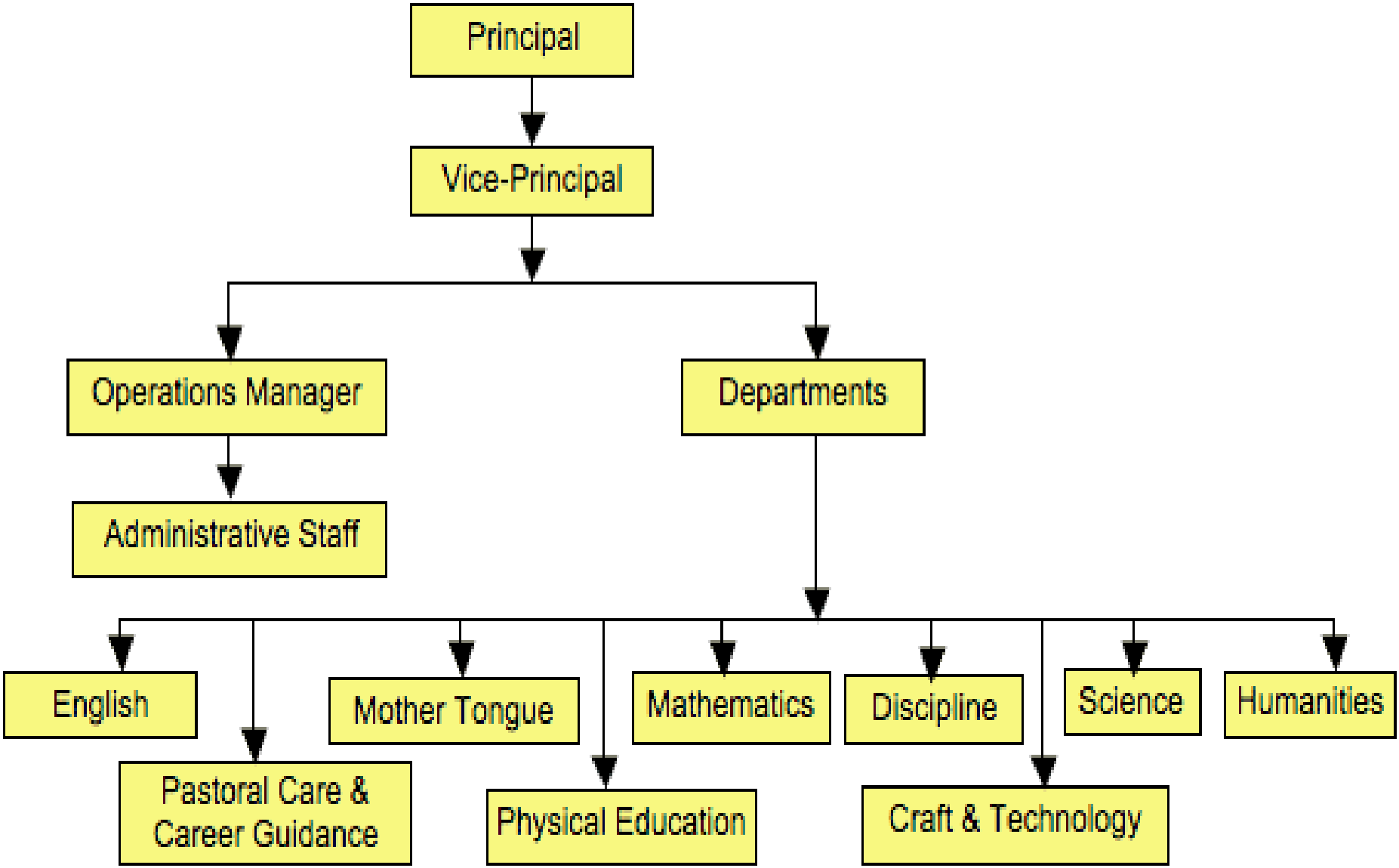
- **overall structure** of an application
  - **between screens** within the application.
- way to organize a system- **hierarchy**.
- The **hierarchy** links **screens, pages** or **states** in logical groupings.
- Two Choices:
  - **Hierarchical diagrams**
  - **Networked diagrams**

HIERARCHICAL DIAGRAMS




DoD
Navy
Army
Air Force
Test & Evaluation
Army Materiel Command
Training & Doctrine Command
Other
Other
Troop Support Command
Transportation School (T-School)
Watercraft PM
Ft. Belvoir R&D Center


Military Position Hierarchy
Commissioned Officers
Non-Commissioned Officers
Warrant Offcers
Enlisted Officers
Report Creator
Report Creator
Company Grade officers
Subordinate Officers
General
Lieutenant General
Major General
Brigadier General
Colonel
Lieutenant Colonel
Major General
Captain
Lieutenant Commander
Lieutenant
Lieutenant Junior Grade
Sub - Lieutenant


Principal
Vice-Principal
Operations Manager
Departments
Administrative Staff
English
Pastoral Care & Career Guidance
Mother Tongue
Physical Education
Mathematics
Discipline
Craft & Technology
Science
Humanities

HIERARCHICAL STRUCTURE

- **deep hierarchies are difficult to navigate**, so
  - it is better to have **broad top-level categories**, or
  - to present several levels of menu on one screen or web page.
- **Miller's magic number** of **7 ± 2** for working memory capacity is often **misused** in this context.
  - guidelines- **menu breadth**, **should be around seven.**
  - However, Miller's result applies **only to working memory**, **not visual search.**
  - **optimal breadth can be quite large**, perhaps **60 or more items for a web index page if the items are organized**

THINK ABOUT DIALOGUE

what does it mean in UI design?

Minister: do you *name* take this woman …
Man: I do
Minister: do you *name* take this man …
Woman: I do
Minister: I now pronounce you man and wife

THINK ABOUT DIALOGUE

Minister: do you *name* take this woman …

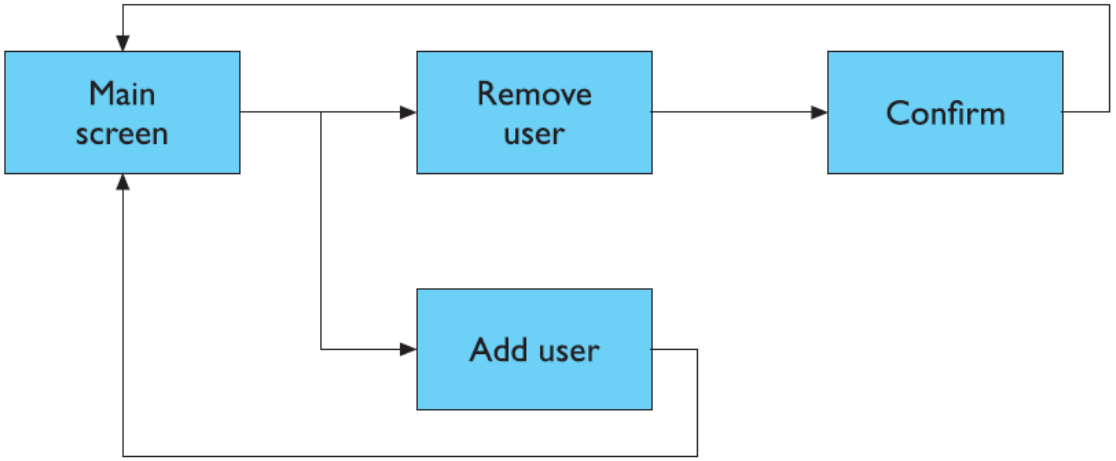
- marriage service
  - general flow, generic – blanks for names
  - pattern of interaction between people
- computer dialogue
  - pattern of interaction between users and system
  - but details differ each time

NETWORK DIAGRAMS

- SCENARIOS have one path through system
- Full system description
  - Different paths of system is needed
  - One such way of solution:
    - Network diagrams
      - Principal states or screens linked together with arrows

NETWORK DIAGRAMS

- Be **more task oriented** then hierarchy
- network diagram illustrates **main screens** for **adding** or **deleting** a user from the messaging system.
  - show **different paths** through system
  - Show **what leads to what**
  - Show **what happens** when
  - Including **branches and loops**



WIDER STILL

- Devices and applications still have several implications
  - **Style issues**
    - platform standards, consistency
  - **Functional issues**
    - File interaction,cut and paste
  - **Navigation issues**
    - embedded applications
    - links to other apps

SCREEN DESIGN AND LAYOUT

BASIC PRINCIPLES

- **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

- **visual tools to read** and **interact** with a **screen** or **device**.
  - Grouping and structure
  - Order of groups and items
  - Decoration
  - Alignment
  - White space

GROUPING AND STRUCTURE

- logically together ⇒ physically together
- Example : **Order Screen**

Order:
- Administrative information
  - Billing details
  - Delivery details
- Order information
  - Order line 1
  - Order line 2
  - ...

Billing details:
Name:
Address: ...
Credit card no:

Delivery details:
Name:
Address: ...
Delivery time:

Order details:

item	quantity	cost/item	cost
size 10 screws (boxes)	7	3.71	25.97
... ...	...	...	...

ORDER OF GROUPS AND ITEMS

- what is natural order
- should match screen order!
  - use boxes, space etc.
  - set up tabbing

Billing details:
Name:
Address: ...
Credit card no:

Delivery details:
Name:
Address: ...
Delivery time:

Order details:

item	quantity	cost/item	cost
size 10 screws (boxes)	7	3.71	25.97
... ...	...	...	...

DECORATION

- use **boxes** to group logical items
- Use **line** to separate the groupings
- use **fonts** for emphasis, **headings**
- **text** or **background colors**
- but not too many
- Example: **Microwave Control Panel**

ALIGNMENT

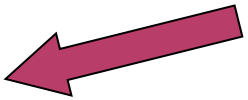
- lists of **text items** should normally be **aligned to the left.**
- **Numbers** should normally be **aligned to the right** (for integers) or at the decimal point.

ALIGNMENT - TEXT

- you read from **left to right** (English and European)

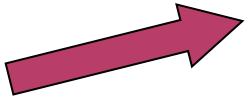
  $\Rightarrow$ align **left hand side**

Willy Wonka and the Chocolate Factory
Winston Churchill - A Biography
Wizard of Oz
Xena - Warrior Princess



boring but readable!

fine for special effects but hard to scan



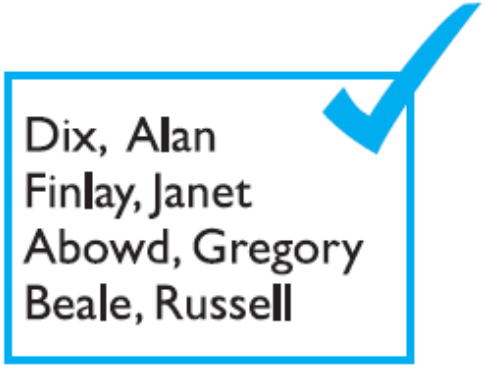
Willy Wonka and the Chocolate Factory
Winston Churchill - A Biography
Wizard of Oz
Xena - Warrior Princess

ALIGNMENT - NAMES

- Usually scanning for surnames
  ⇒ make it easy!

(i)

(ii)



(iii)

ALIGNMENT - NUMBERS

think purpose!

which is biggest?

532.56
179.3
256.317
15
73.948
1035
3.142
497.6256

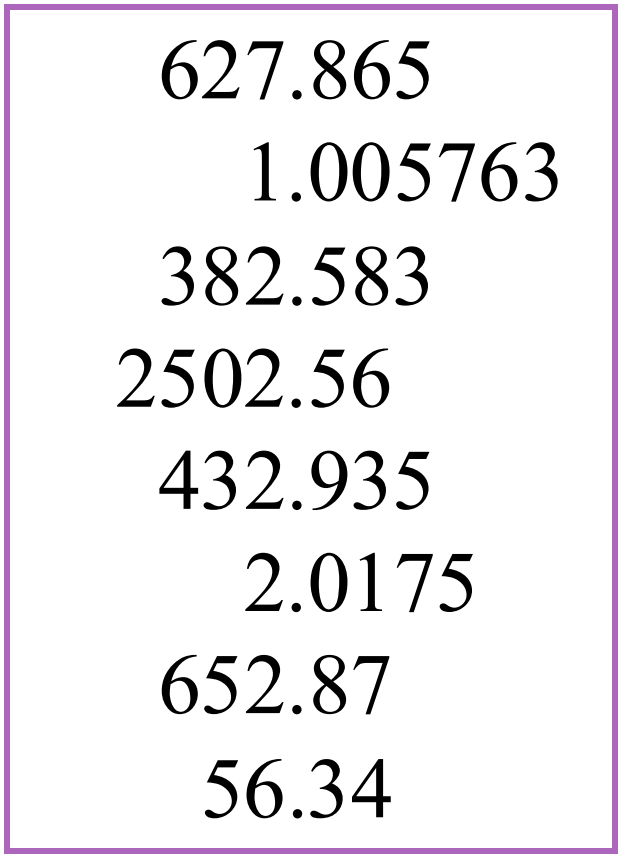
ALIGNMENT - NUMBERS

visually:

long number = big number

align decimal points

or right align integers



ALIGNMENT ON MULTIPLE COLUMNS

- scanning across gaps hard:

(often hard to avoid with large data base fields)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

ALIGNMENT ON MULTIPLE COLUMNS

- use leaders

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

ALIGNMENT ON MULTIPLE COLUMNS - 3

- or greying (vertical too)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

ALIGNMENT ON MULTIPLE COLUMNS - 4

- or even ‘bad’ alignment

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

MULTIPLE COLUMNS - SUMMARY

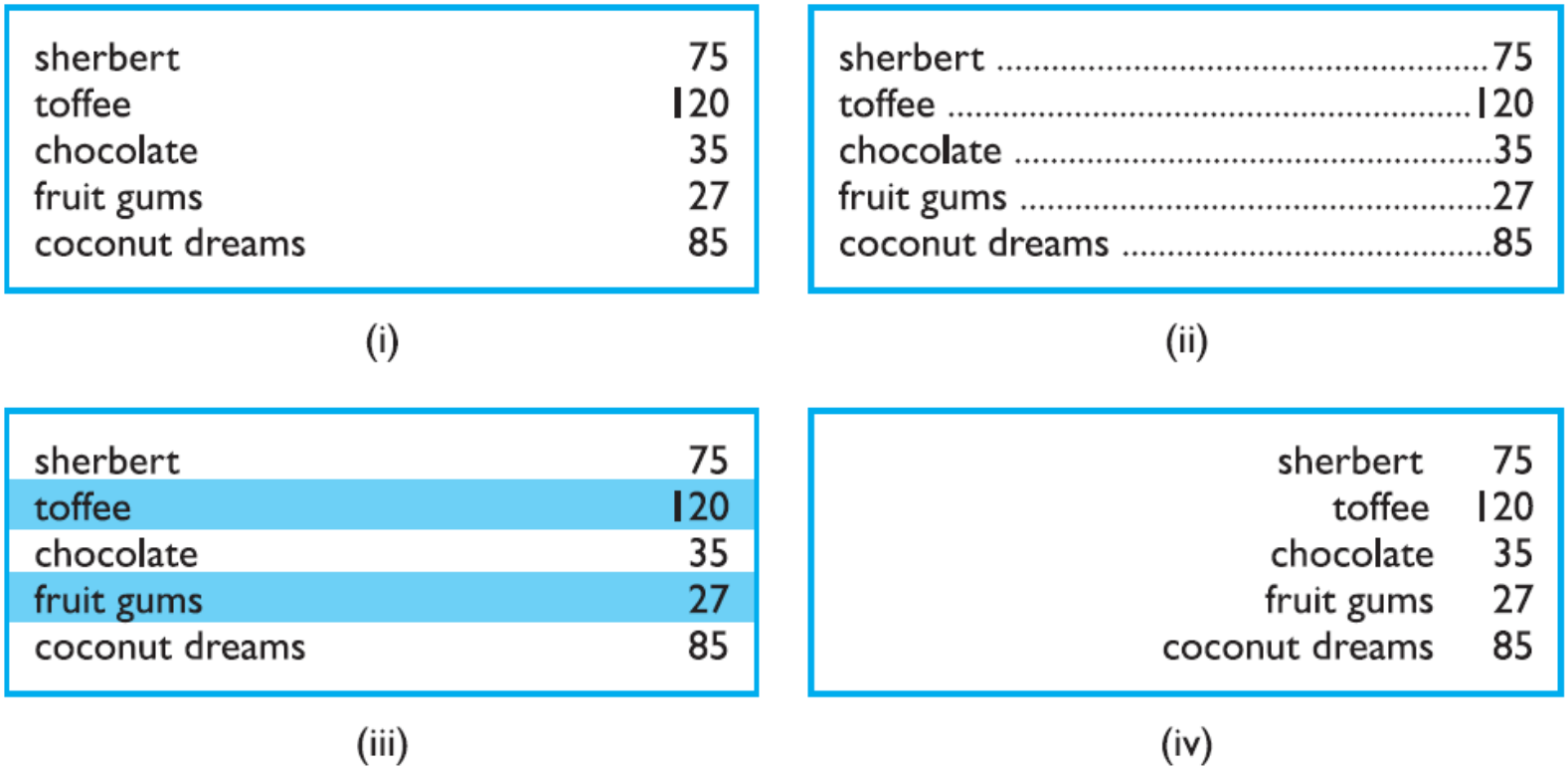
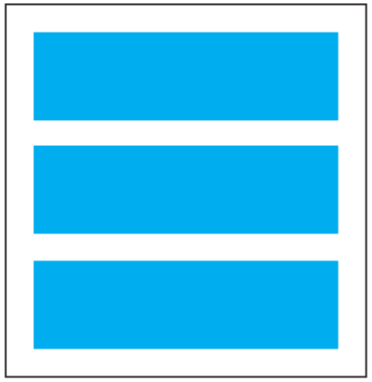


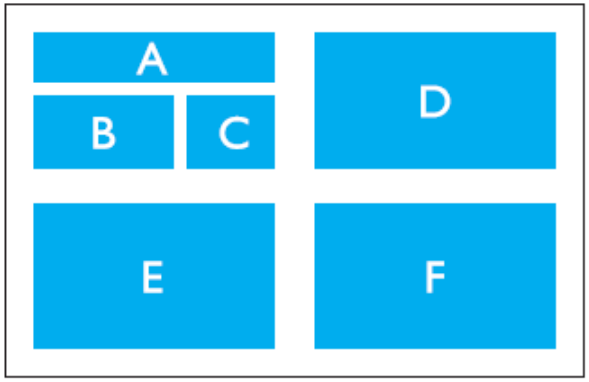
Figure 5.11 Managing multiple columns

WHITE SPACE

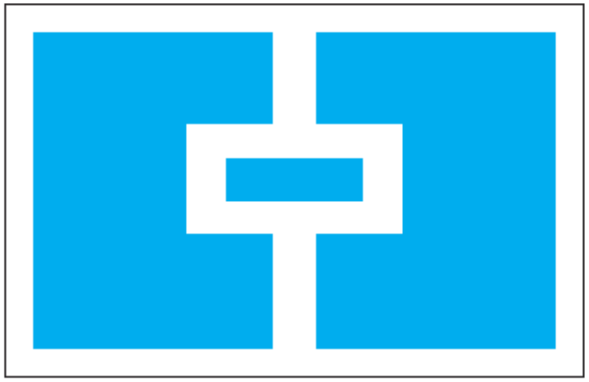
- In typography **the space between the letters** is called the **counter**.
- Space can be used in several ways.



(i) Space to separate



(ii) Space to structure



(iii) Space to highlight

COMPARISON WITH MICROWAVE CONTROL PANEL

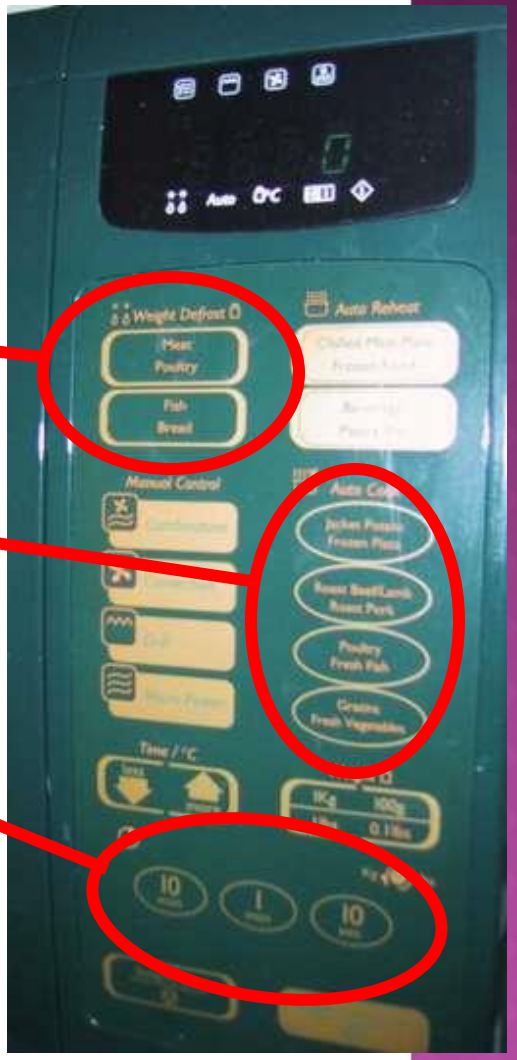
PHYSICAL CONTROLS

- grouping of items

  defrost settings

  type of food

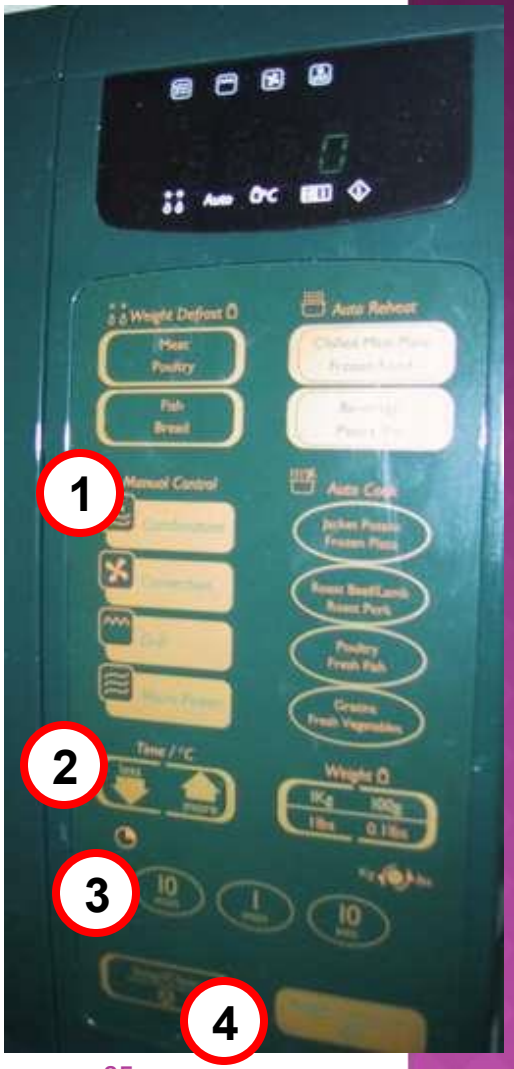
  time to cook



PHYSICAL CONTROLS

- grouping of items
- order of items

1) type of heating
2) temperature
3) time to cook
4) start

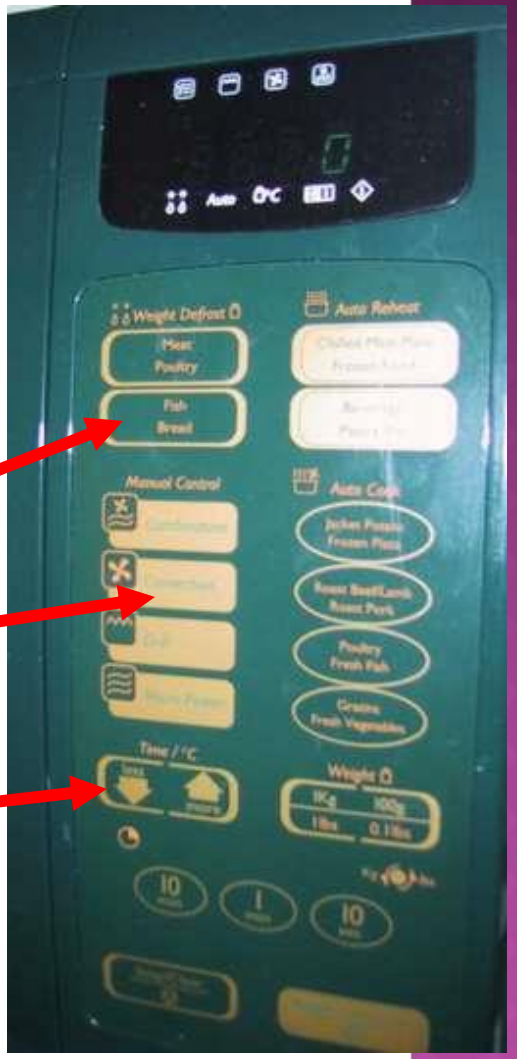


PHYSICAL CONTROLS

- grouping of items
- order of items
- decoration

different colours for different functions

lines around related buttons (temp up/down)

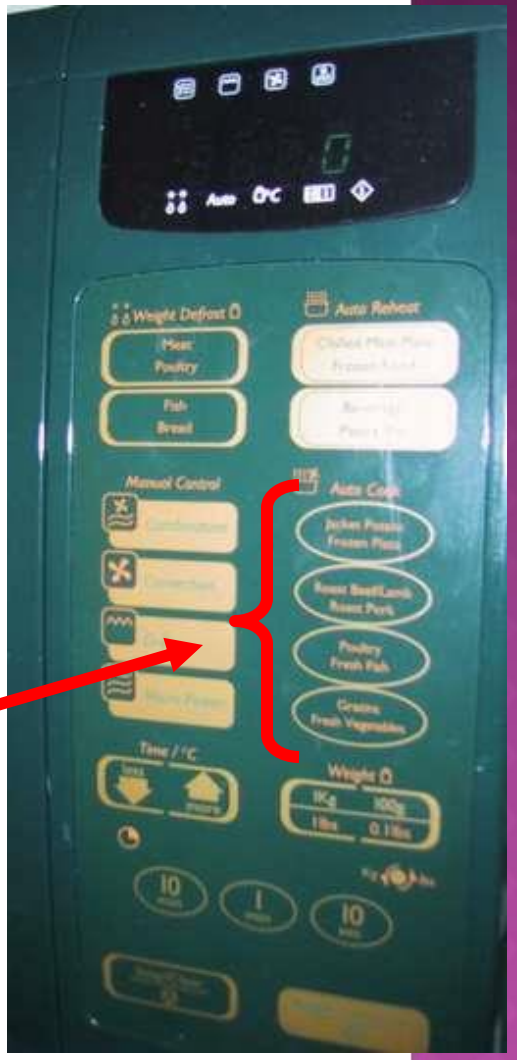


PHYSICAL CONTROLS

- grouping of items
- order of items
- decoration
- alignment

centred text in buttons

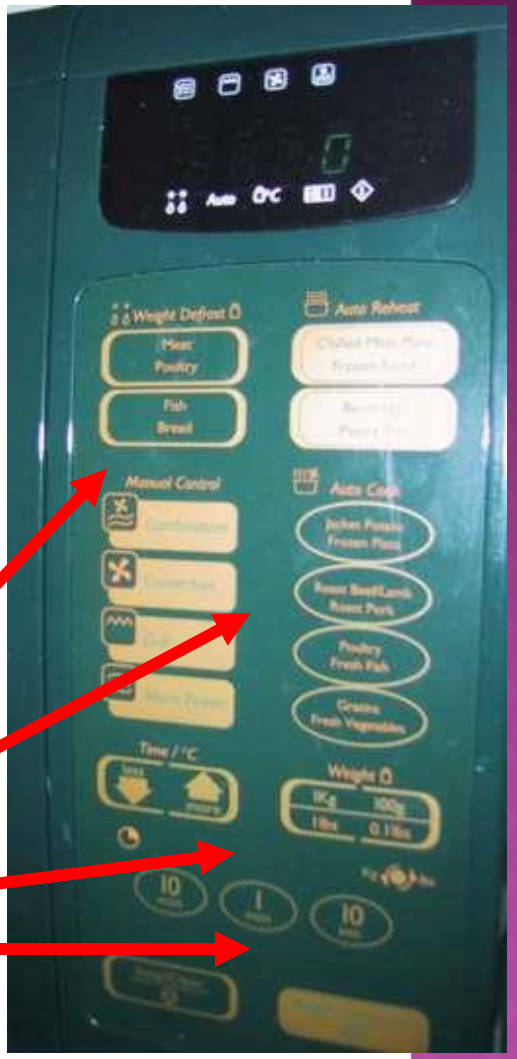
easy to scan



PHYSICAL CONTROLS

- grouping of items
- order of items
- decoration
- alignment
- white space

gaps to aid grouping



USER ACTION AND CONTROL

- Entering Information
- Knowing what to do
- Affordances

ENTERING INFORMATION

- forms, dialogue boxes
  - presentation + data input
  - similar layout issues
  - alignment - N.B. different label lengths

Name: Alan Dix
Address: Lancaster

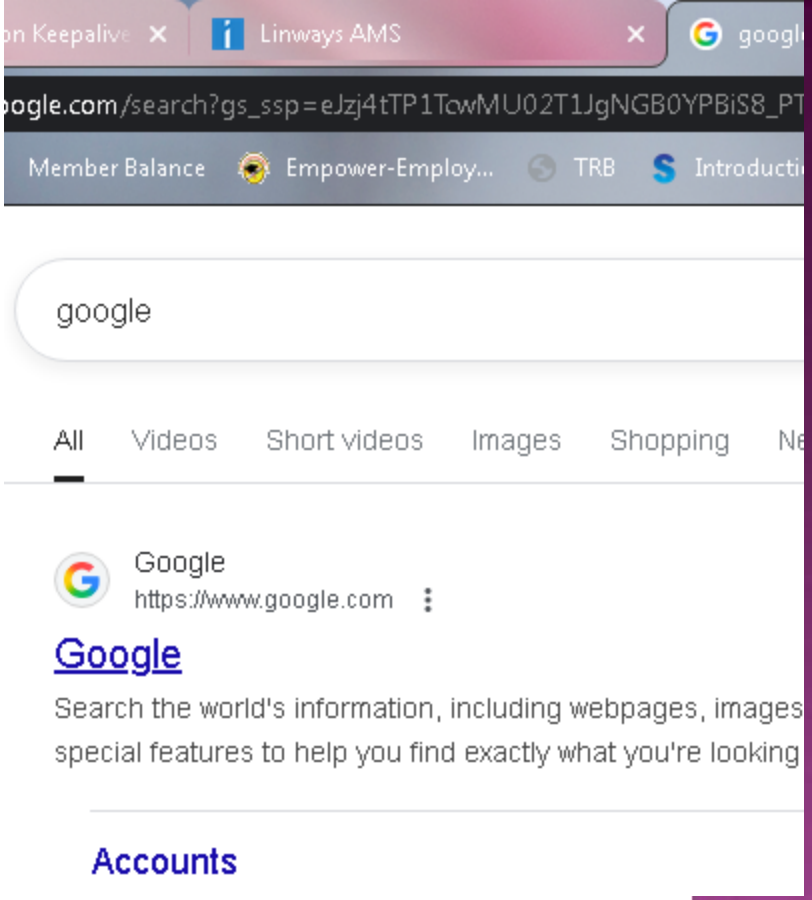
Name: Alan Dix
Address: Lancaster

- logical layout
  - use task analysis
  - groupings
  - natural order for entering information
    - **top-bottom**, **left-right** (depending on culture)
    - **set tab order for keyboard entry**

N.B. see extra slides for widget choice

KNOWING WHAT TO DO

- what is **active**
- what is **passive**
  - where to click
  - Where to type
- **consistent style helps**
  - e.g. web underlined links
- **labels** and **icons**
  - standards for common actions
  - language – bold = current state or action



AFFORDANCES

mug handle

- **psychological term**
- for physical objects
  - **shape and size suggest actions**
    - pick up, twist, throw
  - also cultural – buttons 'afford' pushing
- for screen objects
  - **button-like object 'affords' mouse click**
  - physical-like objects suggest use
- culture of computer use
  - **icons 'afford' clicking**
  - or even double clicking

'affords' grasping

APPROPRIATE APPEARANCE

- presenting information
- aesthetics and utility
- colour and 3D
- localisation & internationalisation

PRESENTING INFORMATION

- purpose matters
  - sort order (which column, numeric alphabetic)
  - text vs. diagram
  - scatter graph vs. histogram

- but add interactivity
  - softens design choices
    - e.g. re-ordering columns
    - ‘dancing histograms’ (chap 21)

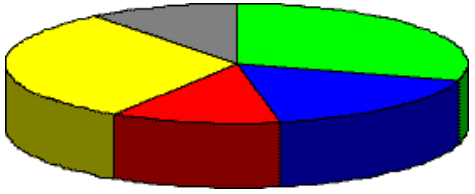
AESTHETICS AND UTILITY

- aesthetically pleasing designs
  - increase user satisfaction and improve productivity
- beauty and utility may conflict
  - mixed up visual styles ⇒ easy to distinguish
  - clean design – little differentiation ⇒ confusing
  - backgrounds behind text
    … good to look at, but hard to read
- but can work together
  - e.g. the design of the counter
  - in consumer products – key differentiator (e.g. iMac)

COLOUR AND 3D

- both often used very badly!
- colour
  - older monitors limited palette
  - use sparingly to reinforce other information
- 3D effects
  - good for physical information and some graphs
  - but if over used

    e.g. text in perspective!! 3D pie charts



BAD USE OF COLOUR

- over use - without very good reason (e.g. kids' site)
- colour blindness
- poor use of contrast
- do adjust your set!
  - adjust your monitor to greys only
  - can you still read your screen?

ACROSS COUNTRIES AND CULTURES

- localisation & internationalisation
  - changing interfaces for particular cultures/languages
- globalisation
  - try to choose symbols etc. that work everywhere
- simply change language?
  - use ‘resource’ database instead of literal text
    … but changes sizes, left-right order etc.
- deeper issues
  - cultural assumptions and values
  - meanings of symbols

    e.g tick and cross … +ve and -ve in some cultures
    … but … mean the same thing (mark this) in others

ITERATION AND PROTOTYPING

ITERATION AND PROTOTYPING

- Any of these prototypes, whether **paper-based or running software**, can then be **evaluated**
  - to see whether they are **acceptable** and where there is **room for improvement**.
- This sort of evaluation, intended to improve designs, is called ***formative* evaluation**.
- This is in contrast to ***summative* evaluation**, which is performed **at the end** to verify whether the product is good enough.
- So iteration and prototyping are the universally accepted 'best practice' approach for interaction design.

PROTOTYPING

- you never get it right first time
- if at first you don't succeed …

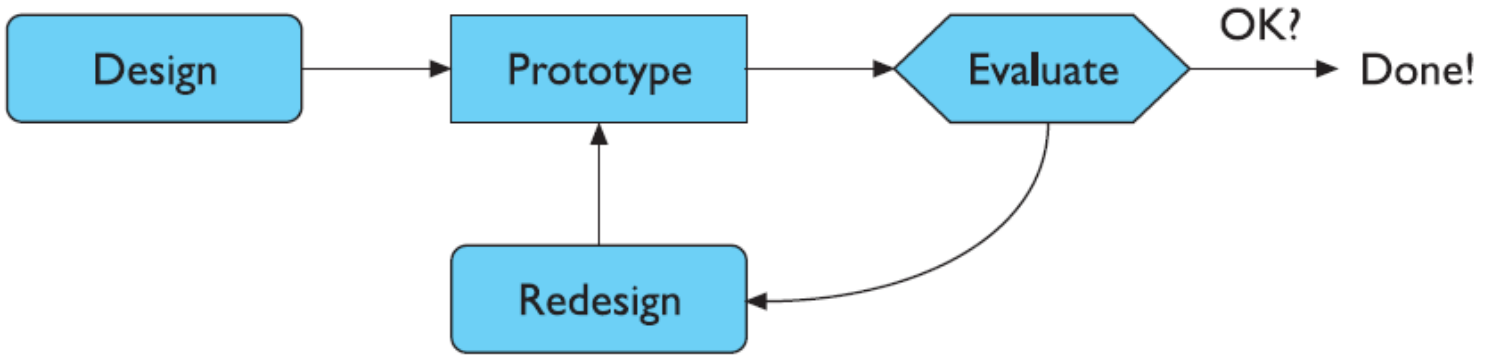


Figure 5.14 Role of prototyping

HILL-CLIMBING APPROACH

- Prototyping is an example of what is known as a ***hill-climbing*** approach.
- However, hill climbing **doesn't always work**.

- good designer might guess a **good initial design** based on **experience** and **Judgment**.
- Have **several initial design ideas** and **drop them one by one** as they are developed further.

PITFALLS OF PROTOTYPING

- moving little by little … but to where
- **Malverns** or the **Matterhorn**?

Figure 5.15 Moving little by little . . . but to where?

1. need a good start point
2. need to understand what is wrong

HCI in the software process

HCI IN THE SOFTWARE PROCESS

- Software engineering and the design process for interactive systems (**Software Life Cycle**)
- **Usability engineering -** explicit usability requirements
- **Iterative design** and **prototyping**
- **Design rationale**

SOFTWARE ENGINEERING AND SOFTWARE LIFE CYCLE

- ***software engineering***
  - addresses **the management** and **technical issues of the development of software systems**.
- ***software life cycle***
  - which describes the **activities** that take place from the **initial concept formation** for a **software system** up until its eventual **phasing** out and replacement.

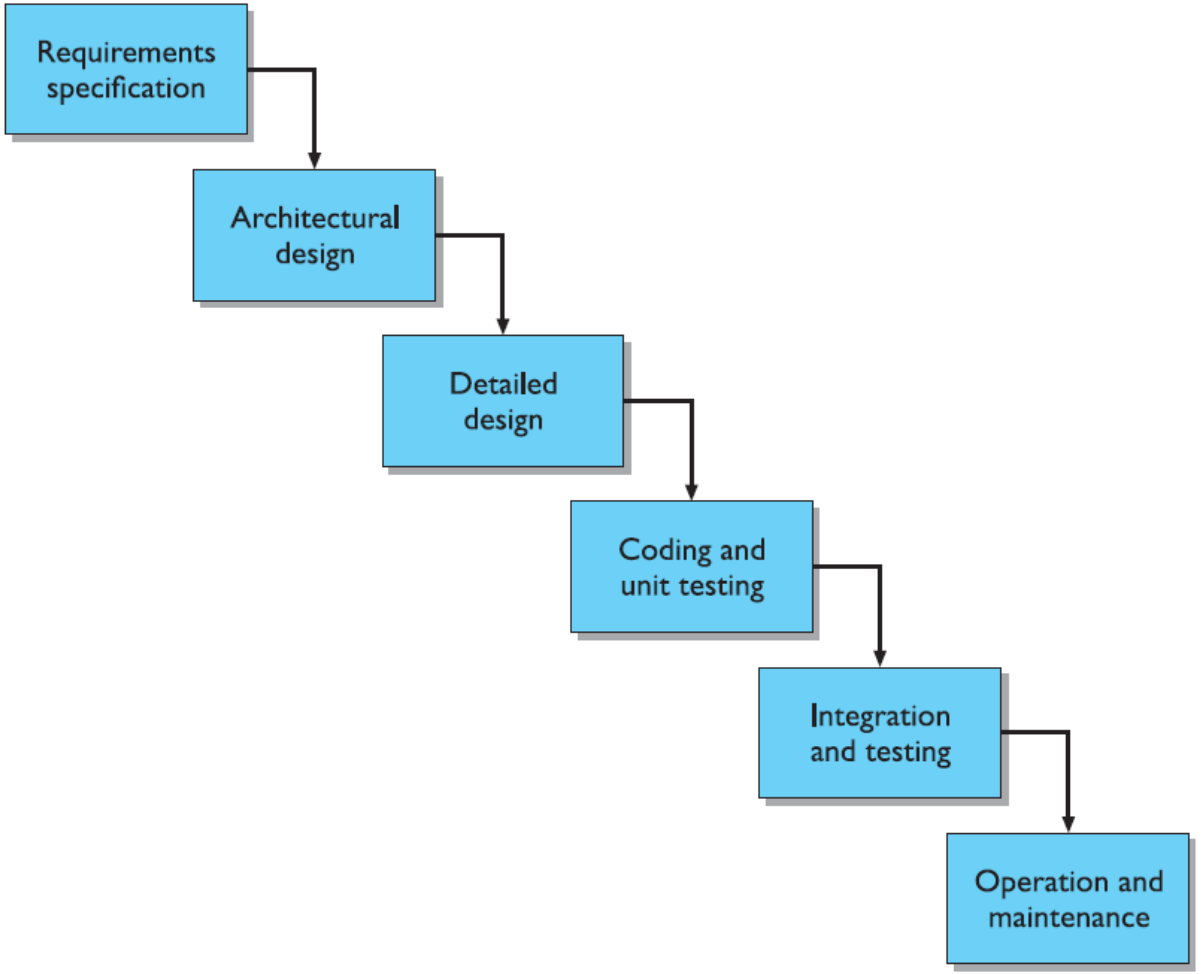
THE SOFTWARE LIFECYCLE

THE SOFTWARE LIFECYCLE

- Software engineering is the **discipline** for **understanding the software design process, or life cycle.**

- Designing for **usability occurs at all stages of the life cycle**, not as a single isolated activity

THE WATERFALL MODEL



REQUIREMENTS SPECIFICATION

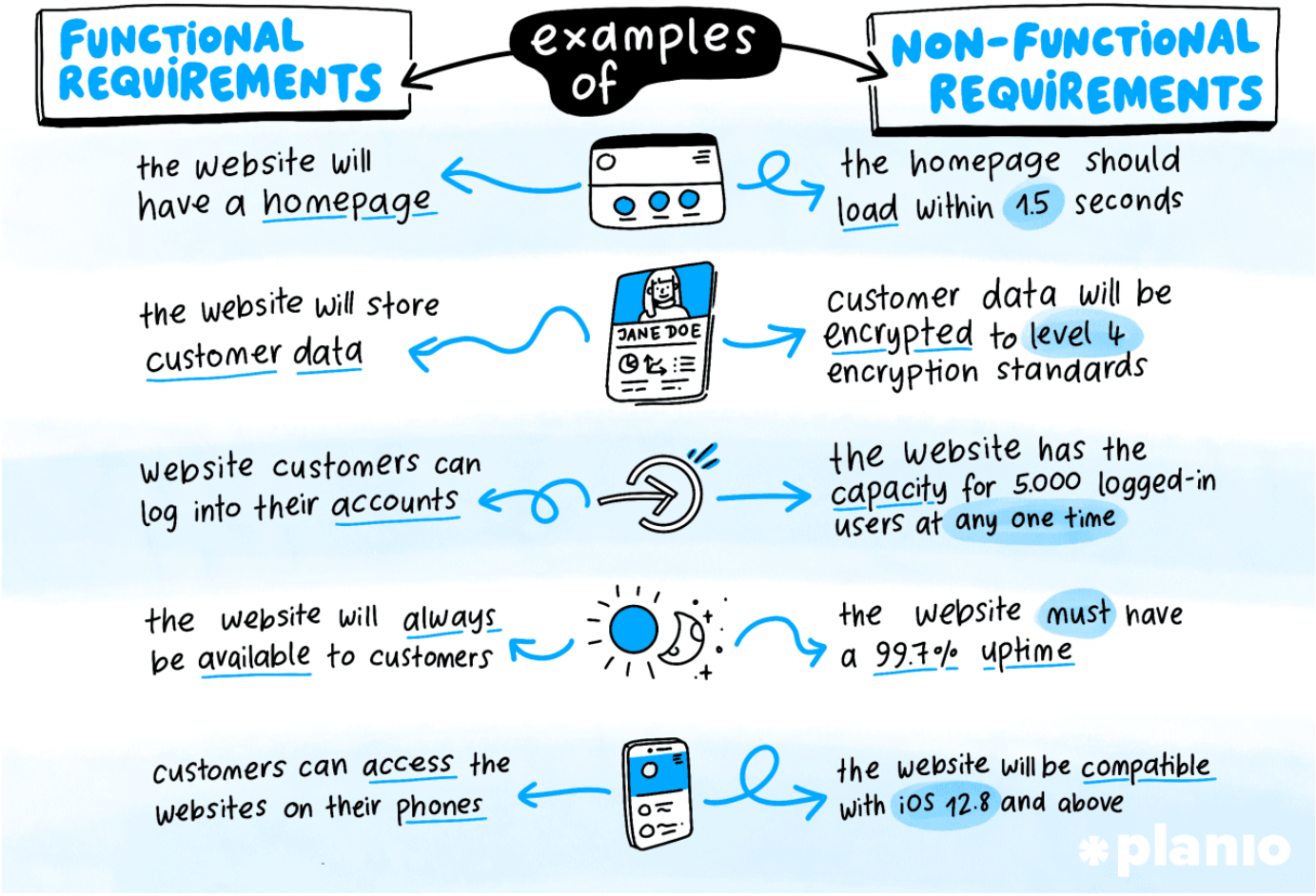
- Designer and customer try to capture **what the system is expected to provide**.
- **Eliciting information** from the customer about the
  - **work environment**, or domain, in which the final product will function.
- Requirements are from the **customer's perspective.**
- Requirements are usually initially expressed in the **native language of the customer**.
- **Task analysis techniques,** which are used to express work **domain requirements** in a form that is both expressive and precise.

ARCHITECTURAL DESIGN

- high-level **decomposition** of the **system** into **components.**
- An architectural design performs this **decomposition.**
- Many **structured techniques** are used. Such as
  - **CORE** (Controlled Requirements Expression)
  - **MASCOT** (Modular Approach to Software Construction Operation and Test)
  - **HOOD** (Hierarchic Object-Oriented Design)
- ***functional requirements*** of the system – **the services the system must provide in the work domain** – but do not provide an immediate way to capture other
- ***non-functional requirements*** – features of the system that are **not directly related to the actual services provided** but relate to the manner in which those services must be provided.
- Some classic examples of **non-functional requirements** are the
  - **Efficiency,reliability, timing** and **safety** features of the system.

Functional Requirements	Non Functional Requirements
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies “What should the software system do?”	It places constraints on “How should the software system fulfill the functional requirements?”
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API testing, etc are done.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done.
Usually easy to define.	Usually more difficult to define.
Example1) Authentication of user whenever he/she logs into the system. 2) System shutdown in case of a cyber attack. 3) A Verification email is sent to user whenever he/she registers for the first time on some software system.	Example1) Emails should be sent with a latency of no greater than 12 hours from such an activity. 2) The processing of each request should be done within 10 seconds 3) The site should load in 3 seconds when the number of simultaneous users are > 10000

EXAMPLE - FUNCTIONAL VS NON FUNCTIONAL REQUIREMENTS



DETAILED DESIGN

- **Designer** must provide a **sufficient detailed description** so that they may be implemented in some programming language.
- The detailed design is a ***refinement*** of the **component description** provided by the architectural design.

CODING AND UNIT TESTING

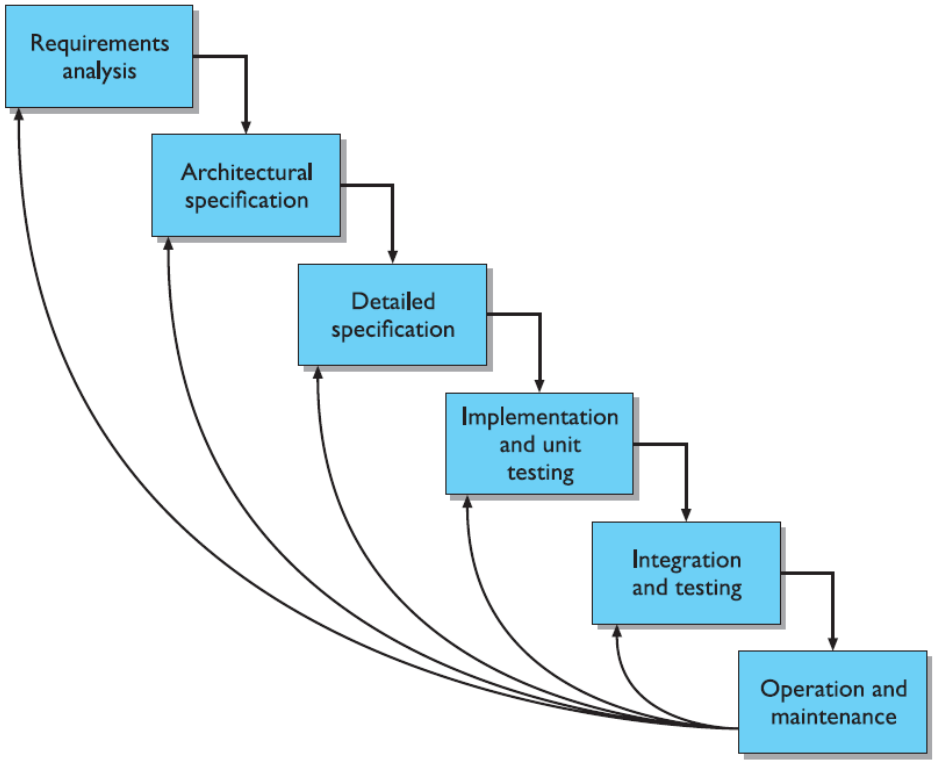
- A **component of the system** should be in such a form that it is possible to implement it in some **executable programming language.**
- After **coding**, the **component can be tested** to **verify** that it performs correctly.
- **automatic generation of tests** from output of earlier activities which can be performed on a piece of code **to verify that it behaves correctly.**

INTEGRATION AND TESTING

- Components must be **integrated** as described in the **architectural design**.
- Further **testing** is done **to ensure correct behavior** and **acceptable** use of any shared resources.
- It is also possible at this time to perform some **acceptance testing** with the **customers** to ensure that the **system meets their requirements**.
- It is only after acceptance of the integrated system that the **product is finally released to the customer.**
- Necessary to **certify** the final system according to requirements imposed by some outside authority such as **ISO**, etc.
- **ISO 9241** provide impetus for designers to take seriously the HCI implications of their design.

MAINTENANCE

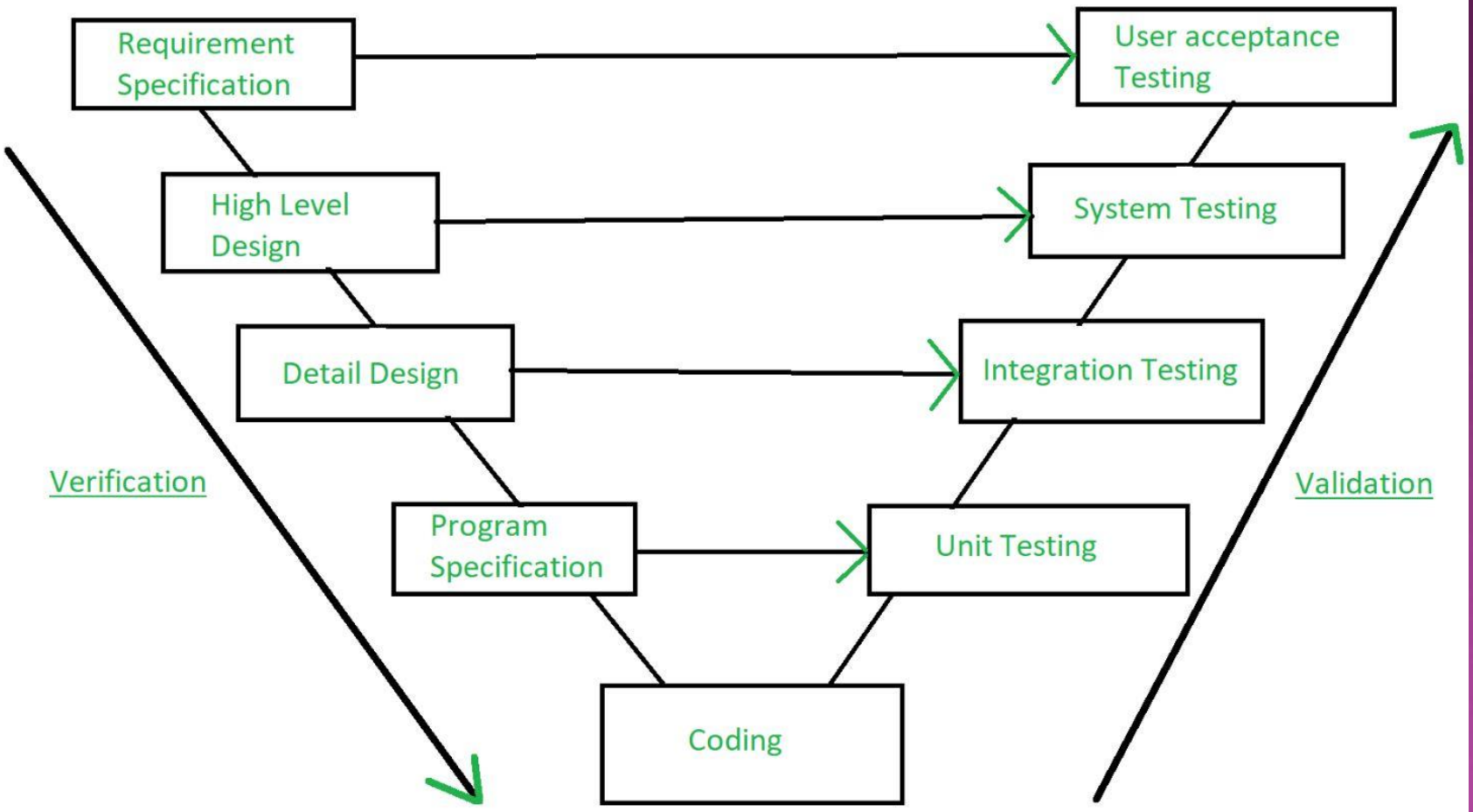
- Maintenance involves the **correction of errors in the system.**
- The **revision of the system** services .
- Maintenance provides **feedback** to all of the other activities in the life cycle.
- The **majority of the lifetime** of a product is spent in the **maintenance activity**.



VALIDATION AND VERIFICATION

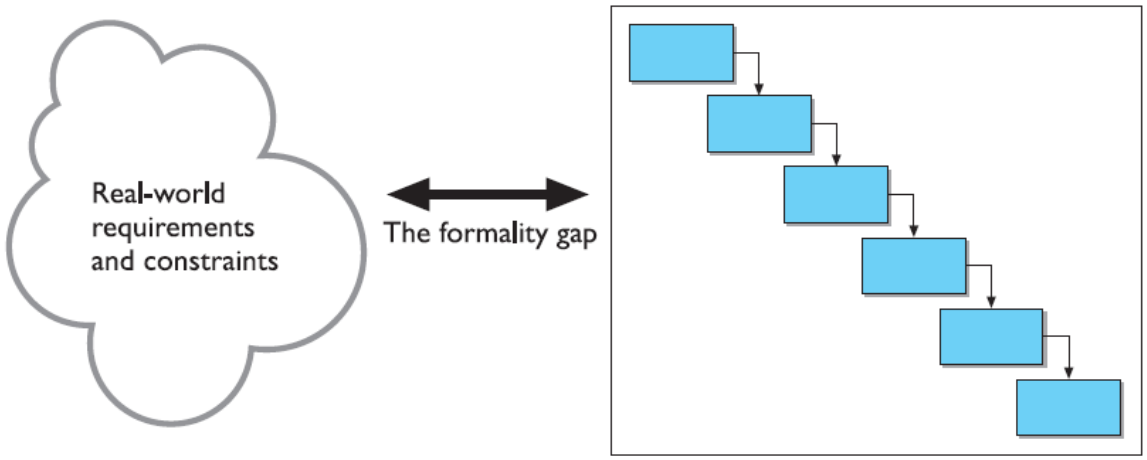
- Throughout the life cycle, **the design must be checked**
  - to ensure that it **both satisfies the high-level requirements** agreed with the **customer** and
  - It is also **complete and internally consistent**.
- These checks are referred to as
  - ***validation*** and
  - ***verification***, respectively.
- **Boehm** [36a] provides a useful distinction between the two, characterizing
  - **validation** as designing '**the right thing**' and
  - **verification** as designing '**the thing right**'.
- **Verification is followed by Validation**.

VALIDATION AND VERIFICATION



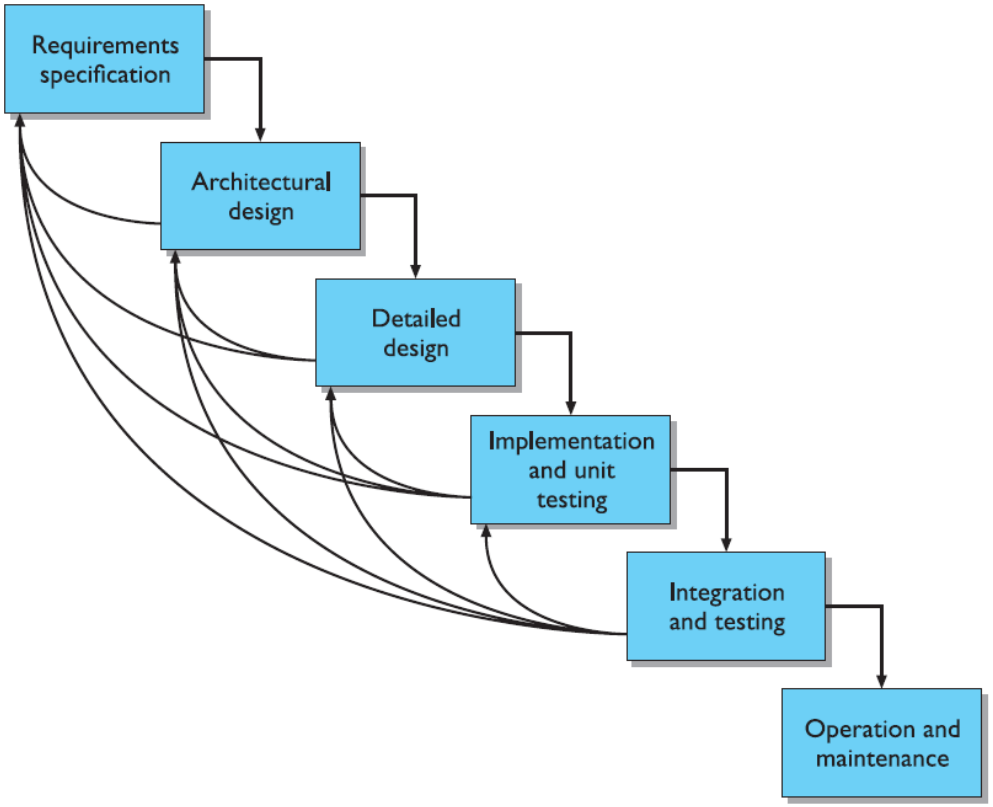
FORMALITY GAP

- Validation proofs are much trickier.
- the origin of **customer requirements arises in the inherent ambiguity of the real world** and not the mathematical world.
- there will always be a leap from the **informal situations** of the real world to any **formal** and structured development process
- We refer to this inevitable disparity as the ***formality gap*** .
- The **formality gap** means that **validation will always rely to some extent on subjective means of proof.**
- We can increase our **confidence** in the subjective proof by **effective use of real-world experts in performing certain validation chores.**
- The **formality gap between the real world and structured design.**

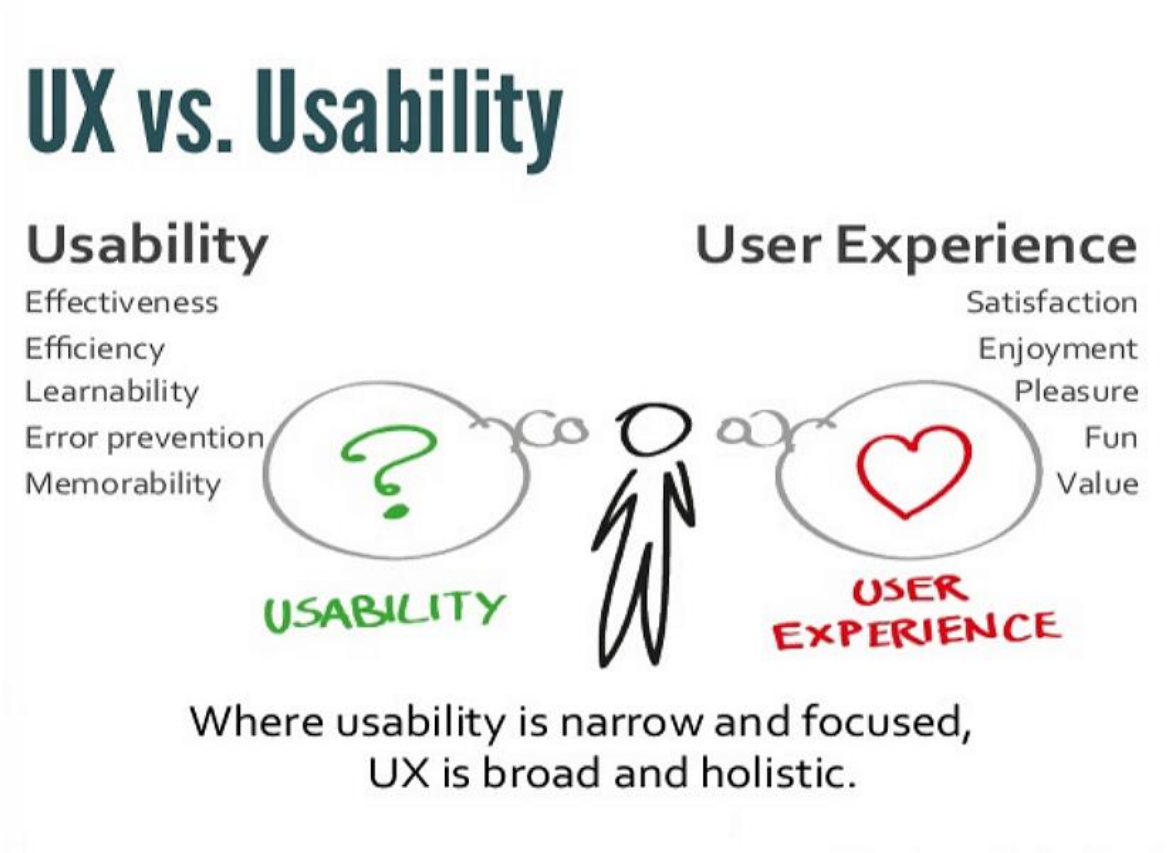


ITERATIVE RELATIONSHIP IN SDLC.

- **cannot assume a linear sequence of activities** as in the waterfall model.
- Finally, The life cycle for **interactive systems** could be shown in below.



USABILITY ENGINEERING



USABILITY

- **Usability** can be described as
  - the **capacity of a system** to provide a condition for its users to perform the tasks **safely, effectively**, and **efficiently** while enjoying the experience.
- A **usability study** may be conducted as a
  - **primary job** function by a ***usability analyst***.
  - **secondary job** function by **designers, technical writers, marketing personnel**, and others.
- "**usability**" also refers to **methods for improving ease-of-use** during the **design process**.

USABILITY ENGINEERING

- **Usability engineering** is a field that is concerned generally with **human-computer interaction** and specifically with devising **human-computer interfaces** that have **high usability** or **user friendliness**.
- It provides **structured methods** for achieving **efficiency** and **elegance** in interface design.
- One approach to **user-centered design** has been the introduction of explicit ***usability engineering* goals** into the **design process**.
- Usability goals must address the **three usability components**, i.e.
  - **Effectiveness**
  - **Efficiency**
  - **Satisfaction**

USABILITY ENGINEERING

- The ultimate test of usability based on measurement of **user experience.**
- The important features of usability engineering is the inclusion of a **usability specification.**
- Various **attributes** of the system are suggested as gauges **for testing the usability**.
- For each attribute, **six items are defined** to form the usability specification of that attribute.
  - **measuring concept**
  - **measuring method**
  - **now level**
  - **worst case**
  - **planned level**
  - **best case**

USABILITY SPECIFICATION ATTRIBUTE ITEMS

- ***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.
- ***now level*** indicates the value for the measurement with the existing system, whether it is computer based or not.
- The ***worst case*** value is the lowest acceptable measurement for the task, providing a clear distinction between what will be acceptable and what will be unacceptable in the final product.
- 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 and technology.

EXAMPLE - VCR

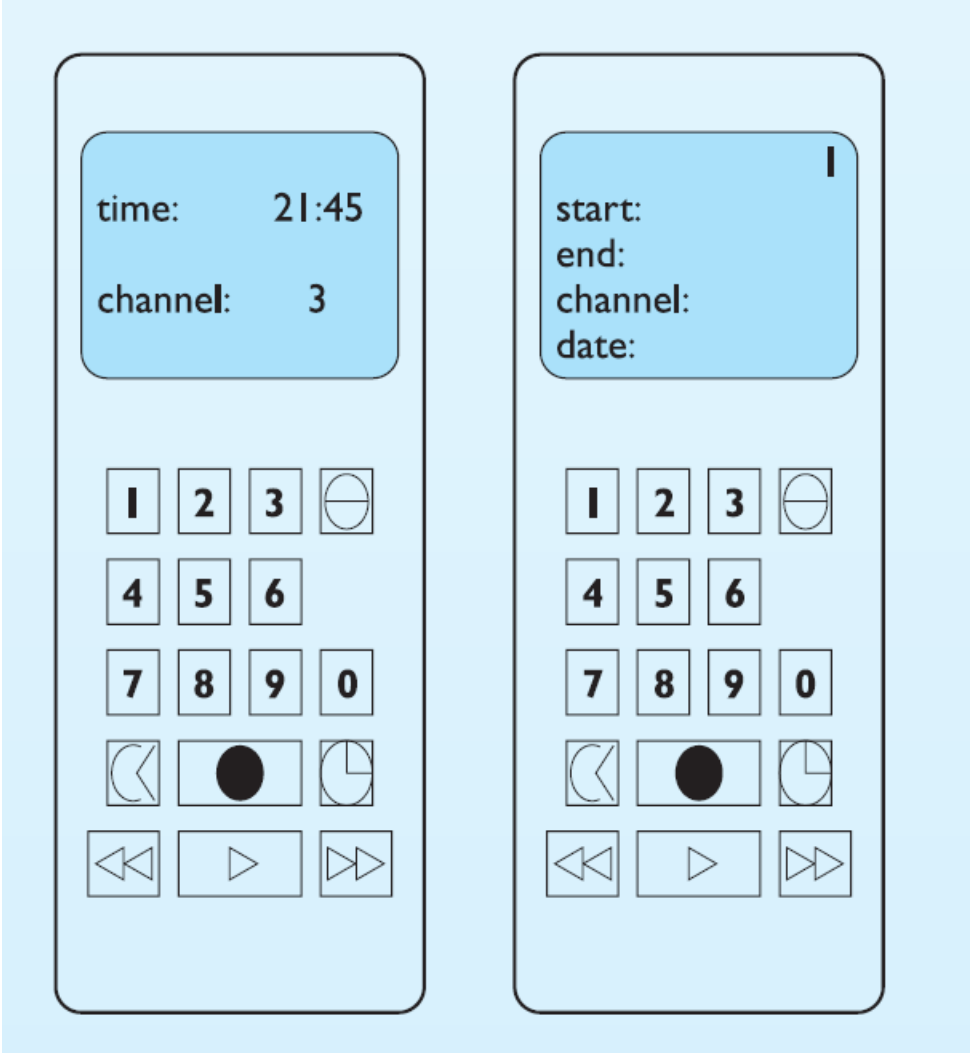
- Usability specification for the design of a **control panel** for a **video cassette recorder (VCR)**

Example: programming a video recorder by remote control

We can illustrate how the walkthrough method works using a simple example. Imagine we are designing a remote control for a video recorder (VCR) and are interested in the task of programming the VCR to do timed recordings. Our initial design is shown in Figure 9.1. The picture on the left illustrates the handset in normal use, the picture on the right after the timed record button has been pressed. The VCR allows the user to program up to three timed recordings in different 'streams'. The next available stream number is automatically assigned. We want to know whether our design supports the user's task. We begin by identifying a representative task.

> Program the video to time-record a program starting at 18.00 and finishing at 19.15 on channel 4 on 24 February 2005.

AN INITIAL REMOTE CONTROL DESIGN



We will assume that the user is familiar with VCRs but not with this particular design.

The next step in the walkthrough is to identify the action sequence for this task. We specify this in terms of the user's action (UA) and the system's display or response (SD). The initial display is as the left-hand picture in Figure 9.1.

UA 1: Press the 'timed record' button
SD 1: Display moves to timer mode. Flashing cursor appears after 'start:'
UA 2: Press digits 1 8 0 0
SD 2: Each digit is displayed as typed and flashing cursor moves to next position
UA 3: Press the 'timed record' button
SD 3: Flashing cursor moves to 'end:'
UA 4: Press digits 1 9 1 5
SD 4: Each digit is displayed as typed and flashing cursor moves to next position
UA 5: Press the 'timed record' button
SD 5: Flashing cursor moves to 'channel:'
UA 6: Press digit 4
SD 6: Digit is displayed as typed and flashing cursor moves to next position
UA 7: Press the 'timed record' button
SD 7: Flashing cursor moves to 'date:'
UA 8: Press digits 2 4 0 2 0 5
SD 8: Each digit is displayed as typed and flashing cursor moves to next position
UA 9: Press the 'timed record' button
SD 9: Stream number in top right-hand corner of display flashes
UA 10: Press the 'transmit' button
SD 10: Details are transmitted to video player and display returns to normal mode

Having determined our action list we are in a position to proceed with the walkthrough. For each action (1–10) we must answer the four questions and tell a story about the usability of the system. Beginning with UA 1:

UA 1: Press the 'timed record' button

Question 1: Is the effect of the action the same as the user's goal at that point?

The timed record button initiates timer programming. It is reasonable to assume that a user familiar with VCRs would be trying to do this as his first goal.

Question 2: Will users see that the action is available?

The 'timed record' button is visible on the remote control.

Question 3: Once users have found the correct action, will they know it is the one they need?

It is not clear which button is the 'timed record' button. The icon of a clock (fourth button down on the right) is a possible candidate but this could be interpreted as a button to change the time. Other possible candidates might be the fourth button down on the left or the filled circle (associated with record). In fact, the icon of the clock is the correct choice but it is quite possible that the user would fail at this point. This identifies a potential usability problem.

Question 4: After the action is taken, will users understand the feedback they get?

Once the action is taken the display changes to the timed record mode and shows familiar headings (start, end, channel, date). It is reasonable to assume that the user would recognize these as indicating successful completion of the first action.

So we find we have a potential usability problem relating to the icon used on the 'timed record' button. We would now have to establish whether our target user group could correctly distinguish this icon from others on the remote.

The analysis proceeds in this fashion, with a walkthrough form completed for each action. We will leave the rest of the walkthrough for you to complete as an exercise. What other usability problems can you identify with this design?

RECOVERABILITY

- **Recoverability** refers to the ability to reach a desired goal after recognition of some error in previous interaction.
- The recovery procedure can be in either a **backward** or **forward** sense.
- In designing a **new VCR control panel**, the designer wants to take into account **how a user might recover from a mistake** he discovers while trying **to program the VCR to record some television program in his absence**.

USABILITY SPECIFICATION FOR A VCR

Attribute:	Backward recoverability
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:	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

MEASUREMENT CRITERIA AND POSSIBLE WAYS TO SET MEASUREMENT LEVELS FOR VCR

- list of **measurement criteria** which can be used to determine the **measuring method for a usability attribute** and the possible ways to set the
- Worst/best case (e.g., slowest or fastest time a user finishes a task)
- Planned target (e.g., users to finish)

1. Time to complete a task
2. Per cent of task completed
3. Per cent of task completed per unit time
4. Ratio of successes to failures
5. Time spent in errors
6. Per cent or number of errors
7. Per cent or number of competitors better than it
8. Number of commands used
9. Frequency of help and documentation use
10. Per cent of favorable/unfavorable user comments
11. Number of repetitions of failed commands
12. Number of runs of successes and of failures
13. Number of times interface misleads the user
14. Number of good and bad features recalled by users
15. Number of available commands not invoked
16. Number of regressive behaviors
17. Number of users preferring your system
18. Number of times users need to work around a problem
19. Number of times the user is disrupted from a work task
20. Number of times user loses control of the system
21. Number of times user expresses frustration or satisfaction

- Example :
- Time to complete a task
- Number of errors
- User satisfaction

USABILITY METRICS

- Measurements such as those promoted by usability engineering are also called ***usability metrics.***
- The **ISO standard 9241**, described earlier, also recommends the use of **usability specifications** as a means of requirements specification.
- Three categories of usability:
  - **effectiveness**, **efficiency** and **satisfaction**.

ISO USABILITY STANDARD 9241

- This Standard is covers the **Ergonomics for HCI**.
- Also adopts traditional **usability categories**:
  - **effectiveness**
    - can you achieve what you want to?
  - **efficiency**
    - can you do it without wasting effort?
  - **satisfaction**
    - do you enjoy the process?
- Example:For Google Search:

Effectiveness – Can user find correct info?

Efficiency – How fast results appear?

Satisfaction – Is interface easy and pleasant?

ISO USABILITY STANDARD 9241

- **Ex:Fund transfer**
- **effectiveness**
  - Ex:able to withdraw money
- **efficiency**
  - Few steps for withdrawal
- **satisfaction**
  - Ex:understandable interface

PROBLEMS WITH USABILITY ENGINEERING

- Three Problems:
- Difficult early measurement: Hard to define usability goals before design is ready.
- Too specific: Usability metrics often measure very small tasks, not overall experience.
- Specification ≠ Real Usability: satisfying a metric doesn't relies on product easy to use.

ITERATIVE DESIGN AND PROTOTYPING

ITERATIVE DESIGN AND PROTOTYPING

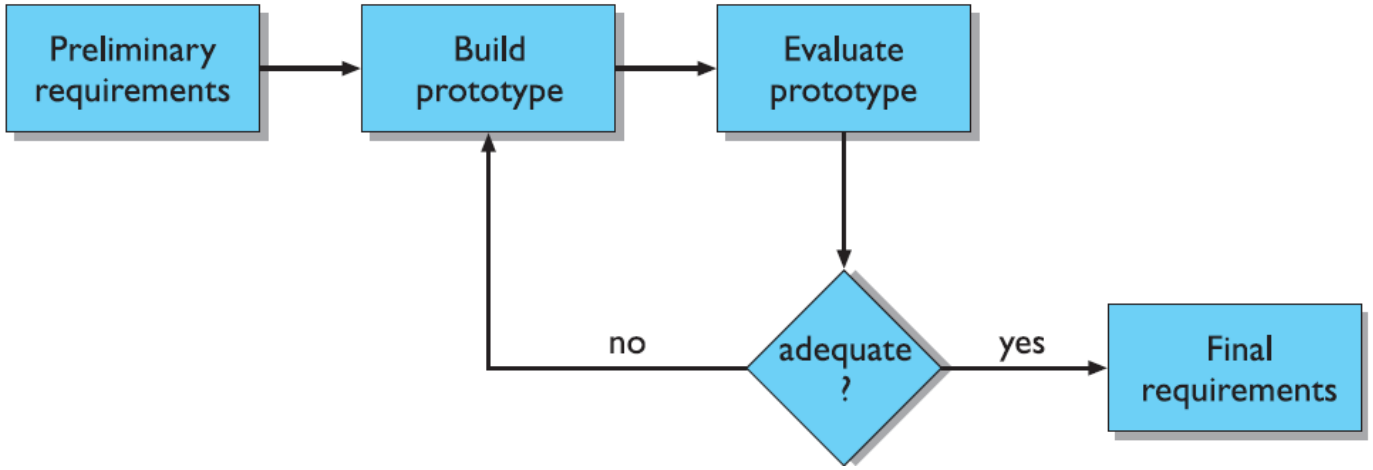
- Prototypes are **artifacts** that **simulate** or **animate** some **but not all features** of the intended system.
- Iterative design = repeatedly improving design based on testing feedback.
- Prototype = sample model of the system.

APPROACHES

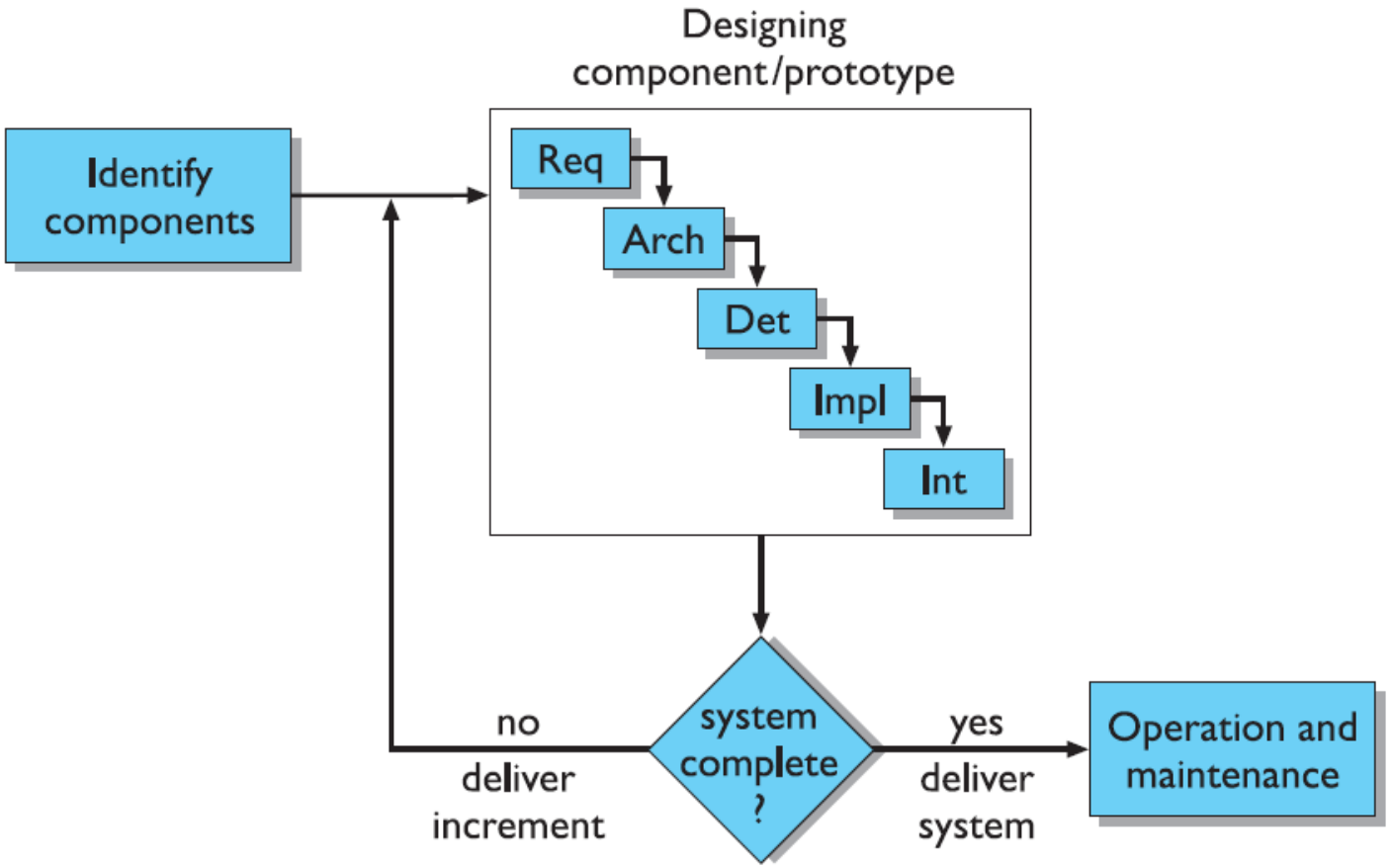
- There are **three main approaches** to prototyping:
  - **throw-away**
    - The prototype is built and tested. The design knowledge gained from this exercise is **used to build the final product**, **but the actual prototype is discarded.**
    - Temporary model → used for testing ideas → discarded.

- **Incremental**
- The final product is built as **separate components, one at a time**. There is one overall design for the final system, but it is **partitioned into independent and smaller components**. The final product is then released as a series of products, each subsequent release including one more component.
- **Evolutionary**
- Here the prototype is **not discarded** and serves as the **basis for the next iteration of design**. In this case, the actual system is seen as evolving from a very limited initial version to its final release.
- Improving the same prototype until final version.

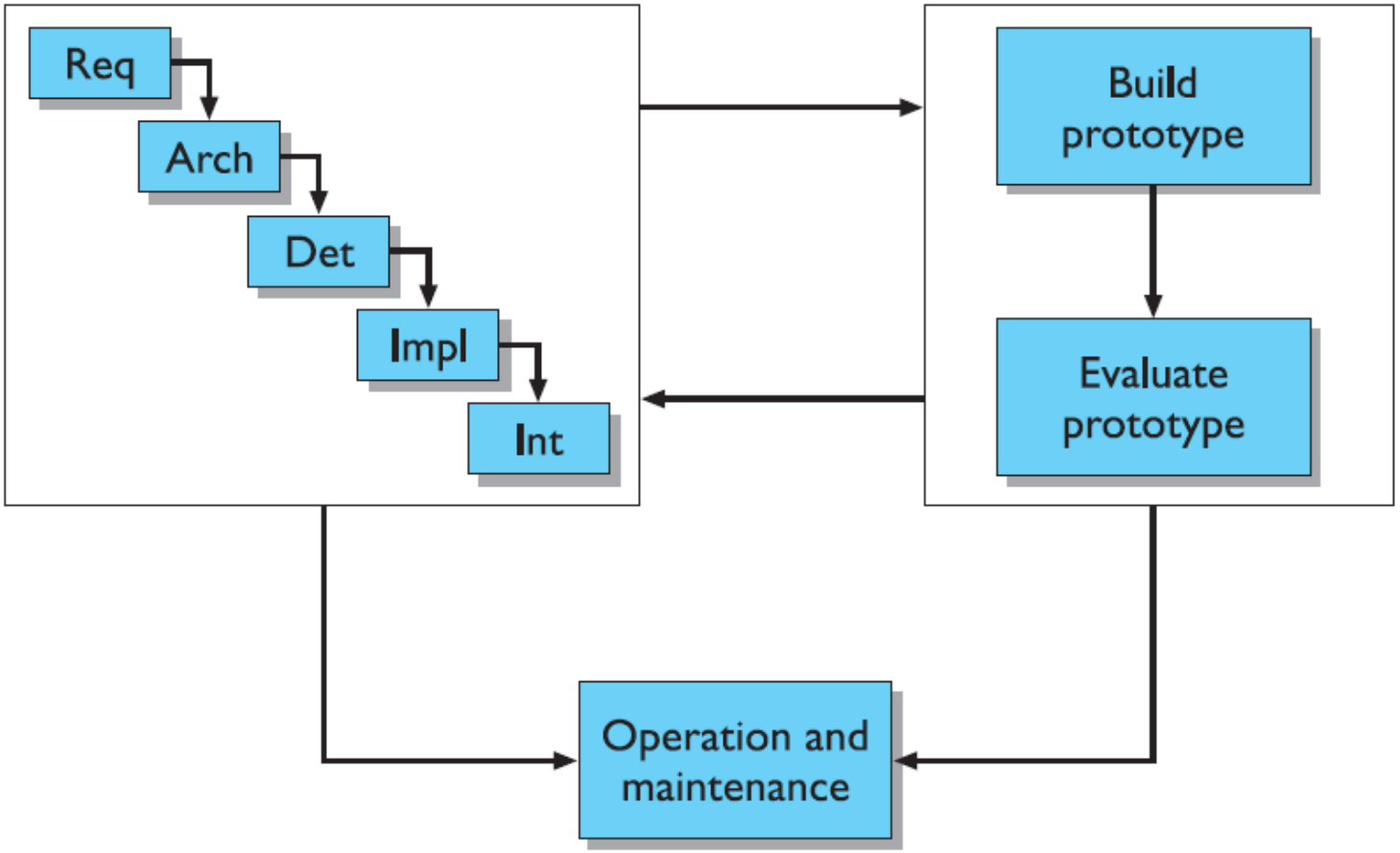
THROW-AWAY PROTOTYPING WITHIN REQUIREMENTS SPECIFICATION



INCREMENTAL PROTOTYPING WITHIN THE LIFE CYCLE



EVOLUTIONARY PROTOTYPING THROUGHOUT THE LIFE CYCLE



ITERATIVE DESIGN AND PROTOTYPING

- On the **management side**, there are several **potential problems**, as pointed out by **Sommerville**
  - **time**
  - **planning**
  - **non-functional features**
  - **contracts**

TECHNIQUES FOR PROTOTYPING

- **Storyboards**
  - need not be computer-based
  - can be animated
- **Limited functionality simulations**
  - some part of system functionality provided by designers.
  - tools like **HyperCard** are common for these
  - **Wizard of Oz** technique
    - With this technique, the designers can develop a limited functionality prototype and enhance its functionality in **evaluation by providing the missing functionality through human intervention.**
- **Warning about iterative design**
  - design decisions made at the very beginning of the prototyping process are wrong, in practice, **the design inertia can be so great** as never to overcome an initial bad decision. (Example, in clock (only 12 hours) 4 digits range are: 00:00 to 99:99, **63:00, 85:49**)
  - it is important to **understand the reason for the problem** and **not just detect the symptom. (**Example, not detect the 24/12 hour discrepancy, AM/PM)

DESIGN RATIONALE

- **Design rationale or reason** is the information that explains **why a computer system is the way it is**, including its **structural** or **architectural** description and its **functional** or **behavioral** description.
- Design rationale relates to an activity of both **reflection** (doing design rationale) and **documentation** (creating a design rationale) that occurs **throughout the entire life cycle**.
- **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

DESIGN RATIONALE (CONT'D)

Types of DR:

- **Process-oriented**
  - preserves order of deliberation and decision-making
- **Structure-oriented**
  - emphasizes post hoc structuring of considered design alternatives

- Two examples:
  - **Issue-based information system (IBIS)**
  - **Design space analysis**

ISSUE-BASED INFORMATION SYSTEM (IBIS)

- basis for much of design rationale research
- **process-oriented**
- main elements:

  issues

  - hierarchical structure with one ‘root’ issue

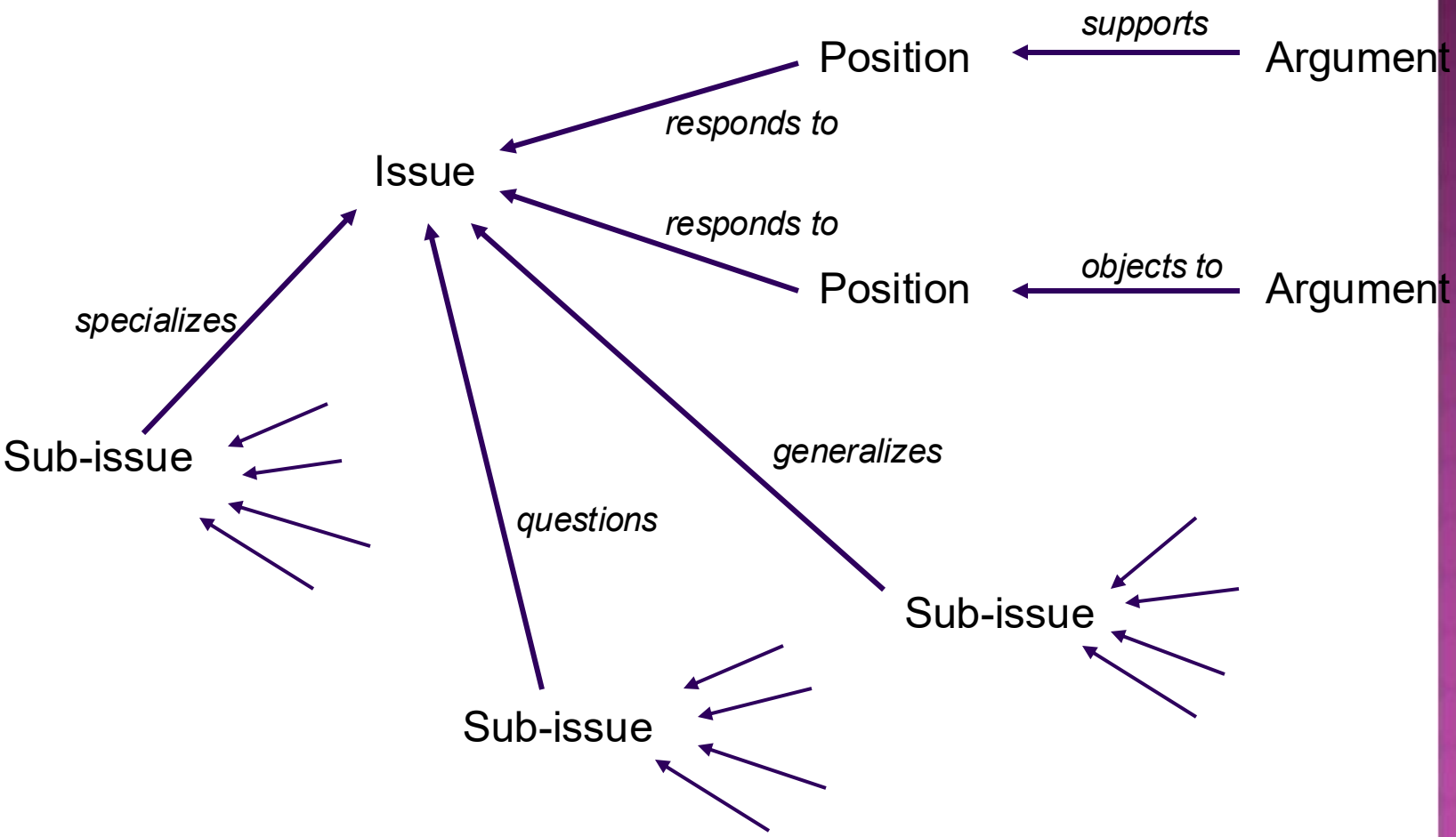
  positions

  - potential resolutions of an issue

  arguments

  - modify the relationship between positions and issues
- **gIBIS** is a **graphical** version

STRUCTURE OF GIBIS



DESIGN SPACE ANALYSIS

- **structure-oriented**
- **QOC** – hierarchical structure:

  questions (and sub-questions)

  – represent major issues of a design

  options

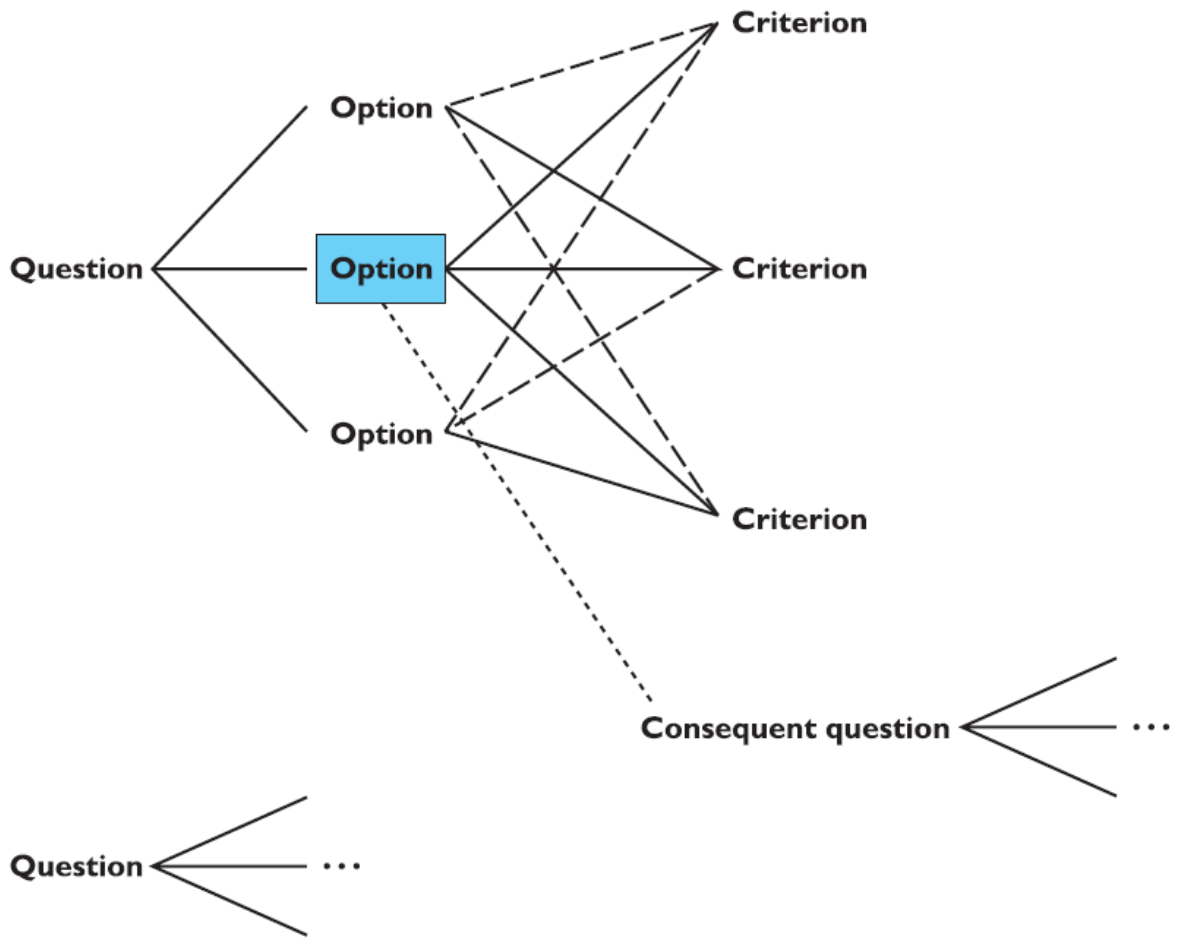
  – provide alternative solutions to the question

  criteria

  – the means to assess the options in order to make a choice

- Another structure-oriented technique, called **Decision Representation Language**(DRL).
- **DRL** – similar to QOC with a **larger language and more formal semantics.**

THE QOC NOTATION



PSYCHOLOGICAL DESIGN RATIONALE

- to support task-artefact cycle in which user tasks are affected by the systems they use
- 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

PRINCIPLES TO SUPPORT USABILITY

- **Learnability**
  - the ease with which **new users** can begin effective interaction and achieve maximal performance
- **Flexibility**
  - the **multiplicity** of ways the user and system exchange information
- **Robustness**
  - the **level of support** provided the user in determining successful achievement and assessment of goal-directed behaviour

PRINCIPLES OF LEARNABILITY

Table 7.1 Summary of principles affecting learnability

Principle	Definition	Related principles
Predictability	Support for the user to determine the effect of future action based on past interaction history	Operation visibility
Synthesizability	Support for the user to assess the effect of past operations on the current state	Immediate/eventual honesty
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	Guessability, affordance
Generalizability	Support for the user to extend knowledge of specific interaction within and across applications to other similar situations	–
Consistency	Likeness in input–output behavior arising from similar situations or similar task objectives	–

PRINCIPLES OF FLEXIBILITY

Table 7.2 Summary of principles affecting flexibility

Principle	Definition	Related principles
Dialog initiative	Allowing the user freedom from artificial constraints on the input dialog imposed by the system	System/user pre-emptiveness
Multi-threading	Ability of the system to support user interaction pertaining to more than one task at a time	Concurrent vs. interleaving, multi-modality
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	–
Substitutivity	Allowing equivalent values of input and output to be arbitrarily substituted for each other	Representation multiplicity, equal opportunity
Customizability	Modifiability of the user interface by the user or the system	Adaptivity, adaptability

PRINCIPLES OF ROBUSTNESS

Table 7.3 Summary of principles affecting robustness

Principle	Definition	Related principles
Observability	Ability of the user to evaluate the internal state of the system from its perceivable representation	Browsability, static/dynamic defaults, reachability, persistence, operation visibility
Recoverability	Ability of the user to take corrective action once an error has been recognized	Reachability, forward/ backward recovery, commensurate effort
Responsiveness	How the user perceives the rate of communication with the system	Stability
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	Task completeness, task adequacy

STANDARDS

- Set by **national** or **international bodies** to ensure **compliance** by a large community of designers.
- Standards can apply specifically to either the **hardware** or the **software** used to build the interactive system.
- **characteristics** between hardware and software is
  - **Underlying Theory**
  - **Change**

- **For Example,**

- **International Organization for Standardization (ISO)**
- **ISO 9241**, entitled **Ergonomic Requirements for Office Work with Visual Display Terminals (VDT)s**, has **17 parts**.
- **ISO 9241** defines usability as **effectiveness, efficiency and satisfaction** with which users accomplish tasks.
- **ISO 14915** covers **software ergonomics for multimedia user interfaces.**

ISO 9241 USABILITY SPECIFICATIONS

- **Usability**
  - The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments.
- **Effectiveness**
  - The accuracy and completeness with which specified users can achieve specified goals in particular environments.
- **Efficiency**
  - The resources expended in relation to the accuracy and completeness of goals achieved.
- **Satisfaction**
  - The comfort and acceptability of the work system to its users and other people affected by its use.

GUIDELINES

- more **suggestive** and **general**
- many **textbooks** and **reports** full of guidelines
- **abstract guidelines** (**principles**)
  - applicable during **early life cycle activities**
- **detailed guidelines** (**style guides**)
  - applicable during **later life cycle activities**
- understanding justification for guidelines aids in resolving conflicts

GUIDELINES - EXAMPLE

- The basic categories of the Smith and Mosier **guidelines** are:
  - **1. Data Entry**
  - **2. Data Display**
  - **3. Sequence Control**
  - **4. User Guidance**
  - **5. Data Transmission**
  - **6. Data Protection**

GUIDELINES - EXAMPLE

1. **Data Entry**

1.1 *Position Designation*

1.1–1 **Distinctive Cursor**

For position designation on an electronic display, provide a movable cursor with distinctive visual features (shape, blink, etc.).

Exception When position designation involves only selection among displayed alternatives, highlighting selected items might be used instead of a separately displayed cursor.

Comment When choosing a cursor shape, consider the general content of the display. For instance, an underscore cursor would be difficult to see on a display of underscored text, or on a graphical display containing many other lines.

Comment If the cursor is changed to denote different functions (e.g. to signal deletion rather than entry), then each different cursor should be distinguishable from the others.

Comment If multiple cursors are used on the same display (e.g. one for alphanumeric entry and one for line drawing), then each cursor should be distinguishable from the others.

Reference Whitfield, Ball and Bird, 1983

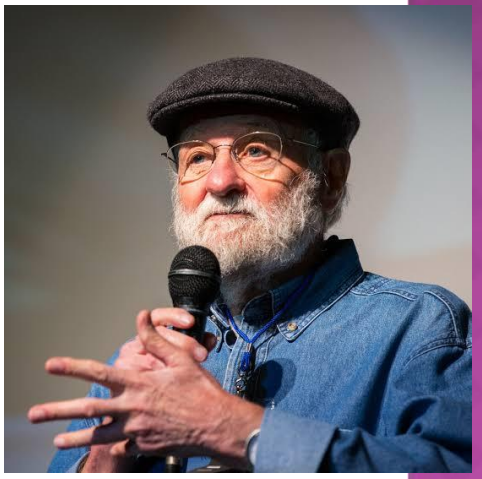
See also 1.1–17 Distinctive multiple cursors
4.0–9 Distinctive cursor

GOLDEN RULES AND HEURISTICS

- "**Broad brush**" design rules
- **Useful check list** for good design
- **Better design using these than using nothing!**
- Golden rules and Heuristics
  - **Nielsen's 10 Heuristics**
  - **Shneiderman's 8 Golden Rules**
  - **Norman's 7 Principles**

Jakob Nielsen

Ben Shneiderman



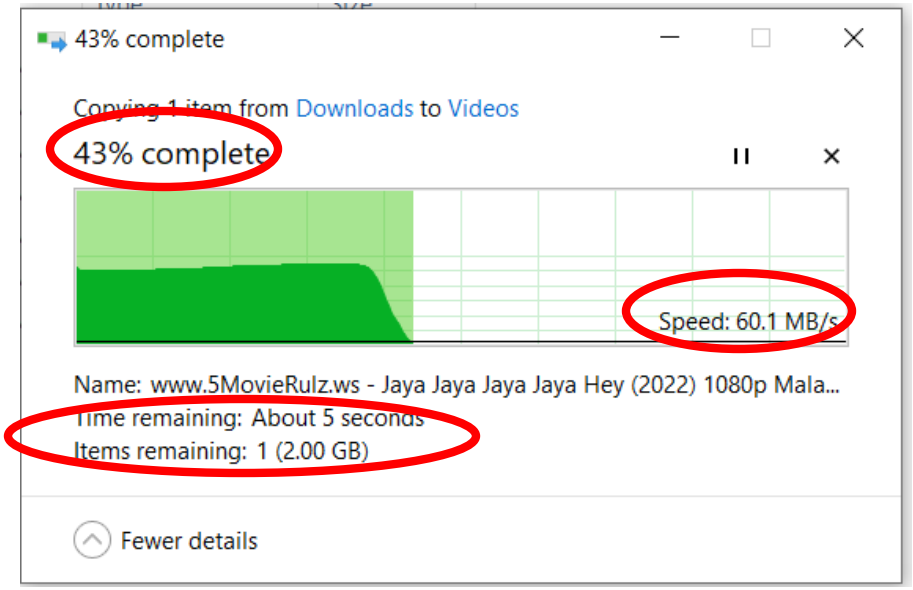
Don Norman

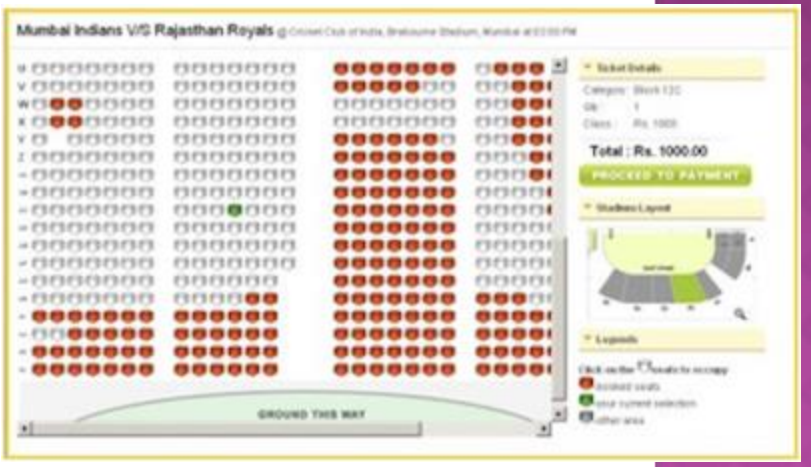
Nielsen's 10 Heuristics

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose and recover from errors
10. Help and documentation

1. VISIBILITY OF SYSTEM STATUS

- **Always keep users informed** about what is going on, through appropriate feedback **within reasonable time**.
- For example, if a system operation will take some time, give an **indication of how long and how much is complete.**



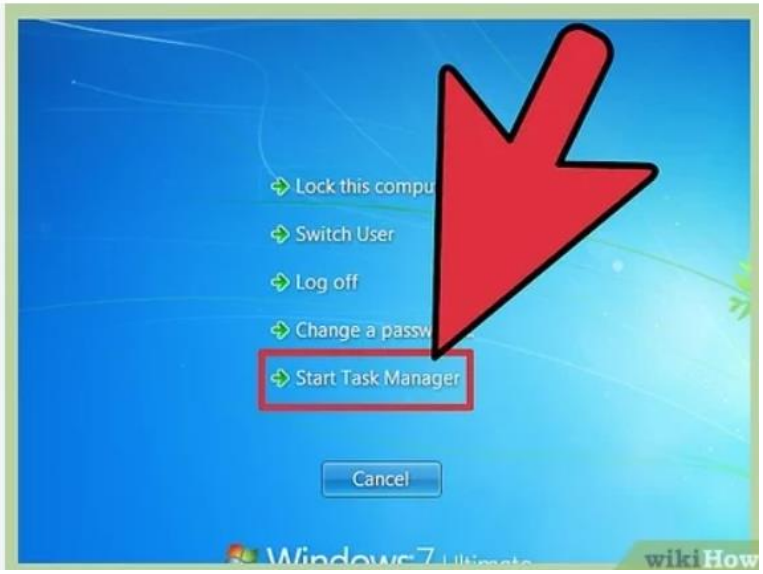


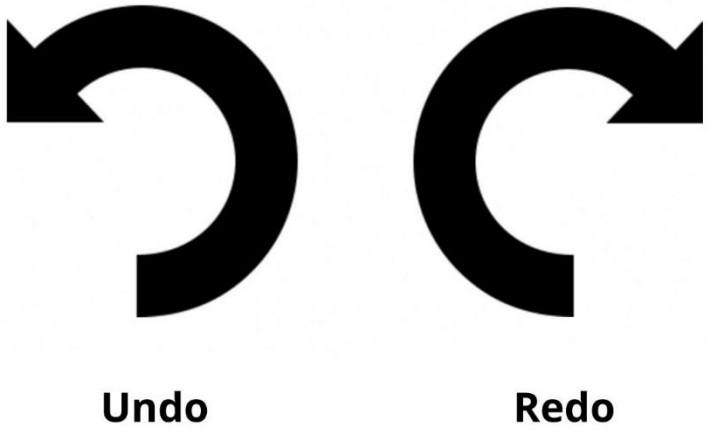
2. MATCH BETWEEN SYSTEM AND THE REAL WORLD

- The system should speak the **user's language**, with **words**, **phrases** and **concepts familiar to the user, rather than system-oriented terms**.
- Follow real-world conventions, making information appear in natural and logical order.

3. USER CONTROL AND FREEDOM

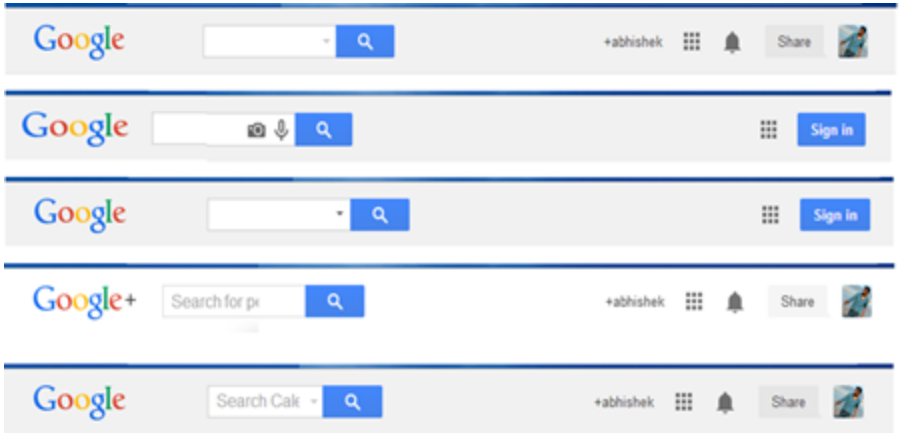
- Users often choose system functions by mistake and need a clearly marked '**emergency exit**' to leave the unwanted state without having to go through an extended dialog.
- Support **undo** and **redo**.





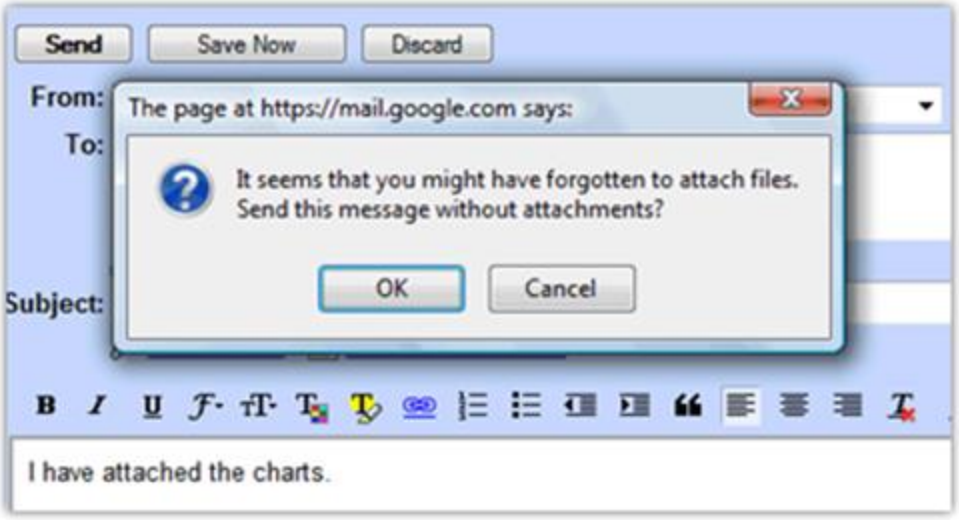
4. CONSISTENCY AND STANDARDS

- **Users should not have to wonder** whether **words**, **situations** or **actions mean the same thing in different contexts.**
- Follow **platform conventions** and **accepted standards.**



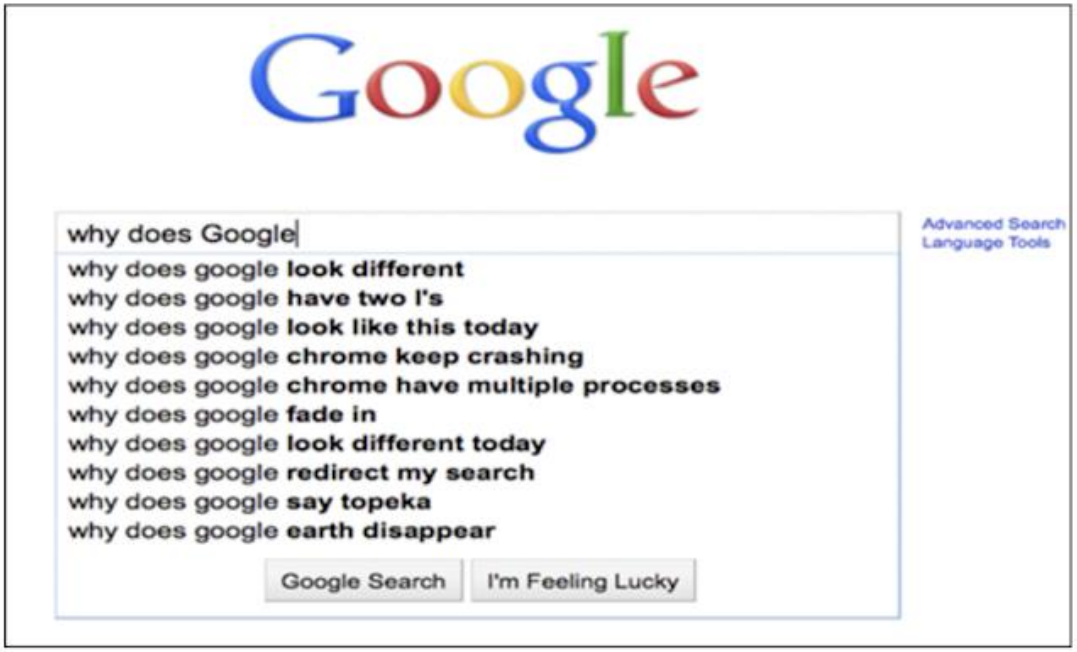
5. ERROR PREVENTION

- **Make it difficult to make errors.**
- Even better than **good error messages** is a careful design that prevents a problem from occurring in the first place.
- Either eliminate **error-prone conditions** or check for them and present users with a **confirmation option before they commit to the action.**



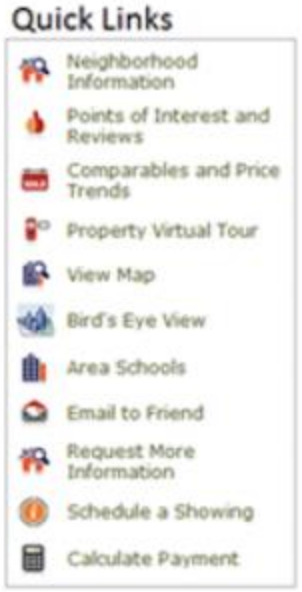
6. RECOGNITION RATHER THAN RECALL

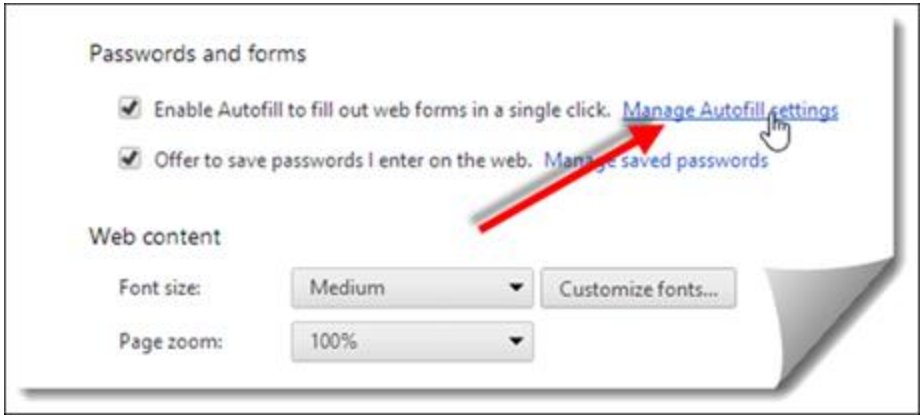
- Minimize the user's memory load by make **objects, actions** and **options visible**.
- The **user should not have to remember information** from one part of the dialog to another.
- **Instructions for use of the system should be visible** or easily retrievable whenever appropriate.



7. FLEXIBILITY AND EFFICIENCY OF USE

- **Allow users to tailor frequent actions.**
- Accelerators – unseen by the novice user – may often speed up the interaction for the expert user to such an extent that **the system can cater to both inexperienced** and **experienced users.**



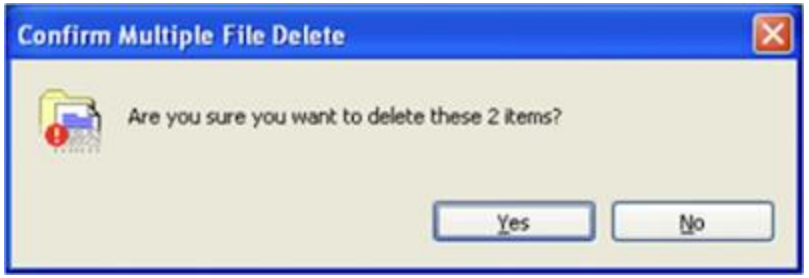


8. AESTHETIC AND MINIMALIST DESIGN

- **Dialogs should not contain information that is irrelevant or rarely needed**.
- Every extra unit of information in a dialog competes with the relevant units of information and diminishes their relative visibility.

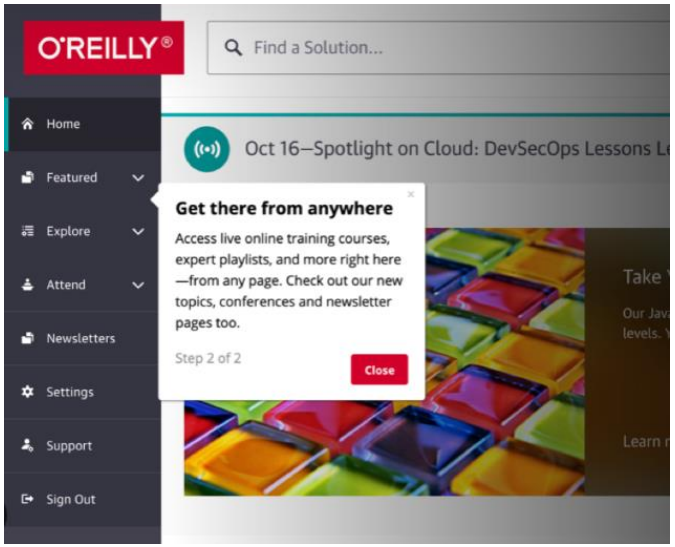
9. HELP USERS RECOGNIZE, DIAGNOSE AND RECOVER FROM ERRORS

- Error messages should be expressed in
  - **plain language** (no codes),
  - **precisely indicate the problem,** and
  - **constructively suggest a solution.**



10. HELP AND DOCUMENTATION

- Few systems can be used with no instructions so it may be **necessary to provide help and documentation.**
- Any such information **should be easy to search, focused on the user's task**, list **concrete steps to be carried out**, and **not be too large.**

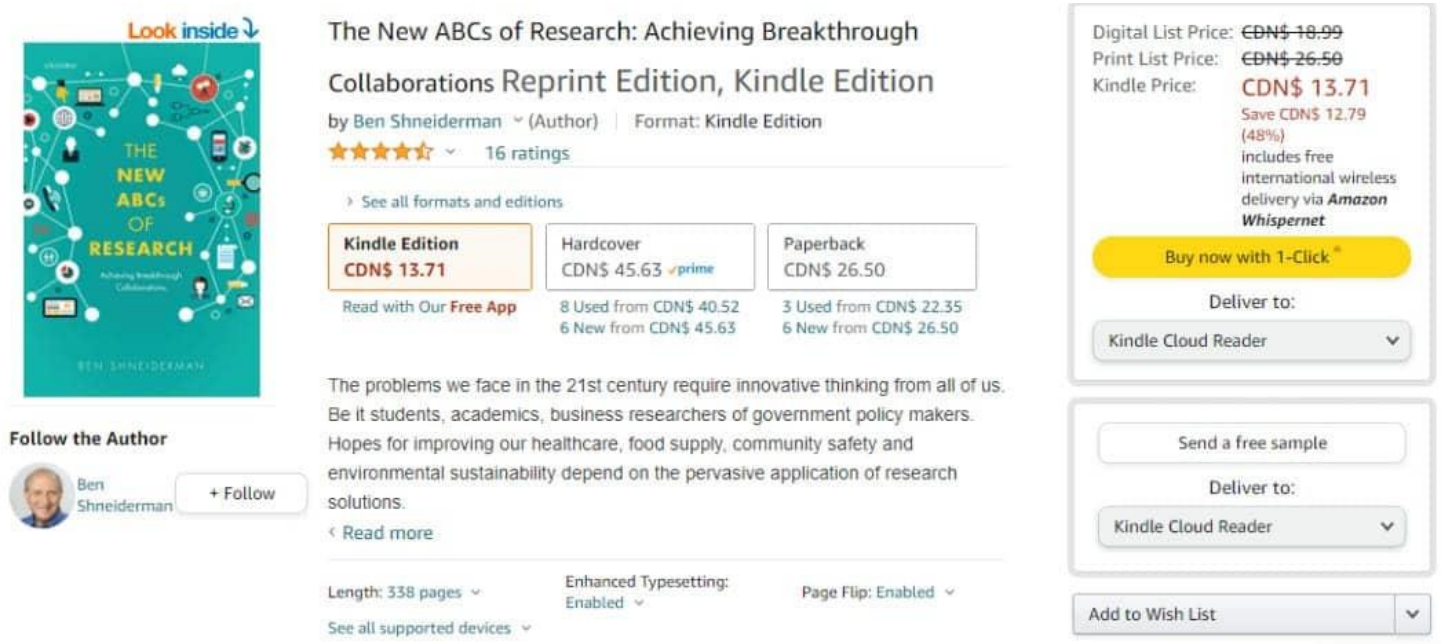


SHNEIDERMAN'S 8 GOLDEN RULES

1. Strive for consistency
2. Enable frequent users to use shortcuts
3. Offer informative feedback
4. Design dialogs to yield closure
5. Offer error prevention and simple error handling
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

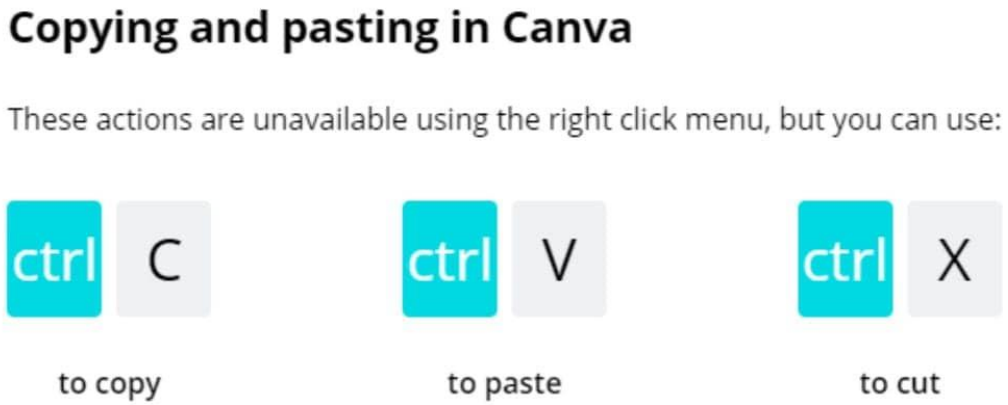
1. STRIVE FOR CONSISTENCY

- ***Strive for consistency*** in **action sequences**, **layout**, **terminology**, **command use** and so on.
- Whether it is the **layout**, the **size of the button**, the **color code** or the **tone** used when writing the page, it is important **to be consistent throughout the site.**
- This consistency will allow you to develop **your identity** and **not lose users as they navigate your site**.



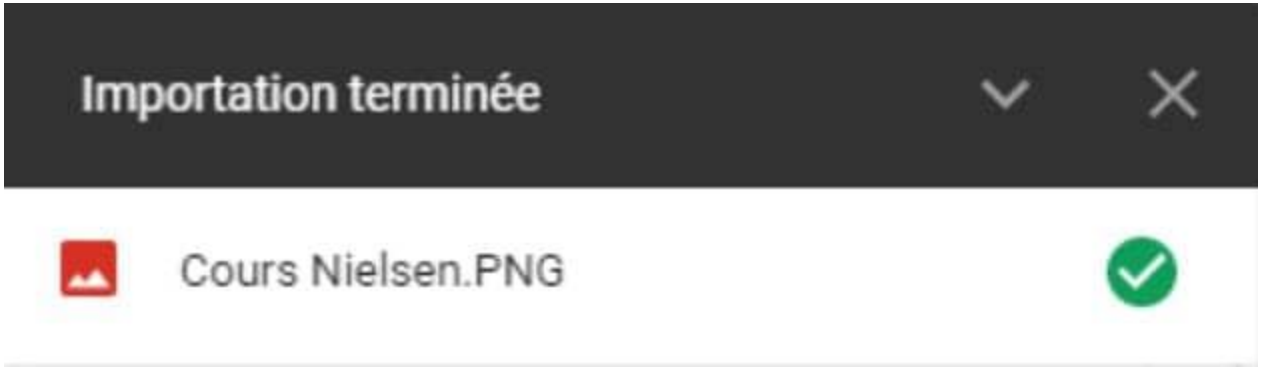
2. ENABLE FREQUENT USERS TO USE SHORTCUTS

- ***Enable frequent users to use shortcuts***, such as **abbreviations**, **special key sequences** and **macros**, to perform regular, familiar actions more quickly.



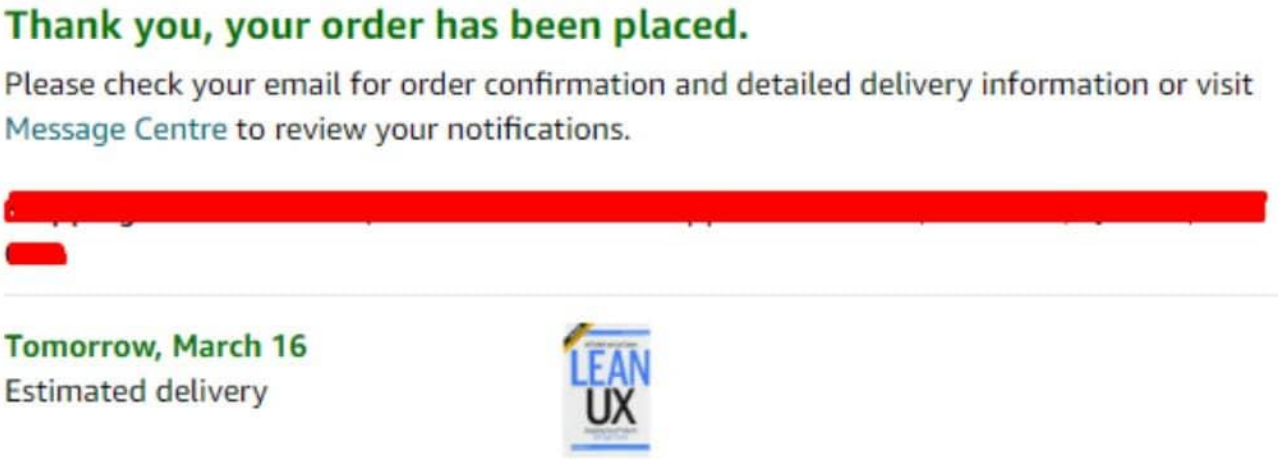
3. OFFER INFORMATIVE FEEDBACK

- ***Offer informative feedback*** for **every user action,** at a **level appropriate** to the magnitude of the action.



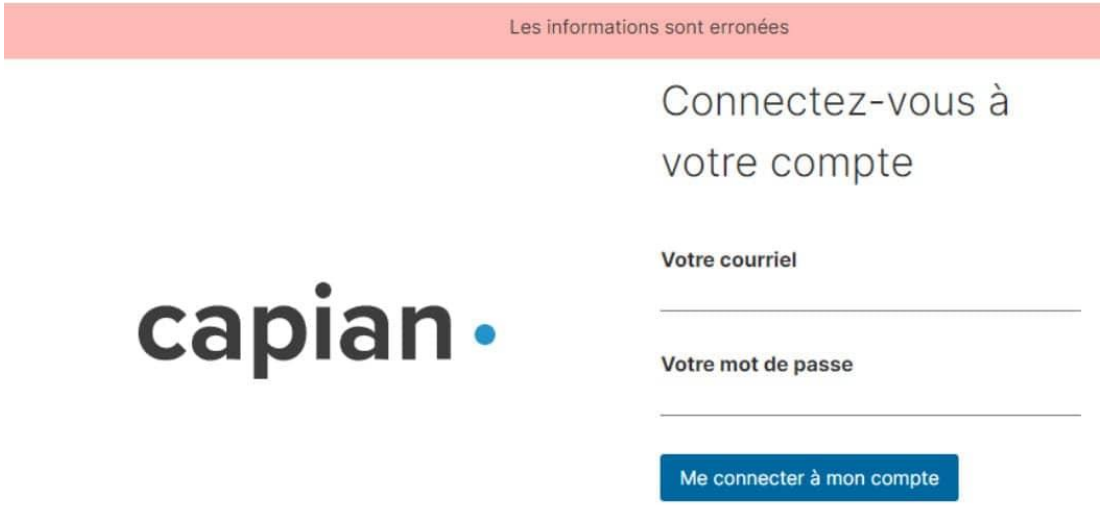
4. DESIGN DIALOGS TO YIELD CLOSURE

- ***Design dialogs to yield closure*** so that the **user knows when they have completed a task**.
  - Thank you message.
  - Validation message.
  - Summary message during a purchase.



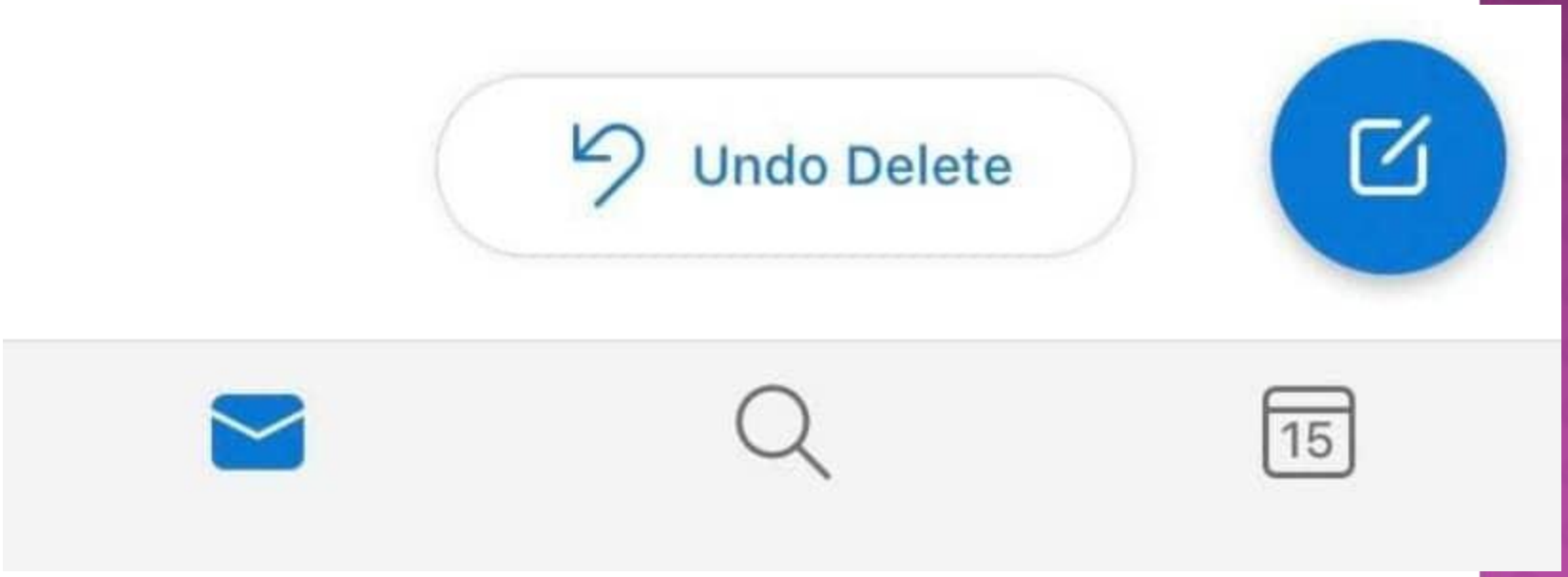
5. OFFER ERROR PREVENTION AND SIMPLE ERROR HANDLING

- ***Offer error prevention and simple error handling*** so that, ideally, **users are prevented from making mistakes** and, if they do, they are offered **clear** and **informative instructions** **to enable them to recover.**
- A good interface should be designed to **avoid errors as much as possible**.
- For example, Login error message.



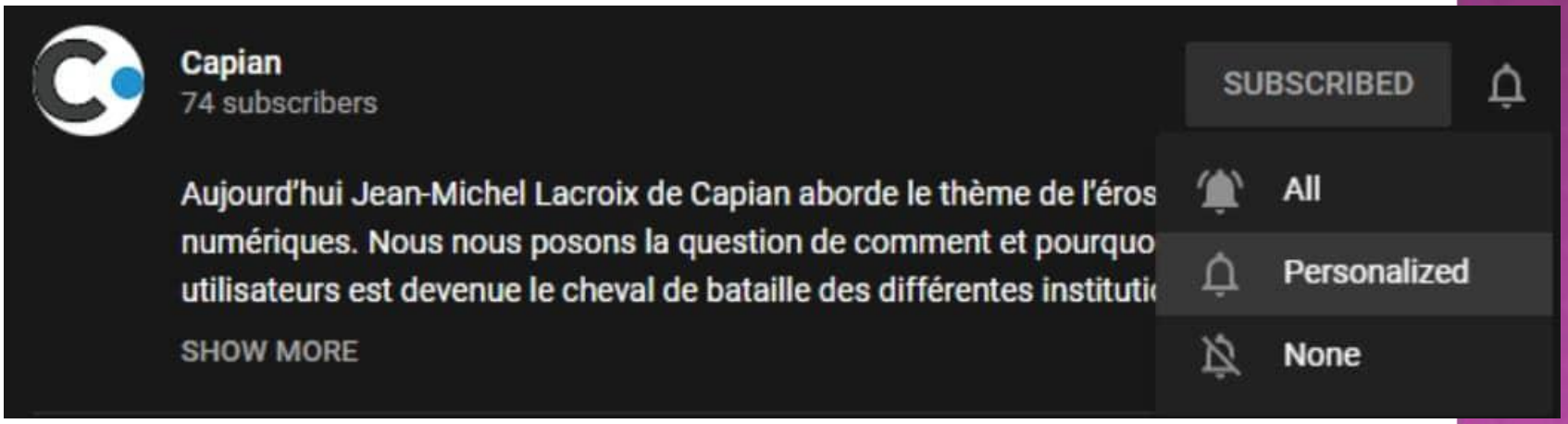
6. PERMIT EASY REVERSAL OF ACTIONS

- ***Permit easy reversal of actions*** in order to relieve **anxiety** and **encourage exploration**, since the user knows that he can always return to the **previous state**.



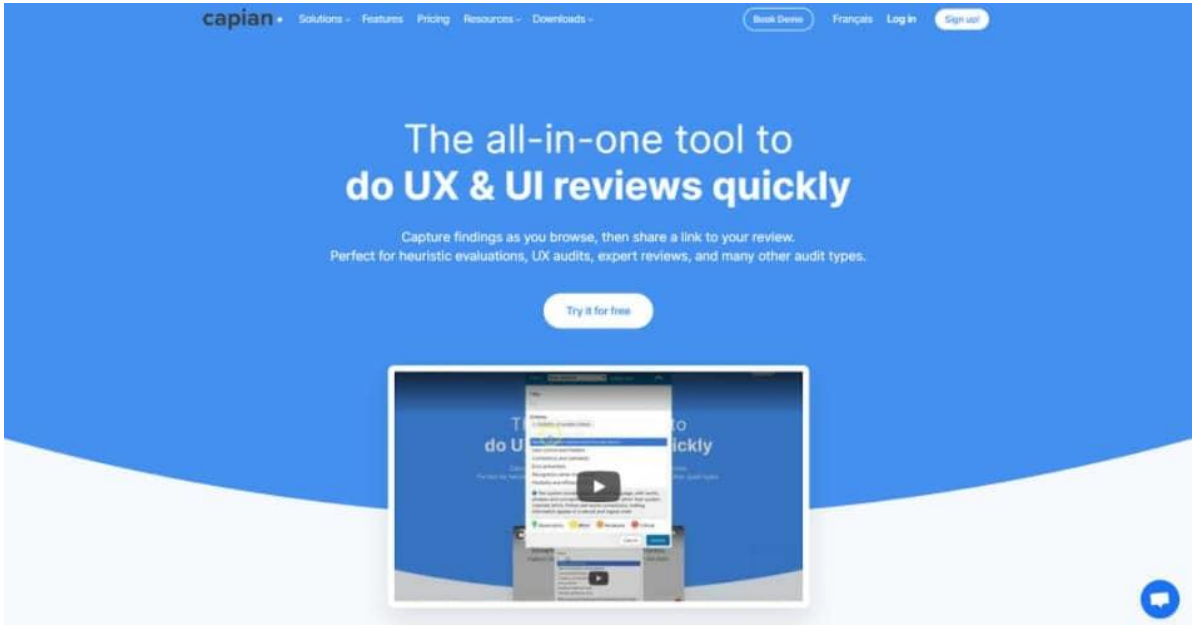
7. SUPPORT INTERNAL LOCUS OF CONTROL

- ***Support internal locus of control*** so that the **user** is **in control of the system**, which responds to his actions.
- We need to **give control and freedom** to the **users**, so that they can feel that they are in control of the system themselves, giving them some form of free will helps to reassure the user.
- For Example, YouTube is giving freedom to the users.



8. REDUCE SHORT-TERM MEMORY LOAD

- ***Reduce short-term memory load*** by keeping displays **simple**, **consolidating multiple page displays** and providing time for learning action sequences.
- For example, Capian home page



NORMAN'S 7 PRINCIPLES

1. Use both knowledge in the world and knowledge in the head.
2. Simplify the structure of tasks.
3. Make things visible: bridge the gulfs of Execution and Evaluation.
4. Get the mappings right.
5. Exploit the power of constraints, both natural and artificial.
6. Design for error.
7. When all else fails, standardize.

1. USE BOTH KNOWLEDGE IN THE WORLD AND KNOWLEDGE IN THE HEAD.

- **People work better** when the knowledge they need to do a task is available **externally** – either **explicitly** or through the **constraints** imposed by the environment.
- But experts also need to be able to internalize regular tasks to increase their efficiency.
- So syste**ms should provide the necessary knowledge within the environment** and their operation should be transparent to support the user in building an appropriate mental model of what is going on.

2. SIMPLIFY THE STRUCTURE OF TASKS.

- **Tasks need to be simple** in order **to avoid complex problem solving** and **excessive memory load**.
- **Provide mental aids** to help the user keep track of stages in a more complex task.
- **Use technology** to provide the user with more information about the task and better feedback.
- **Automate the task** or part of it, as long as this does not detract from the user's experience.
- **Change the nature of the task** so that it becomes something more simple.
- **Not to take control away from the user.**

3. MAKE THINGS VISIBLE: BRIDGE THE GULFS OF EXECUTION AND EVALUATION.

- Bridge the **gulfs of execution** and **evaluation**.
- The interface should make clear **what the system can do** and **how this is achieved**, and should enable the user to see clearly the effect of their actions on the system.

4. GET THE MAPPINGS RIGHT.

- **User intentions should map clearly** onto **system controls**.
- **User actions should map clearly** onto **system events.**
- So it should be clear what does what and by how much.
- Controls, sliders and dials should reflect the task – so a **small movement has a small effect** and **a large movement a large effect.**

5. EXPLOIT THE POWER OF CONSTRAINTS, BOTH NATURAL AND ARTIFICIAL.

- **Constraints** are **things in the world** that **make it impossible to do anything** but the correct action in the correct way.
- A simple example is a **jigsaw puzzle**, where the pieces only fit together in one way.
- Here the physical constraints of the design guide the user to complete the task.

6. DESIGN FOR ERROR.

- To err is human, so **anticipate the errors** the user could make and **design recovery** into the system.

7. WHEN ALL ELSE FAILS, STANDARDIZE.

- **If there are no natural mappings** then **arbitrary mappings should be standardized** so that users only have to learn them once.
- It is this **standardization principle** that enables drivers to get into a new car and drive it with very little difficulty – key controls are standardized.
- Occasionally one might switch on the indicator lights instead of the windscreen wipers, but the **critical controls** (**accelerator, brake, clutch, steering**) **are always the same**.

HCI DESIGN PATTERNS

- An approach to **reusing knowledge** about **successful design solutions**
- Originated in architecture: Alexander
- A **pattern** is an invariant solution to a **recurrent problem within a specific context**.
- Examples
  - **Light on Two Sides of Every Room (architecture)**
  - **Go back to a safe place (HCI)**
- Patterns do not exist in isolation but are linked to other patterns in *languages* which enable **complete designs** to be generated.

HCI DESIGN PATTERNS (CONT.)

- **Characteristics** of patterns
  - capture design practice not theory
  - capture the essential common properties of good examples of design
  - represent design knowledge at varying levels: social, organisational, conceptual, detailed
  - embody values and can express what is humane in interface design
  - are intuitive and readable and can therefore be used for communication between all stakeholders
  - a pattern language should be generative and assist in the development of complete designs.

NUCLEAR REACTOR MAIN CONTROL PANEL