Lecture #08

S1 –Interaksi Manusia Komputer

HCI in the software process
Design rules

Novy NRA Mokobombang, ST, MsTM, PhD
Informatics Hasanuddin University

Material by: Alan Dix (The author of Human-Computer Interaction book – chapter 6 & 7)

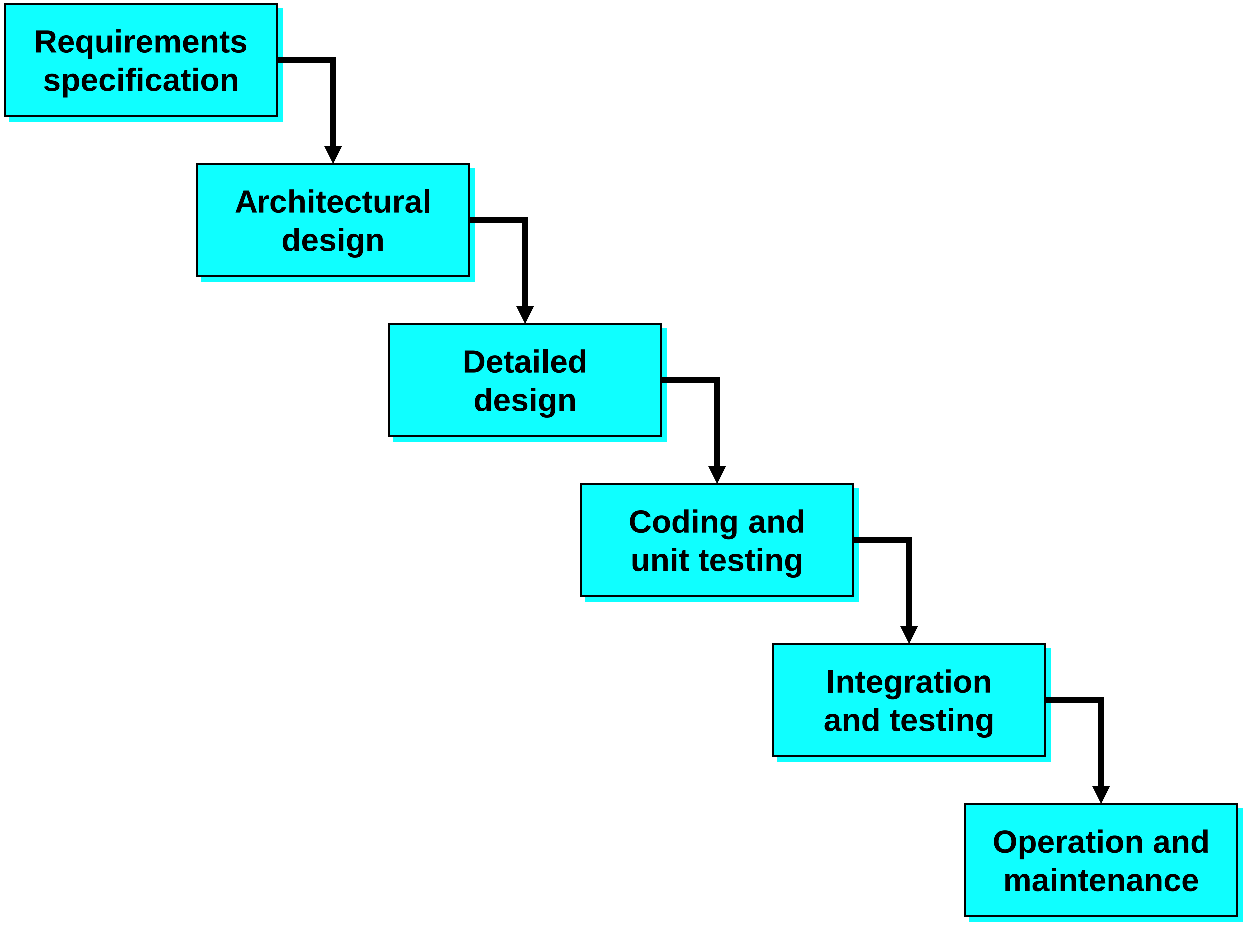
HCI in the software process

- Software engineering and the design process for interactive systems
- Usability engineering
- Iterative design and prototyping
- Design rationale

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



Activities in the life cycle

Requirements specification

designer and customer try capture what the system is expected to provide can be expressed in natural language or more precise languages, such as a task analysis would provide

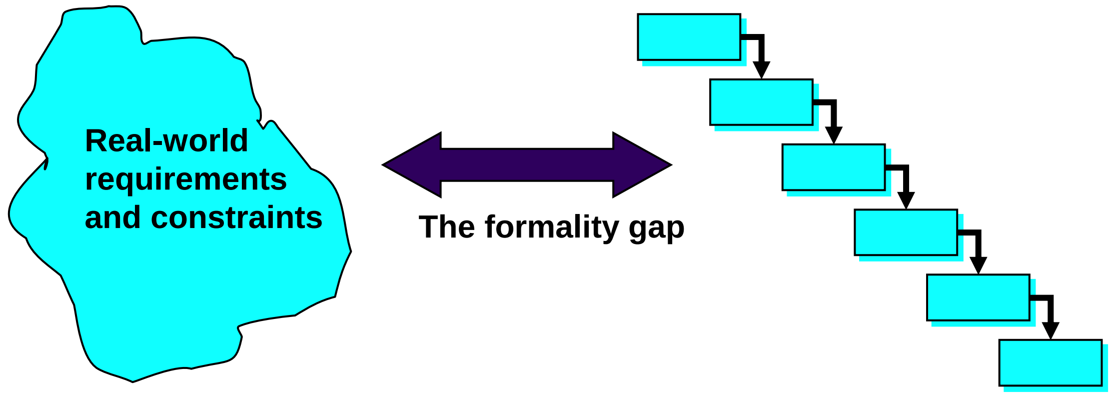
Architectural design

high-level description of how the system will provide the services required factor system into major components of the system and how they are interrelated needs to satisfy both functional and nonfunctional requirements

Detailed design

refinement of architectural components and interrelations to identify modules to be implemented separately the refinement is governed by the nonfunctional requirements

Verification and validation



Verification

designing the product right

Validation

designing the right product

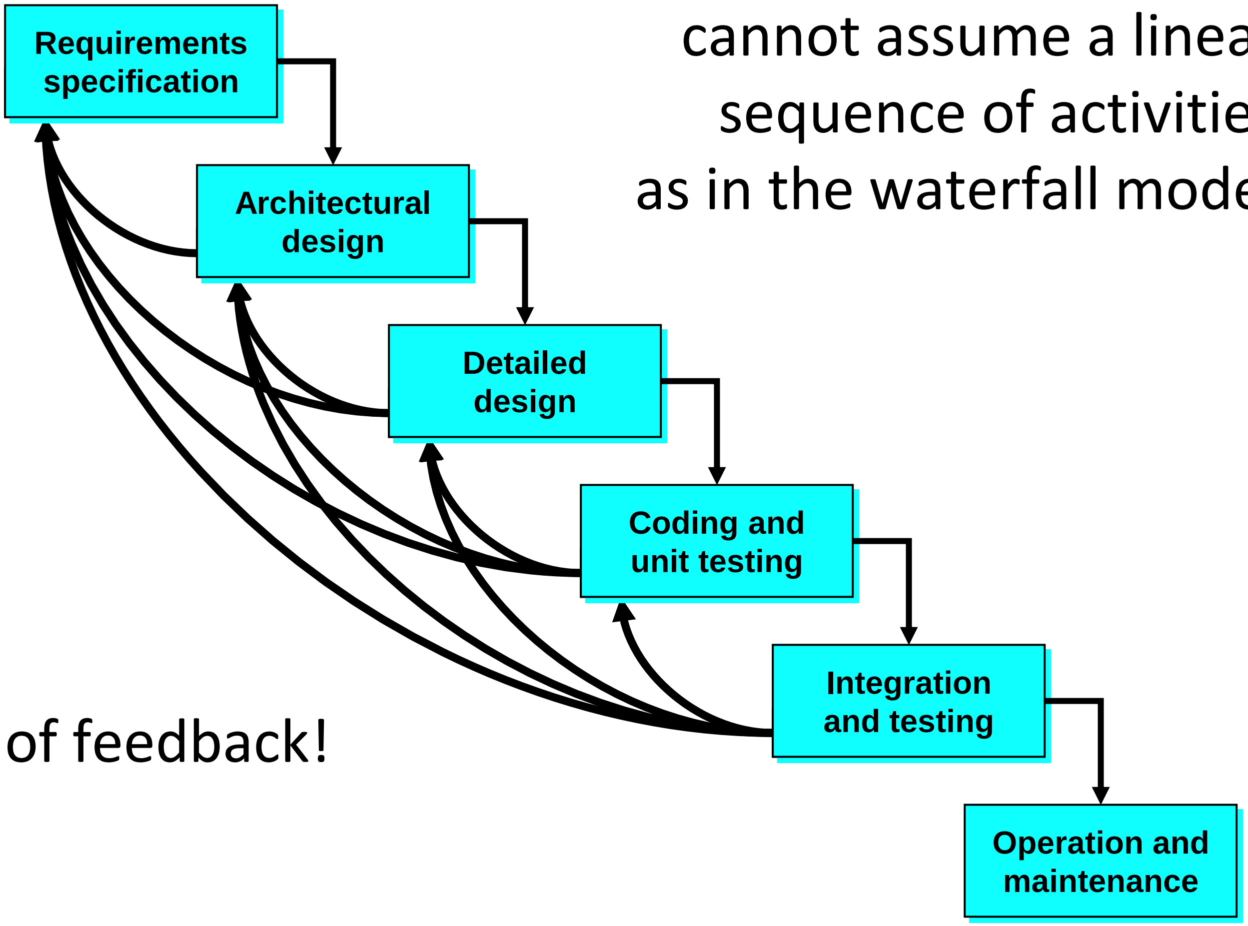
The formality gap

validation will always rely to some extent on subjective means of proof

Management and contractual issues

design in commercial and legal contexts

The life cycle for interactive systems



cannot assume a linear sequence of activities as in the waterfall model

lots of feedback!

Usability engineering

The ultimate test of usability based on measurement of user experience

Usability engineering demands that specific usability measures be made explicit as requirements

Usability specification

- usability attribute/principle
- measuring concept
- measuring method
- now level/ worst case/ planned level/ best case

Problems

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

Iterative design and prototyping

- Iterative design overcomes inherent problems of incomplete requirements

- Prototypes
  - simulate or animate some features of intended system
  - different types of prototypes
    - throw-away
    - incremental
    - evolutionary

- Management issues
  - 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

Warning about iterative design

- design inertia – early bad decisions stay bad
- diagnosing real usability problems in prototypes….
  - …. and not just the symptoms

Design rationale

Design rationale is information that explains why a computer system is the way it is.

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)

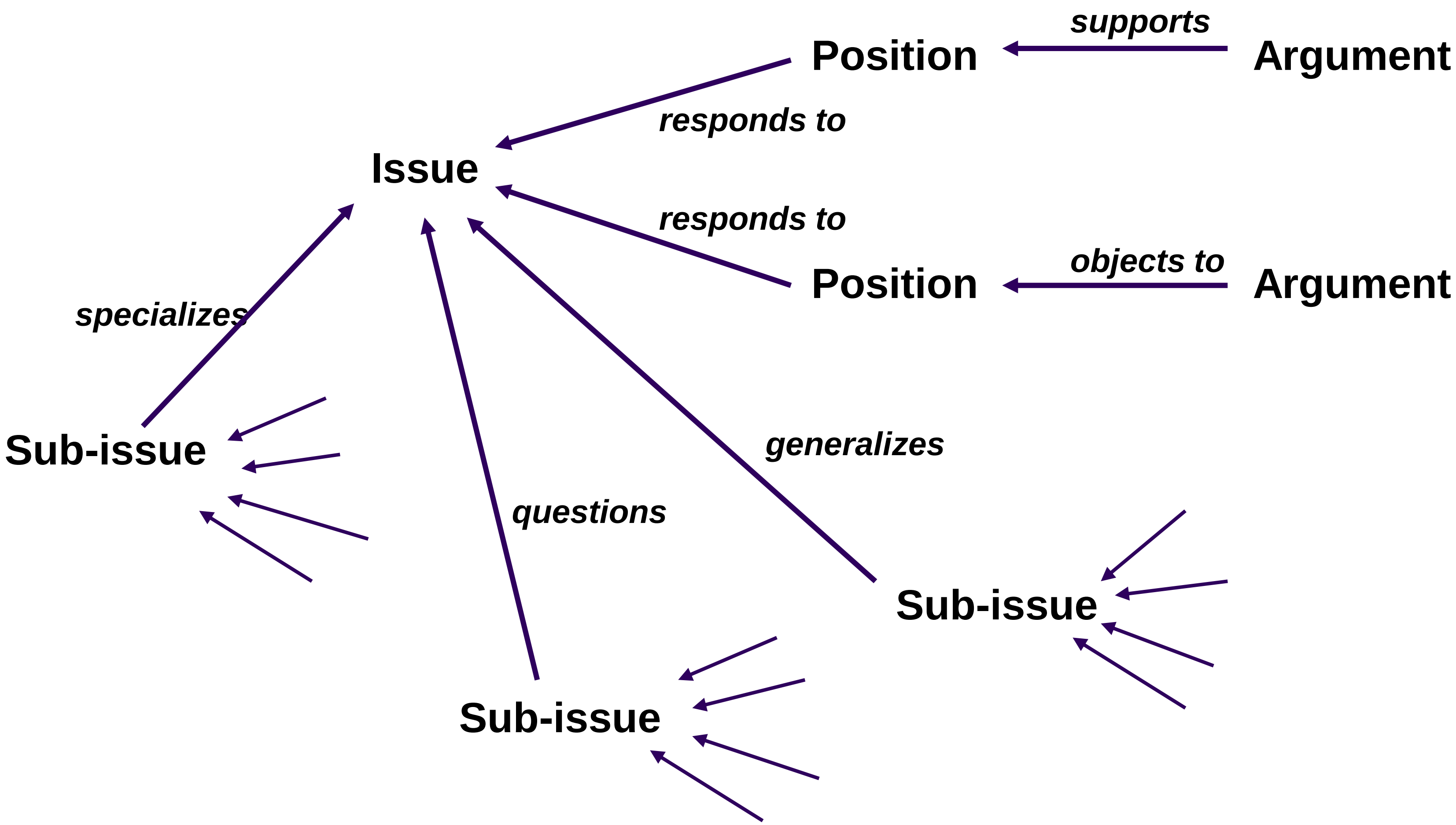
- basis for much of design rationale research
- process-oriented
- main elements:

  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

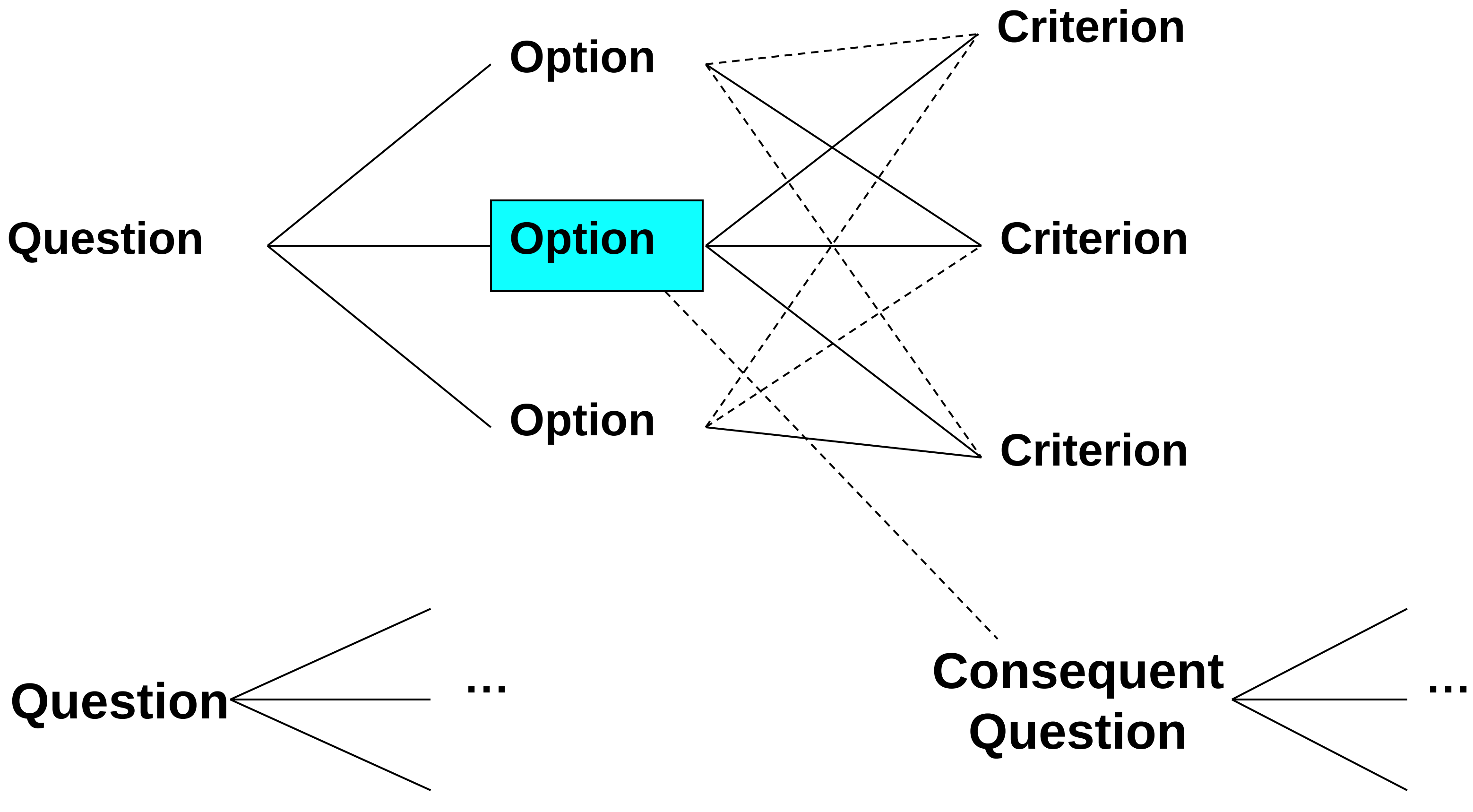
  options

  – provide alternative solutions to the question

  criteria

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

Summary

The software engineering life cycle

- distinct activities and the consequences for interactive system design

Usability engineering

- making usability measurements explicit as requirements

Iterative design and prototyping

- limited functionality simulations and animations

Design rationale

- recording design knowledge
- process vs. structure

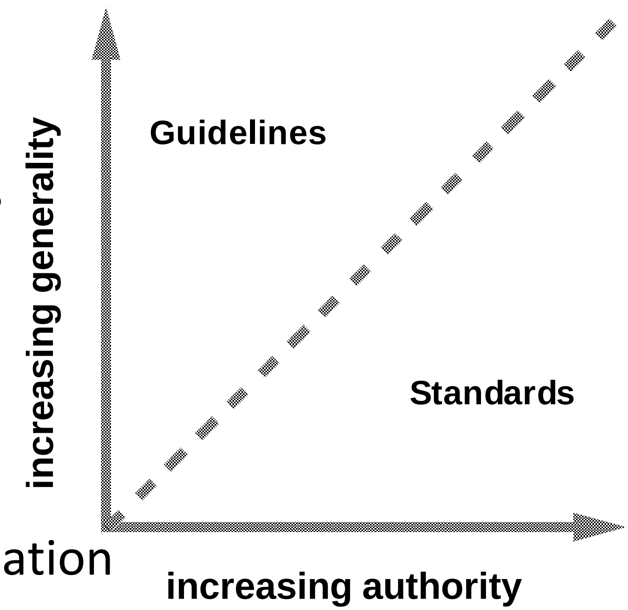
design rules

Designing for maximum usability
– the goal of interaction design

- Principles of usability
  - general understanding
- Standards and guidelines
  - direction for design
- Design patterns
  - capture and reuse design knowledge

types of design rules

- principles
  - abstract design rules
  - low authority
  - high generality
- standards
  - specific design rules
  - high authority
  - limited application
- guidelines
  - lower authority
  - more general application



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

Predictability

- determining effect of future actions based on past interaction history
- operation visibility

Synthesizability

- assessing the effect of past actions
- immediate vs. eventual honesty

Principles of learnability (ctd)

Familiarity

- how prior knowledge applies to new system
- guessability; affordance

Generalizability

- extending specific interaction knowledge to new situations

Consistency

- likeness in input/output behaviour arising from similar situations or task objectives

Principles of flexibility

Dialogue initiative

- freedom from system imposed constraints on input dialogue
- system vs. user pre-emptiveness

Multithreading

- ability of system to support user interaction for more than one task at a time
- concurrent vs. interleaving; multimodality

Task migratability

- passing responsibility for task execution between user and system

Principles of flexibility (ctd)

Substitutivity

- allowing equivalent values of input and output to be substituted for each other
- representation multiplicity; equal opportunity

Customizability

- modifiability of the user interface by user (adaptability) or system (adaptivity)

Principles of robustness

Observability

- ability of user to evaluate the internal state of the system from its perceivable representation
- browsability; defaults; reachability; persistence; operation visibility

Recoverability

- ability of user to take corrective action once an error has been recognized
- reachability; forward/backward recovery; commensurate effort

Principles of robustness (ctd)

Responsiveness

- how the user perceives the rate of communication with the system
- Stability

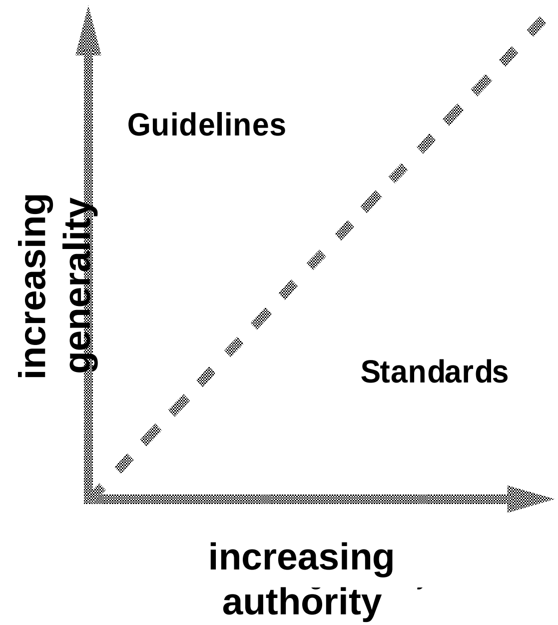
Task conformance

- degree to which system services support all of the user's tasks
- task completeness; task adequacy

Using design rules

Design rules

- suggest how to increase usabilit
- differ in generality and authorit



Standards

- set by national or international bodies to ensure compliance by a large community of designers standards require sound underlying theory and slowly changing technology
- hardware standards more common than software high authority and low level of detail
- ISO 9241 defines usability as effectiveness, efficiency and satisfaction with which users accomplish tasks

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

Golden rules and heuristics

- “Broad brush” design rules
- Useful check list for good design
- Better design using these than using nothing!
- Different collections e.g.
  - Nielsen’s 10 Heuristics (see Chapter 9)
  - Shneiderman’s 8 Golden Rules
  - Norman’s 7 Principles

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*

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.*

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.

Summary

Principles for usability

- repeatable design for usability relies on maximizing benefit of one good design by abstracting out the general properties which can direct purposeful design
- The success of designing for usability requires both creative insight (new paradigms) and purposeful principled practice

Using design rules

- standards and guidelines to direct design activity