Coping with diversity in HCI:

Techniques for Adaptable and Adaptive Interaction

C. Stephanidis, D. Akoumianakis, A. Paramythis

Foundation for Research and Technology-Hellas (FORTH)

Institute of Computer Science (ICS)

Assistive Technology & Human Computer Interaction Laboratory

E-mails: {cs,demosthe,alpar}@ics.forth.gr

Presentation plan

Part I - Dimensions of diversity (14.00 - 14.30)

Part II - Adaptation in HCI- The challenges (14.30 - 15.30)

- ✔ HCI Design Traditions
- ✔ User interface architecture

Coffee break (15.30 - 16.00)

Part III - Towards meeting the challenges (16.00 - 17.00)

- ✔ The Unified User Interface Design Method
- ✔ The Unified user interface architecture

Part IV - Example case study (17.00 -17.15)

- ✔ The AVANTI Browser

Part V - Summary, Conclusions & Discussion (17.15 - 17.30)

Part I: Dimensions of diversity

Dimensions of Diversity

- Diversity in HCI arises from
  - ✔ variety in the user population
  - ✔ variety in the nature of work
  - ✔ variety in the context of use
  - ✔ variety in the user access

Diversity in user population (1/4)

- Experts versus novices
- Young versus elderly
- Able-bodied versus people with disabilities
- Business workers versus casual/residential users
- Users versus non-users

Diversity in user population (2/4)

- Dimensions of user differentiation from an information processing perspective (Sternberg, 1985):
  - ✔ general intellectual ability
  - ✔ verbal ability
  - ✔ reading ability
  - ✔ second-language abilities
  - ✔ individual differences in learning and memory
  - ✔ mathematical ability
  - ✔ mental imagery ability
  - ✔ deductive reasoning ability
  - ✔ inductive reasoning ability
  - ✔ problem-solving ability

Diversity in user population (3/4)

- General categories of individual differences from (Browne, Norman and Riches, 1990):
  - ✔ psycho-motor skills
  - ✔ capability
  - ✔ understanding
  - ✔ expectations
  - ✔ motives
  - ✔ requirements
  - ✔ cognitive strategies
  - ✔ cognitive abilities
  - ✔ preferences

Diversity in user population (4/4)

- A different perspective from (Benyon, 1999):

- Physiological
  - ✔ cerebral hemisphericity
  - ✔ vision
  - ✔ hearing
  - ✔ mobility / dexterity

- Psychological
  - ✔ intelligence
  - ✔ cognitive style
  - ✔ personality

- Socio-cultural
  - ✔ language
  - ✔ culture
  - ✔ environmental

Variety in the nature of work

- Business versus residential activities
- Collaborative tasks
- New virtualities which augment human capabilities for problem solving:
  - ✔ Digital libraries
  - ✔ Virtual cities
  - ✔ Virtual university
  - ✔ Electronic market places
  - ✔ Digital money
  - ✔ etc.

Variety in the context of use

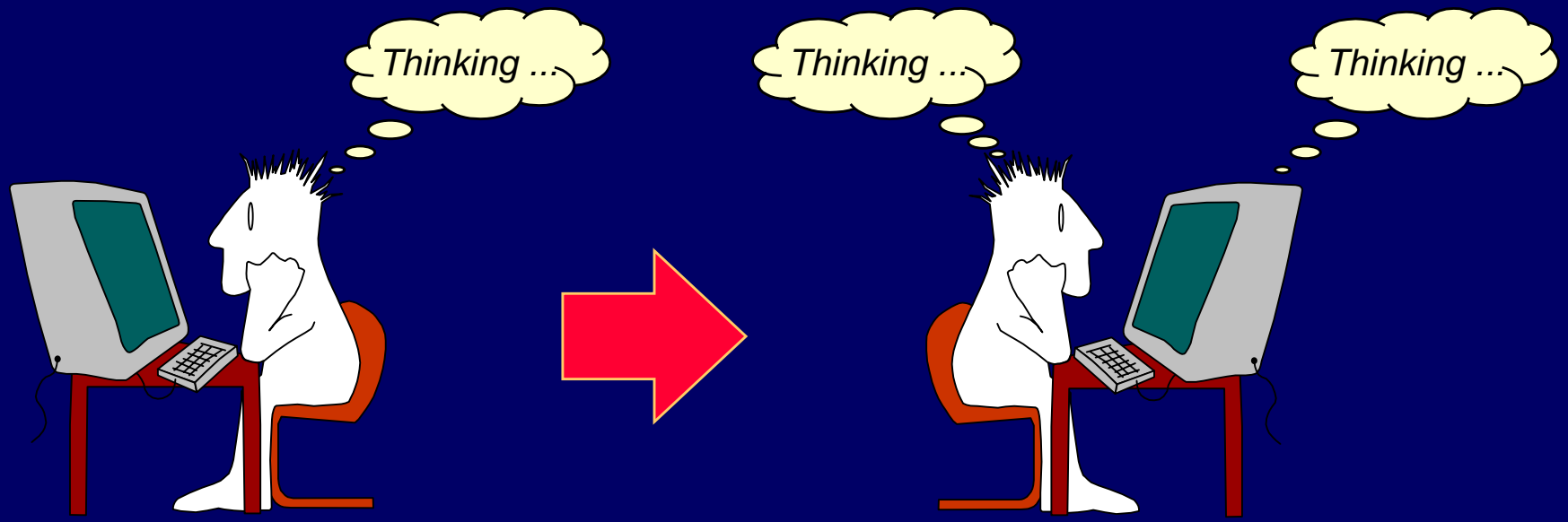
- Laboratory
- Business
- Nomadic use
- Fixed-point access versus mobile access

Variety in user access

- Plethora of platforms and access terminals:
  - Desktop PC
  - Mobile devices
  - Information kiosks
  - Personal assistants
  - Wearables
  - Network attachable devices
  - etc.

HCI versus the challenges

- Adaptation
  - ✔ adaptable user interfaces
  - ✔ adaptive user interfaces



Part II: Adaptation in HCI: The challenges

Two "faces" of adaptation ...

- Adaptability
  - refers to selecting / modifying (manually) aspects of an interactive system
  - several synonyms: customisation, tailorability, flexibility
- Adaptivity
  - refers to dynamic / run-time changes in the dialog that the system initiates or suggests to the user

Taxonomies (1/7)

- Proposed taxonomy by Totterdell and Rautenbach (1990)
  - ✔ Compares AUI features with Game Theory's "prisoner dilemma" and natural selection
  - ✔ Identifies three roles in the design of adaptation
    - designer
    - user
    - system

Taxonomies (2/7)

Prisoner's Dilemma	Evolution	Features	Computer Systems
Nasty / Friendly	Natural selection	Selection by external agent	Designed systems
Player selects from range of strategies	Genetic engineering	Deferred selection	Adaptable / Tailorable
Tit for Tat	Tropism / reflexes	Apparent learning (i.e. fully determined by design)	Adaptive
Learner	Operant conditioning	Learning; varied responses selected for different situations, evaluation by trial and error	Self-regulating
Modeller	Internal evaluation	Planning; problem solving; rule mediated representation; initial evaluation internal to the system	Self-mediating
Introspector	Abstraction	Evaluating the evaluation; generalisation; meta knowledge	Self-modifying

Totterdell and Rautenbach, 1990

Taxonomies (3/7)

Totterdell and Rautenbach, 1990

	Design Facets		
Level of system	***Variation***	***Selection***	***Testing***
Designed	*Designer*	*Designer*	*Designer*
Adaptable / tailorable	*Designer*	*User*	*Designer*
Adaptive	*Designer*	*System*	*Designer*
Self-regulating	*Designer*	*System*	*System*
Self-mediating	*Designer*	*System*	*System*
Self-modifying	*System*	*System*	*System*

Taxonomies (4/7)

- Proposed taxonomy by Kuhme, Dieterich, Malinowski, and Schneider-Hufschmidt (1992)
  - ✔ Based on four "phases of adaptation"
  - ✔ Identifies two roles for the execution of the phases
    - user
    - system

Taxonomies (5/7)

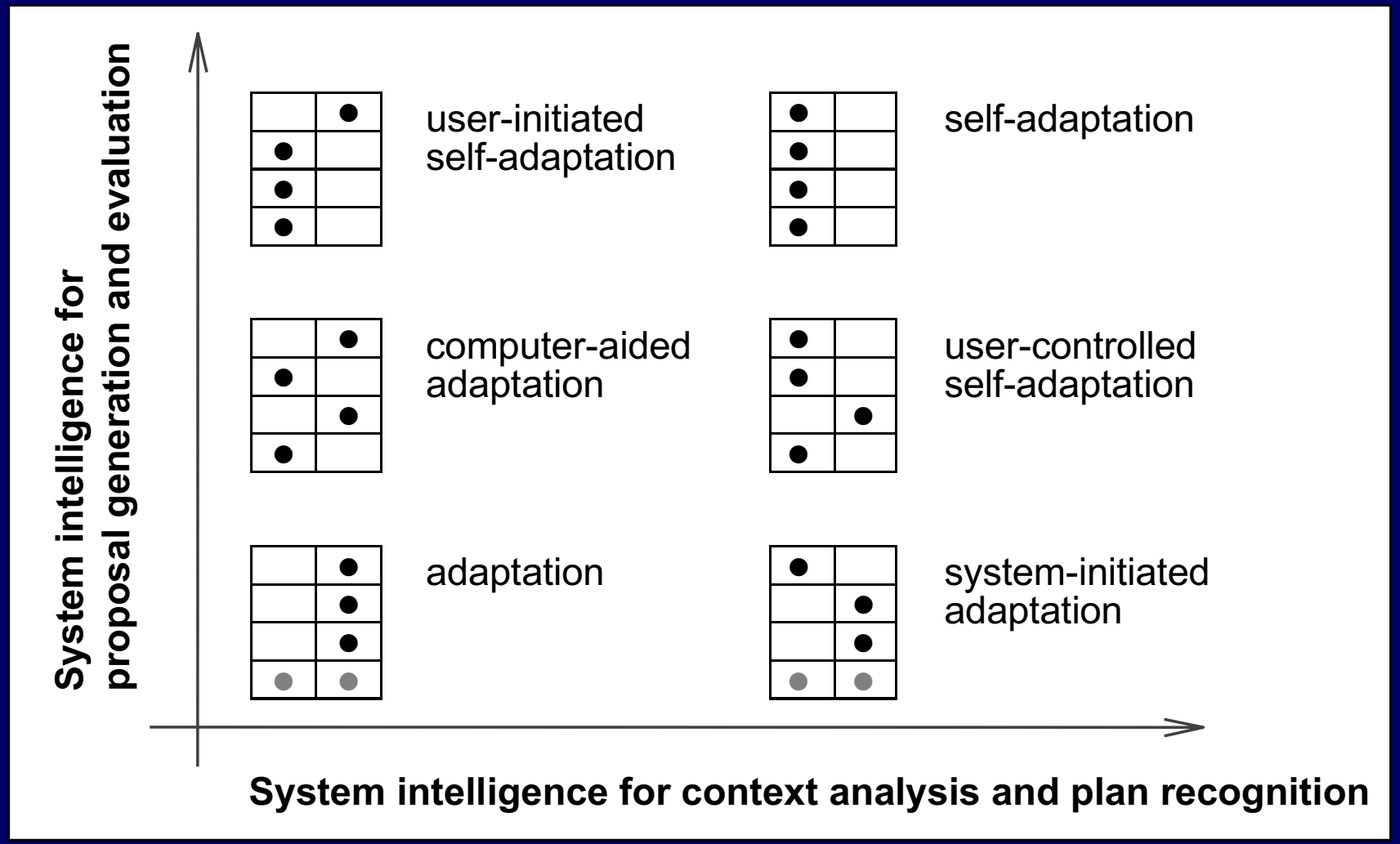
- Phase-based classification matrix

Kuhme, Dieterich, Malinowski, and Schneider-Hufschmidt, 1992

	System	User	
Initiative	●		System initiates adaptation
Proposal	●		System proposes some change / alternatives
Decision		●	User decides upon action to be taken
Execution	●		System executes user's choice

Taxonomies (6/7)

Dieterich, Malinowski, Kuhme and Schneider-Hufschmidt, 1993



Taxonomies (7/7)

- Other dimensions of adaptation can also be used for classification
  - adapted constituents (level of interaction, type of constituent, etc.)
  - method of adaptation (en-/disabling, switching, etc.)
  - considered information (user, context, application domain, etc.)
  - timing of adaptation (e.g., before or during interactive session)
  - embedded or utilised models (e.g., user model, task model, dialogue model, application model)

Adaptation constituents

- Physical aspects of interaction
  - size, colour combinations, font family and size, temporal and spatial arrangement, volume, pitch, grouping, etc.
  - input/output devices
- Interaction syntax
  - confirmation, object-function vs function-object, multiple levels of help / guidance, etc.
- Interaction semantics
  - metaphor, etc.

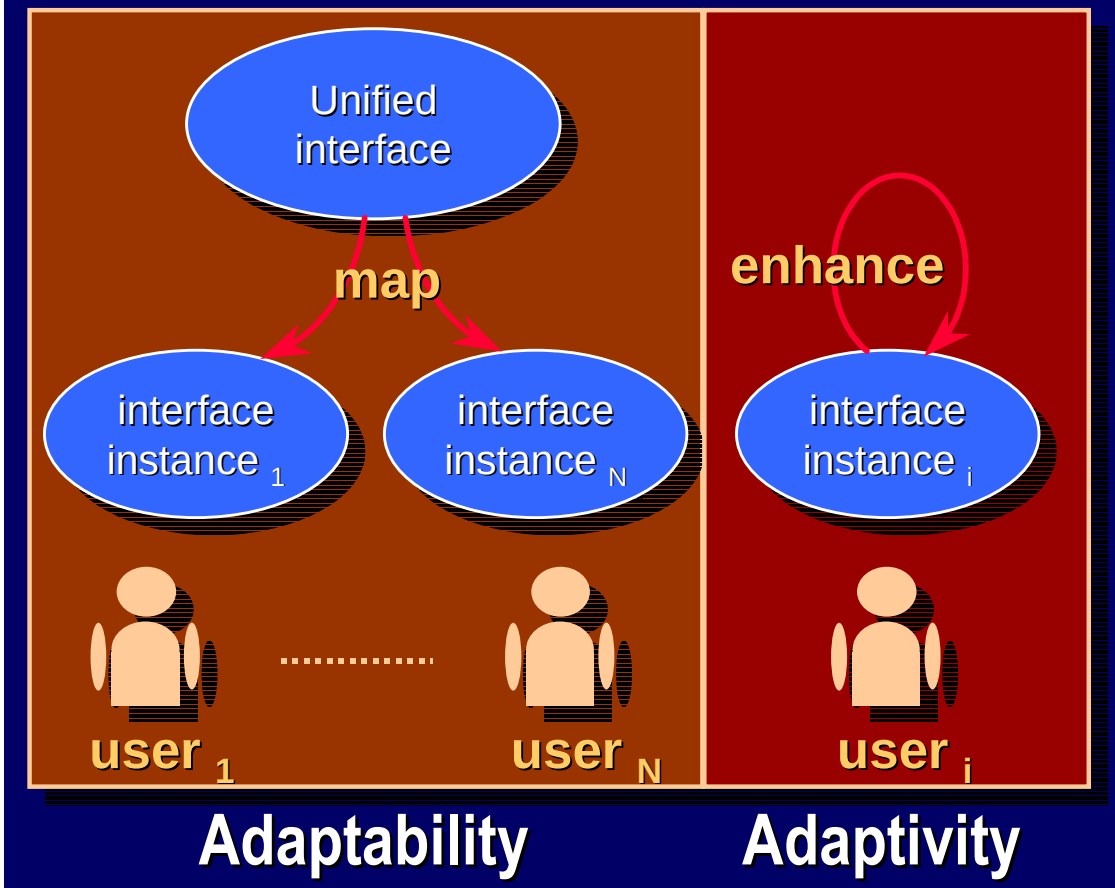
Focus on three dimensions

1. **Timing** of adaptation

2. **Level of interaction** at which adaptation is applied

3. **Information considered** in deciding upon an adaptation

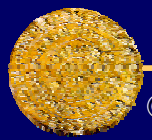
Adaptation "timing"



- adaptability
  - ✔ modification of the user interface, based on "static" - known prior to the initiation of interaction- user characteristics (e.g. user abilities, preferences) and usage context (e.g. kiosk mode)
- adaptivity
  - ✔ dynamic modification of the user interface, based on interaction states detected at run-time (e.g. user is disoriented, user has high error rate)

Level of interaction

- Semantic-level adaptations
  - selection of appropriate metaphor for mapping concepts from the source to the target domain
- Syntactic-level adaptations
  - selection of appropriate (sequences) of actions / sub-tasks for the completion of a task
- Lexical-level adaptations
  - selection of presentation and interaction attributes of the interface objects



Information considered (1/2)

- Design constraints
  - ✔ Availability of I/O devices
    - for user with disabilty X, appropriate input devices, in descending order of preference, are: Y1, Y2, Y3
  - ✔ User characteristics
    - abilities, skills, requirements, preferences, expertise, cultural background, etc
  - ✔ Task requirements
    - urgency, criticality, error-proneness, sequencing, etc
  - ✔ ...

Information considered (2/2)

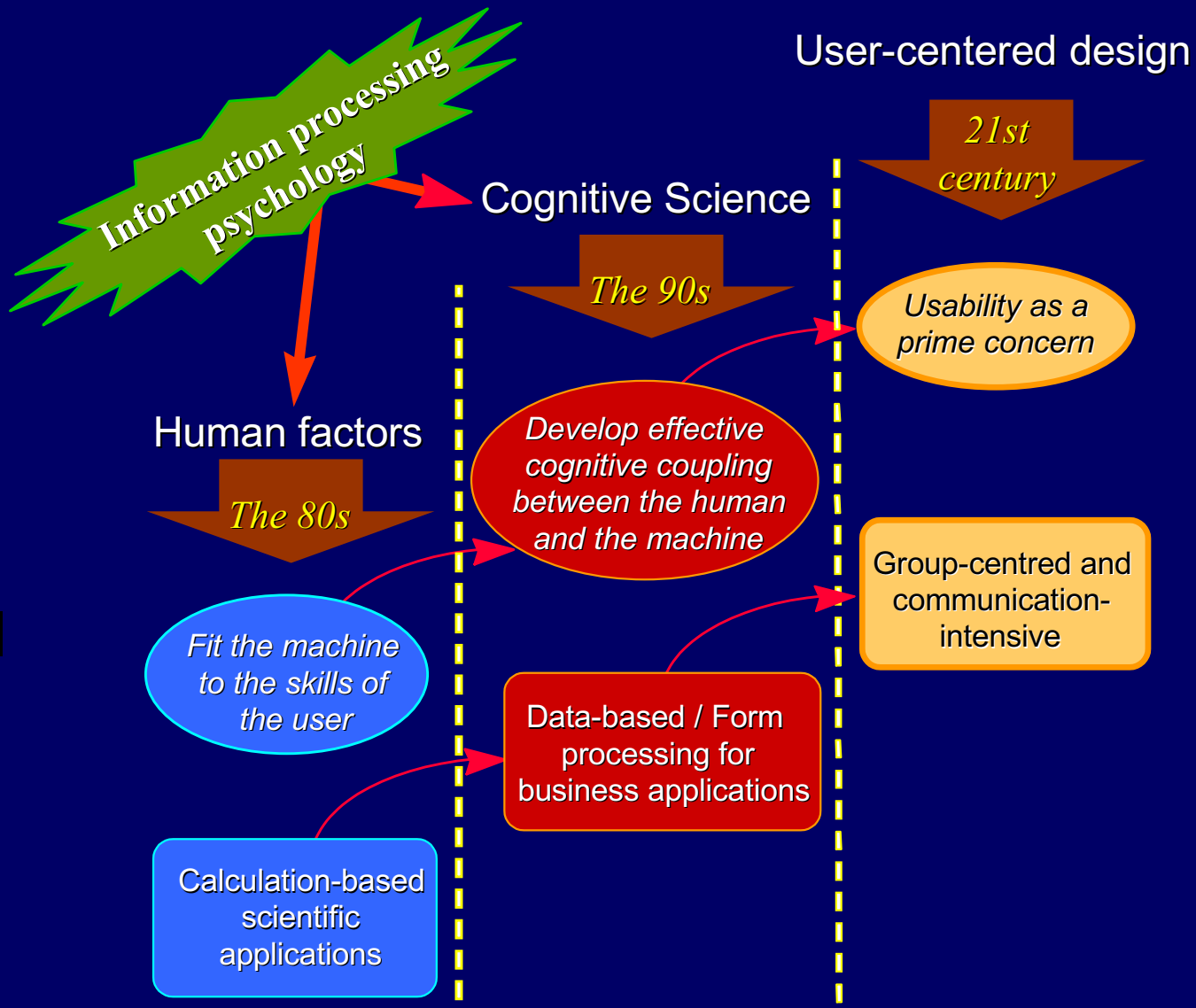
- Dynamically (during interaction) derived information
  - ✔ additional information about user characteristics (requires interaction monitoring)
  - ✔ additional information about the machine environment (requires device-level monitoring)
  - ✔ additional information about the context of use (requires environment-level monitoring)

In summary ...

- Adaptation poses two primary challenges for:
  - ✔ HCI design
  - ✔ User interface architecture

HCI design traditions

- The roots:
  - ✔ Information processing psychology
  - ✔ Developmental approaches



Information processing psychology

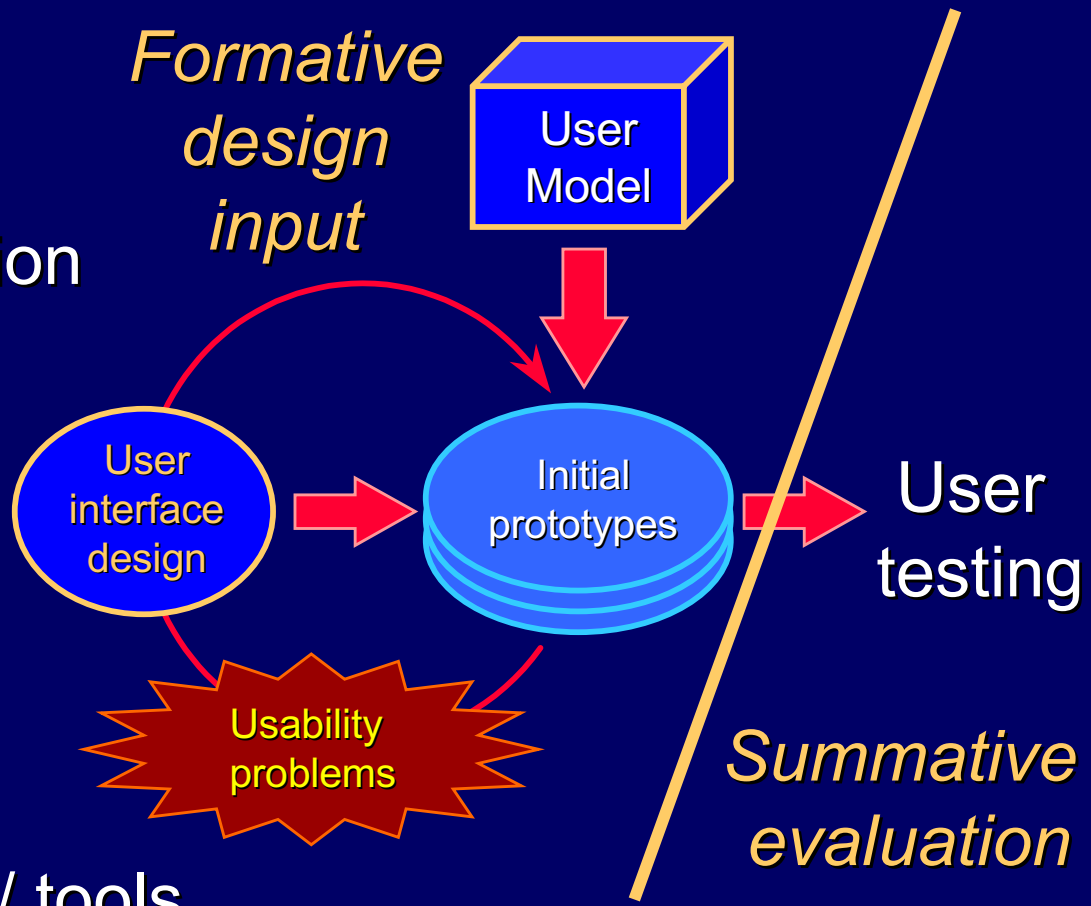
- Two prime perspectives
  - ✔ Human factors evaluation
  - ✔ Cognitive engineering

Human factors evaluation

- Reactive approach
- Laboratory based
- Costly
- Limited focus since
  - emphasis is on evaluation
  - it does not cover the whole design process
  - fails to translate results of an evaluation to design improvements or new designs

Cognitive science

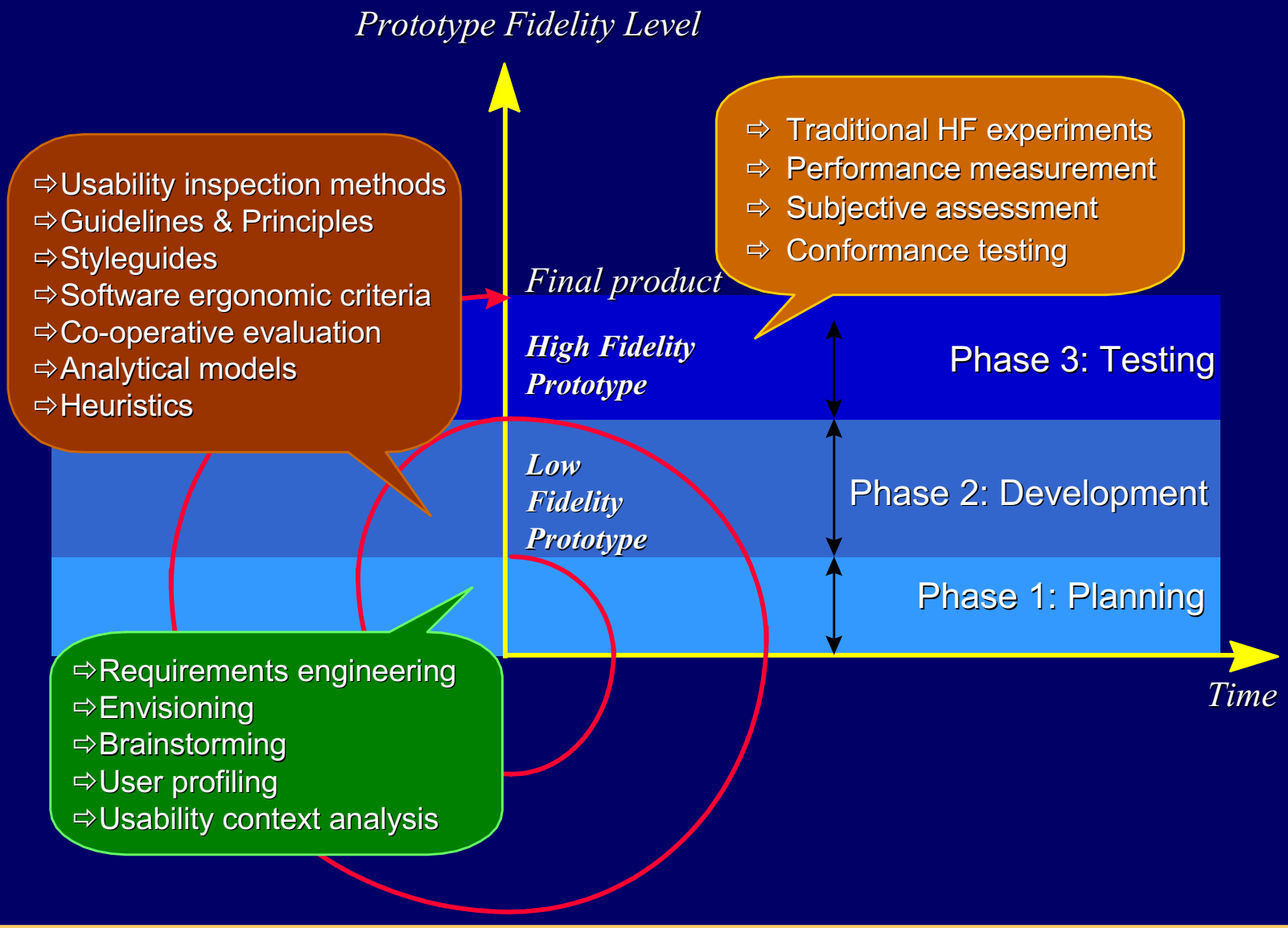
- Assumptions
  - ✔ Cognition as computation
  - ✔ Laboratory experiment
- Objective
  - ✔ Predictive theory
- Deliverables
  - ✔ Analytical HCI models / tools



Critique of Cognitive Science

- Lack of demonstrable impact
- Focus on what users should do rather than what users actually do
- Limited applicability for specialised users
- Unit of analysis does not allow contextual interpretation

User-centered design



Developmental perspectives

- Emerging perspectives
  - Activity theory (Bodker, 1989; 1991)
  - Distributed cognition (Hutchins, 1995)
  - Situated action models (Suchman, 1987)
- Common theme
  - Studying context
- Points of departure (Nardi, 1996)
  - Unit of analysis
  - Categories used for the description of context
  - Treatment of user actions

Suitability for studying adaptations

- Adaptations have a situational aspect
- Adaptations require contextual inquiries
- The development of adaptable and adaptive systems benefits from methodological frames of reference which provide the means to study and capture context

Note:

- Neither Human Factors evaluation nor cognitive science provide effective means to study context and situational aspects of interaction

User interface architectures

"User interface software architectures can be defined as structural descriptive models of Human-Computer interface, which serve as frameworks for understanding the elements of interfaces and for guiding the dialogue developer in their construction"

adapted from (Hartson et al., 1989)

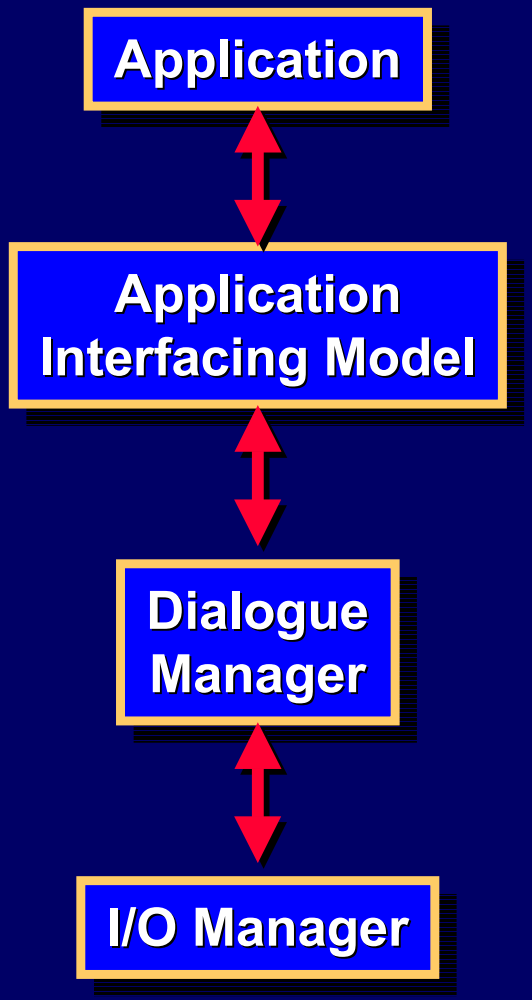
Role of UI software architecture

- To guide the development of interactive software
- To analyse and compare interactive systems
- To assess quality attributes of interactive systems

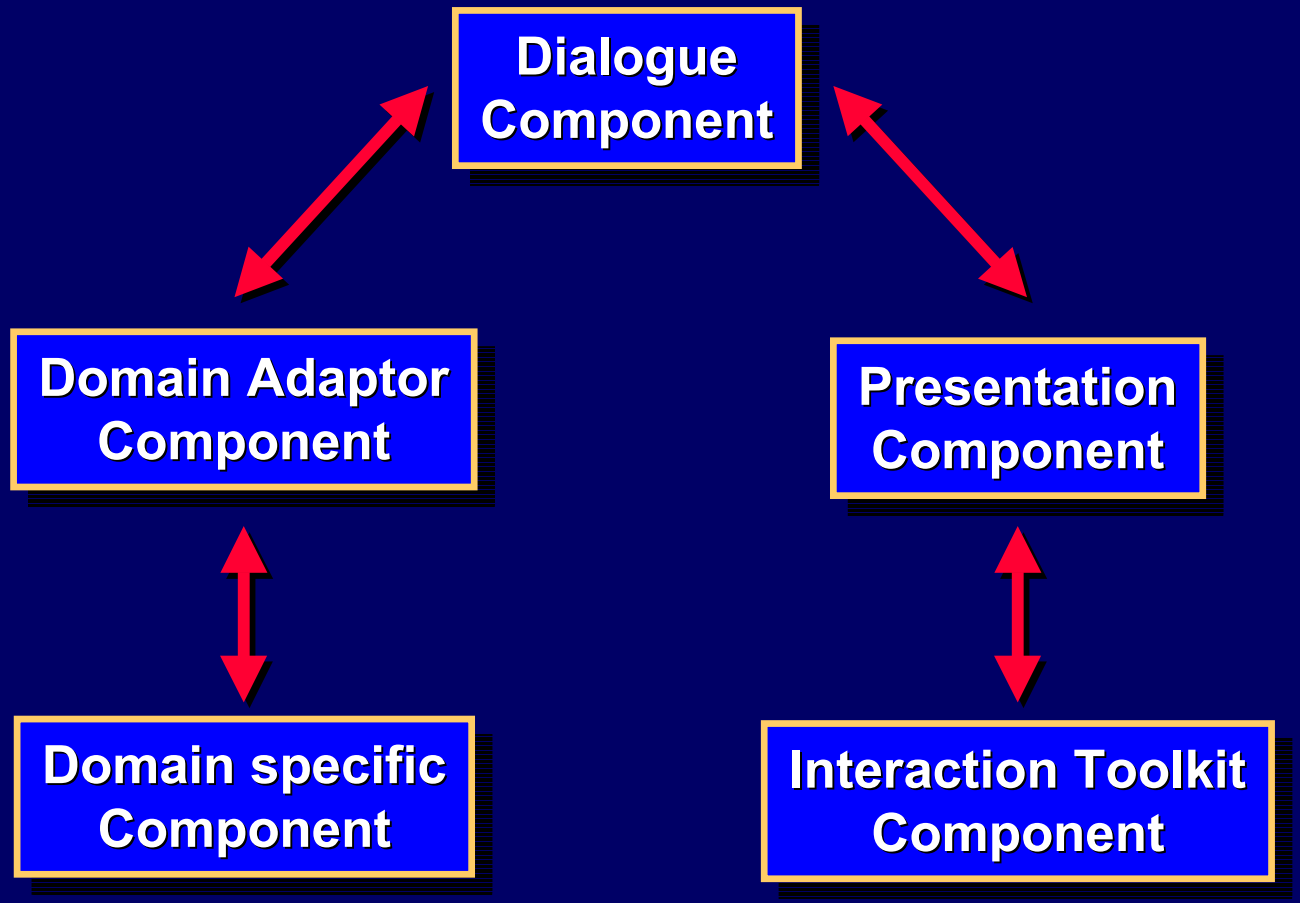
Proposals

- Traditional models
  - The Seeheim meta-model
  - The Arch meta-model
- More recent models
  - Model View Controller (MVC)
  - Presentation Abstraction Control (PAC)

The Seeheim meta-model



The Arch meta-model



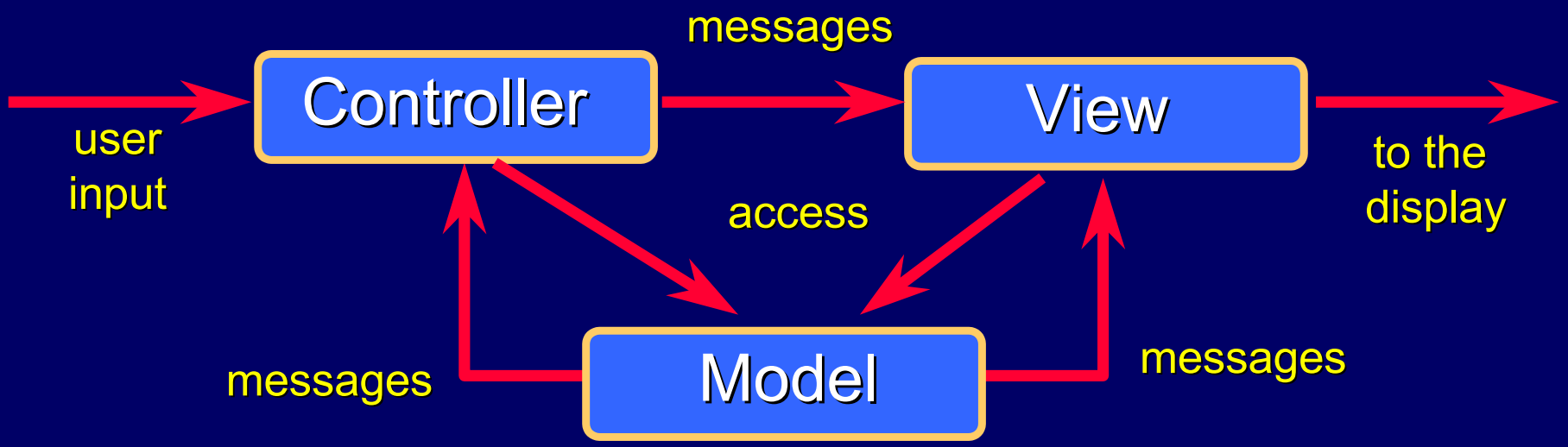
Seeheim & Arch: Common themes

- They feature the inter-layer organisation of interactive systems.
- They were defined with the aim to preserve the principle of separation between interactive and non-interactive code of computer-based applications.
- They became popular as a result of early work in UIMS.

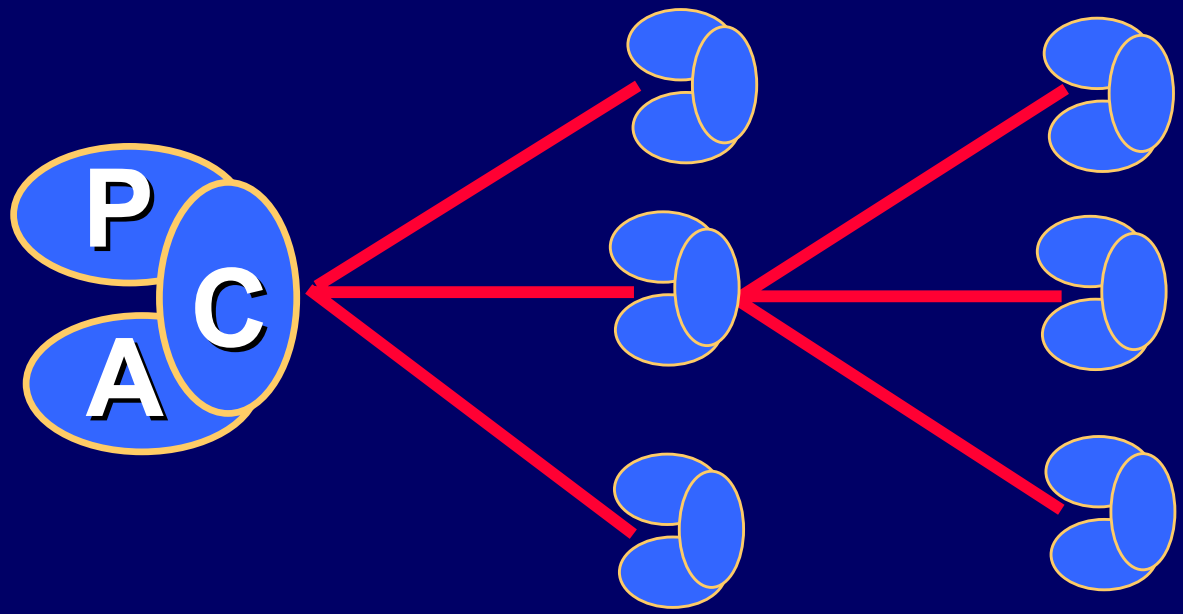
Subsequent proposals

- Model View Controller (MVC)
- Presentation Abstraction Control (PAC)

The MVC


messages
Controller
View
user
input
to the
display
access
messages
Model
messages

The PAC model



MVC & PAC: Common themes

- Both proposals link with Object-Oriented traditions
- Their focus is on intra-layer organisation of interactive software
- They have an explicit implementation flavour

Suitability for implementing adaptation

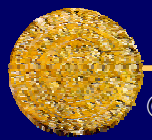
- Low-level and implementation-oriented proposals
- No insight for adaptation
- Lack of reference to the functional units and components needed to support adaptation

Coffee Break: 15.30 - 16.00

Part III: Towards meeting the challenges

Part III presentation plan

- Unified User Interface Design method
  - ✔ Process and conduct
  - ✔ Techniques
  - ✔ Outcomes
  - ✔ Examples & exercises
- The Unified User Interface architecture
  - ✔ Architectural abstraction
  - ✔ Functional modules



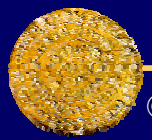
Unified Design Method

The need for unified design

- Plethora of design alternatives
  - During the design process, diverse user requirements and alternative interaction platforms lead to a wide range of plausible design options or patterns.
- Need to represent design alternatives
  - The implementation of a unified interface should:
    - include all such alternative patterns to realise automatically adapted behaviour.
    - reason about the alternatives and select the most appropriate for a particular situation of use

The problem being addressed ...

- A design process and supporting techniques, which allow the designer to:
  - ✔ collect and organise all design alternatives in a semantically meaningful manner
  - ✔ produce an outcome which forms the specification of a single interactive application



Design process

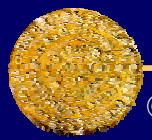
- Populate the design space (enumerating plausible design alternatives)
- Identify abstractions
- Rationalise the design alternatives

Populating the design space (1/2)

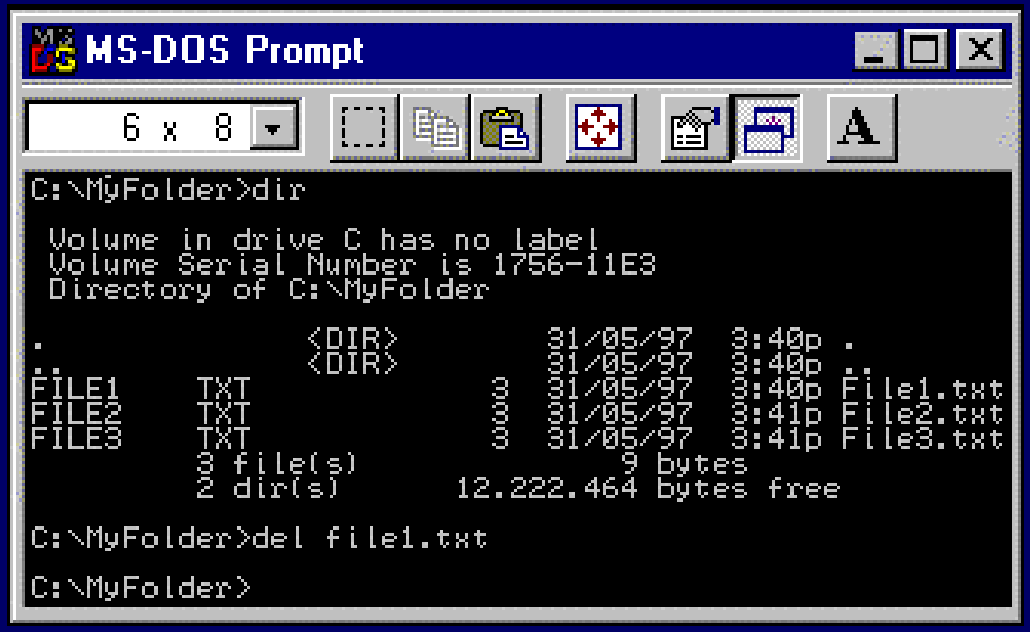
- Seeking to identify:
  - ✔ who the different users are
  - ✔ how they perform a designated set of tasks
  - ✔ what artefacts they are familiar with

Populating the design space (2/2)

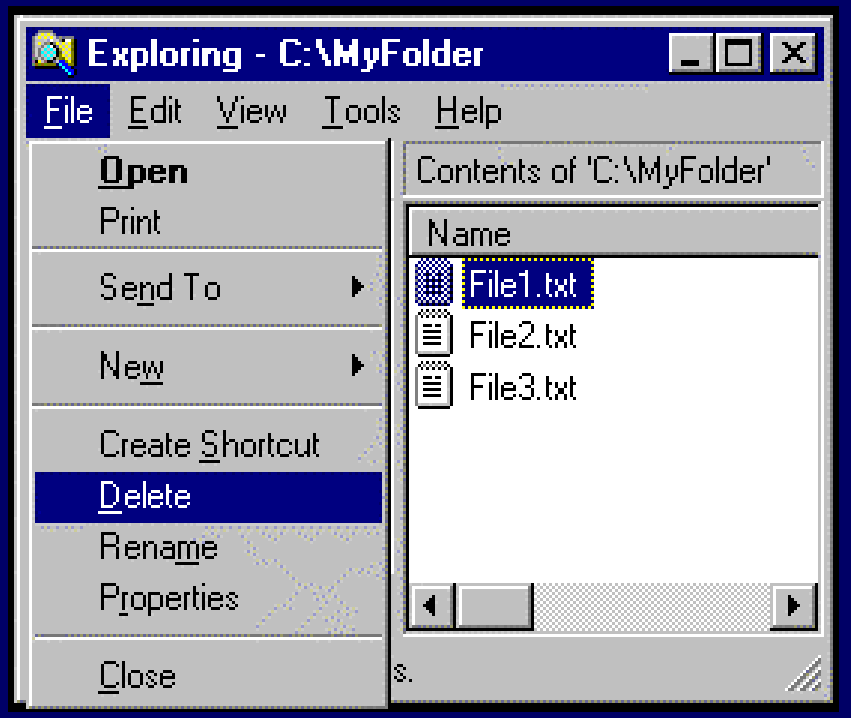
- Techniques used:
  - observation which reveals patterns and artefacts in use
  - task analysis in cases where a "system" is already available
  - envisioning and rapid prototyping to assess with users likely options
  - other formative techniques which reveal artefacts and patterns of use



Enumerating visual alternatives (1/2)

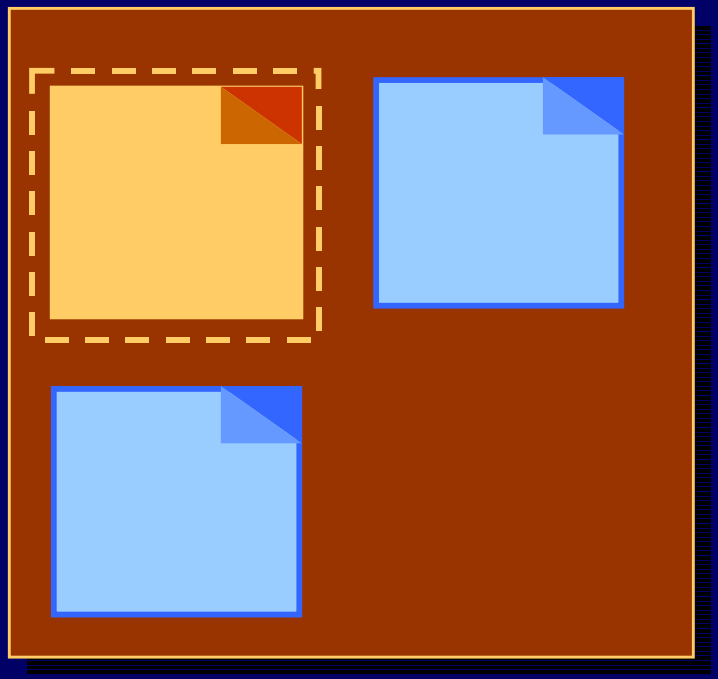


Deleting a file in Function-Object syntax

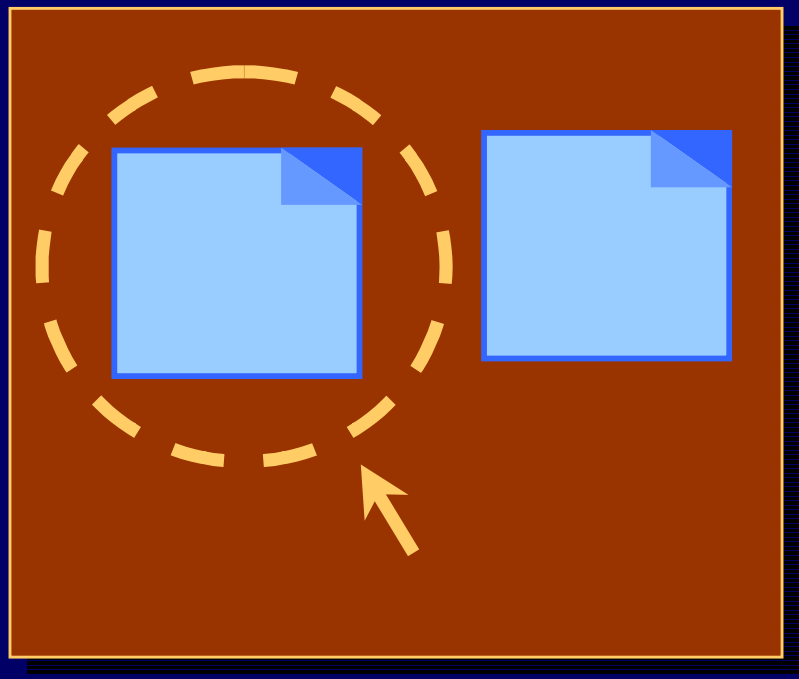


Deleting a file in Object-Function syntax

Enumerating visual alternatives (2/2)

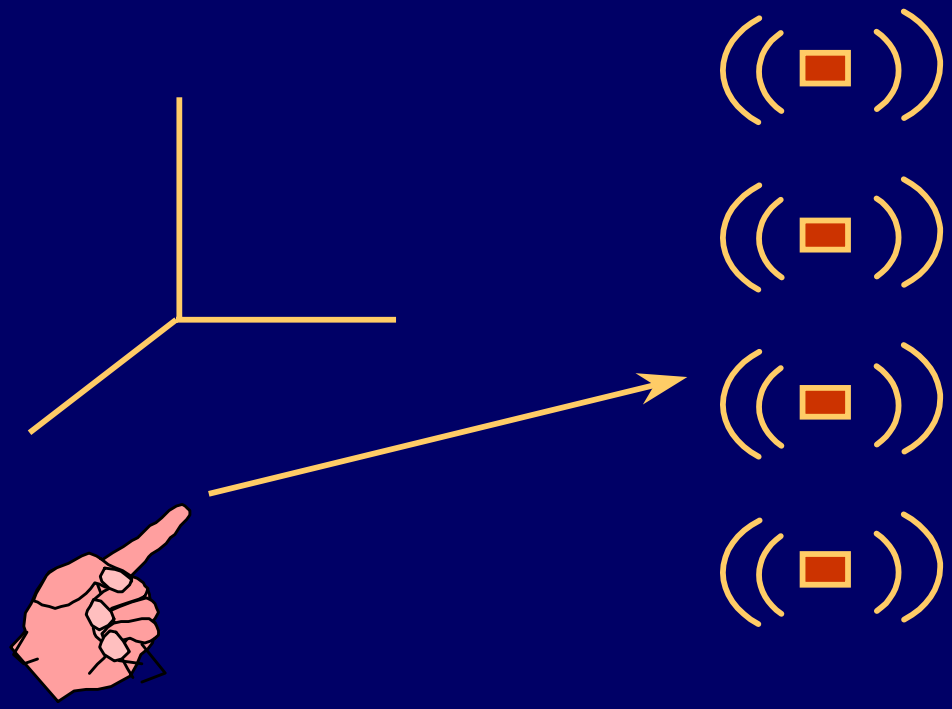


Deleting through scanning options in a selection set

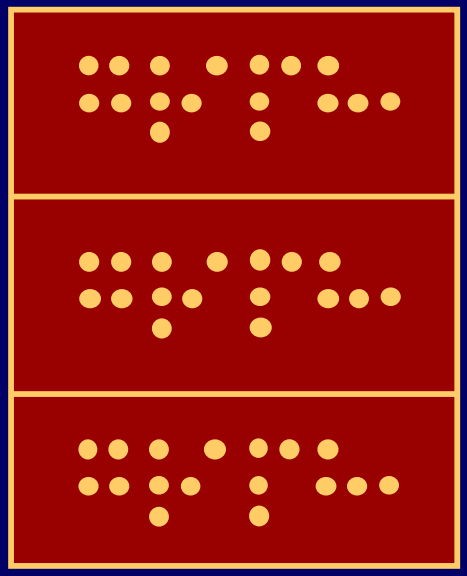


Deleting through marking the target in the selection set

Enumerating non-visual alternatives



Deleting a file using a direct manipulation auditory interface

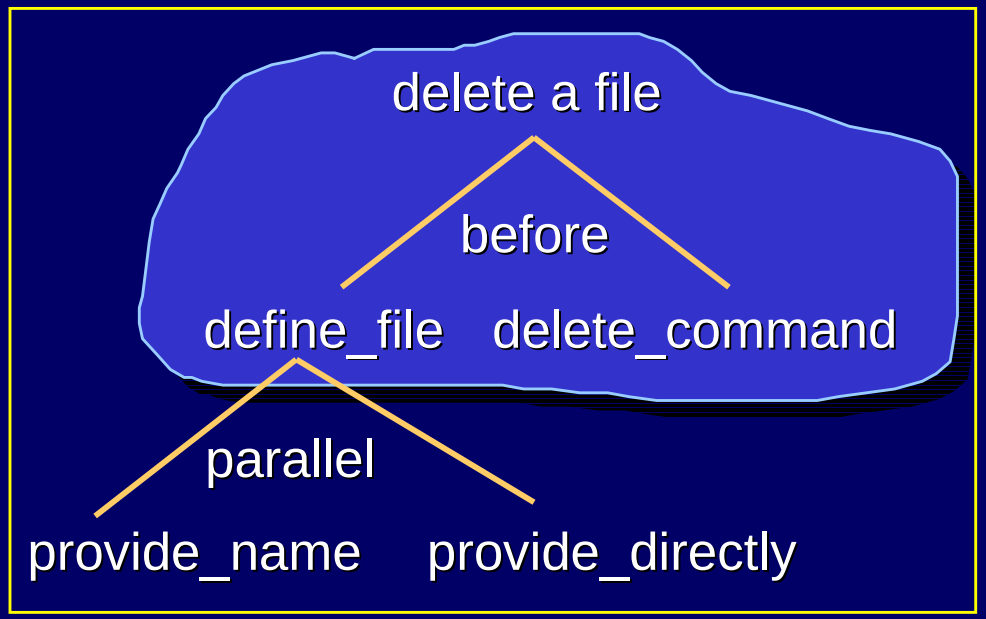


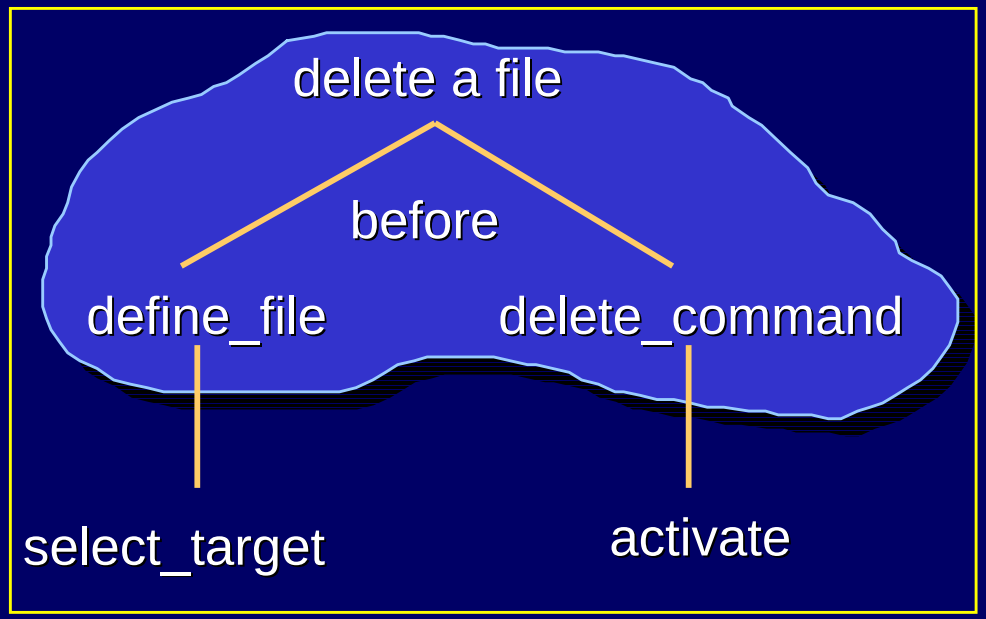
Deleting a file using a tactile listbox

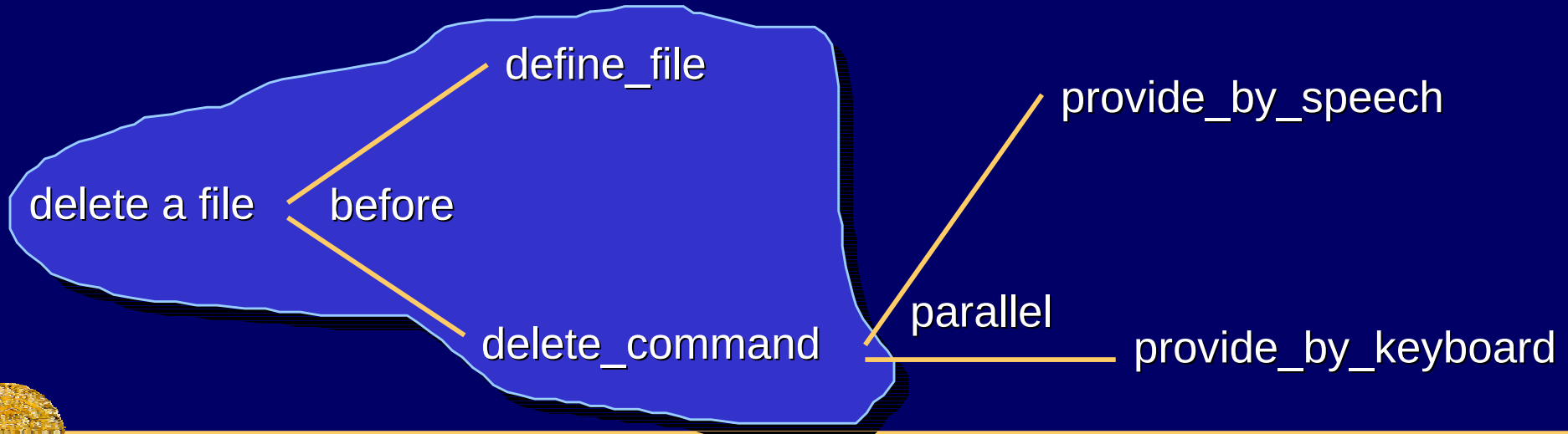
Identifying abstractions

- Seeking to group concrete artefacts to higher level abstractions
- The resulting abstractions should be:
  - ✔ metaphor independent
  - ✔ platform independent
  - ✔ user independent

For example ...


delete a file
before
define_file
delete_command
parallel
provide_name
provide_directly

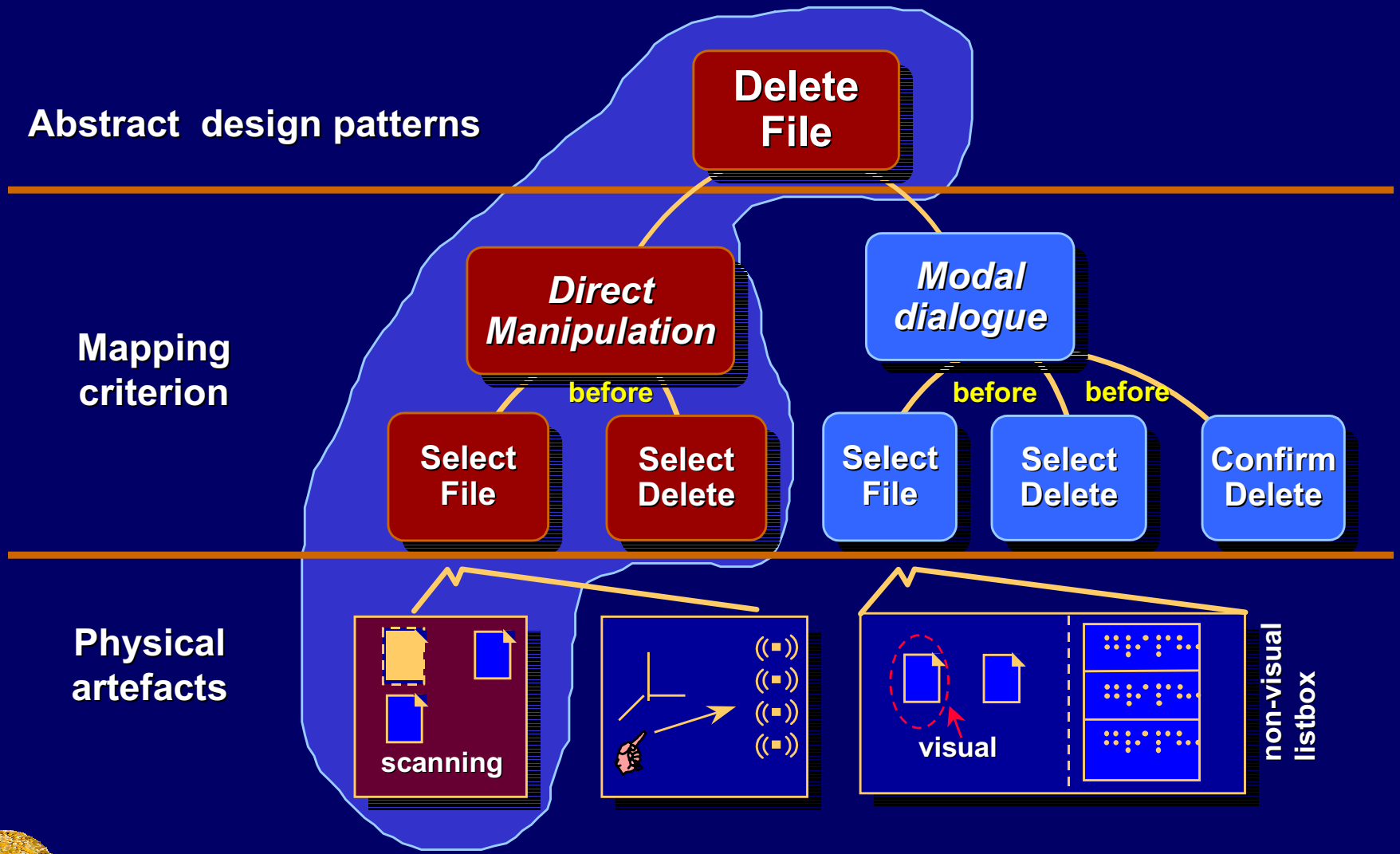

delete a file
before
define_file
delete_command
select_target
activate


define_file
delete a file
before
delete_command
provide_by_speech
parallel
provide_by_keyboard

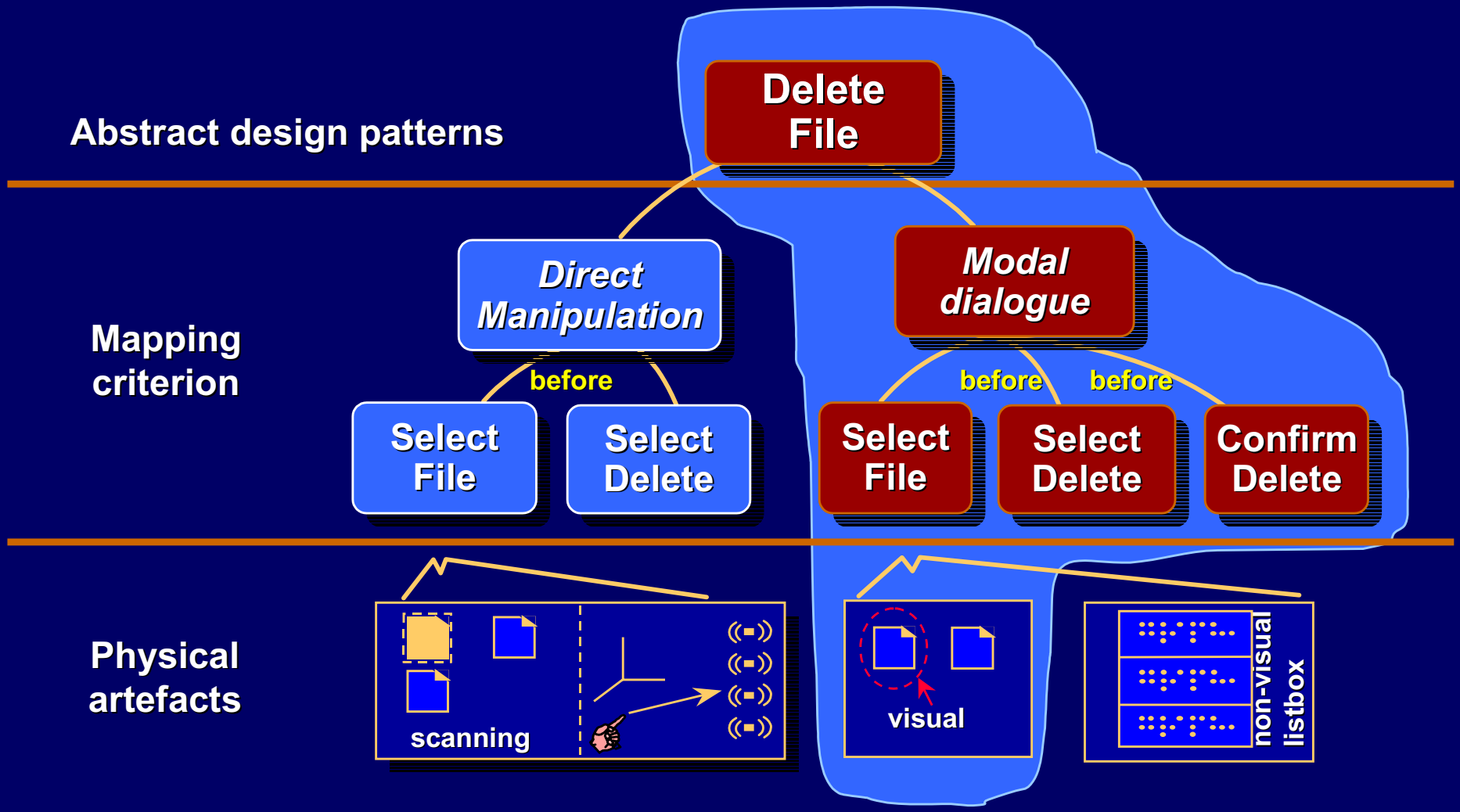
Rationalising the design space

- Find out the design logic or evidence which can guide unambiguously the mapping of an abstract component to different concrete alternatives
- Repetitive process which requires:
  - ✔ early prototyping for users to gain hands-on experience
  - ✔ formative evaluation (e.g. comparative assessment, user consultation, etc) to obtain the evidence

For example ...


Abstract design patterns
Mapping criterion
Physical artefacts
Delete File
Direct Manipulation
Modal dialogue
before
before
before
Select File
Select Delete
Select File
Select Delete
Confirm Delete
scanning
visual
non-visual listbox

Or, alternatively ...



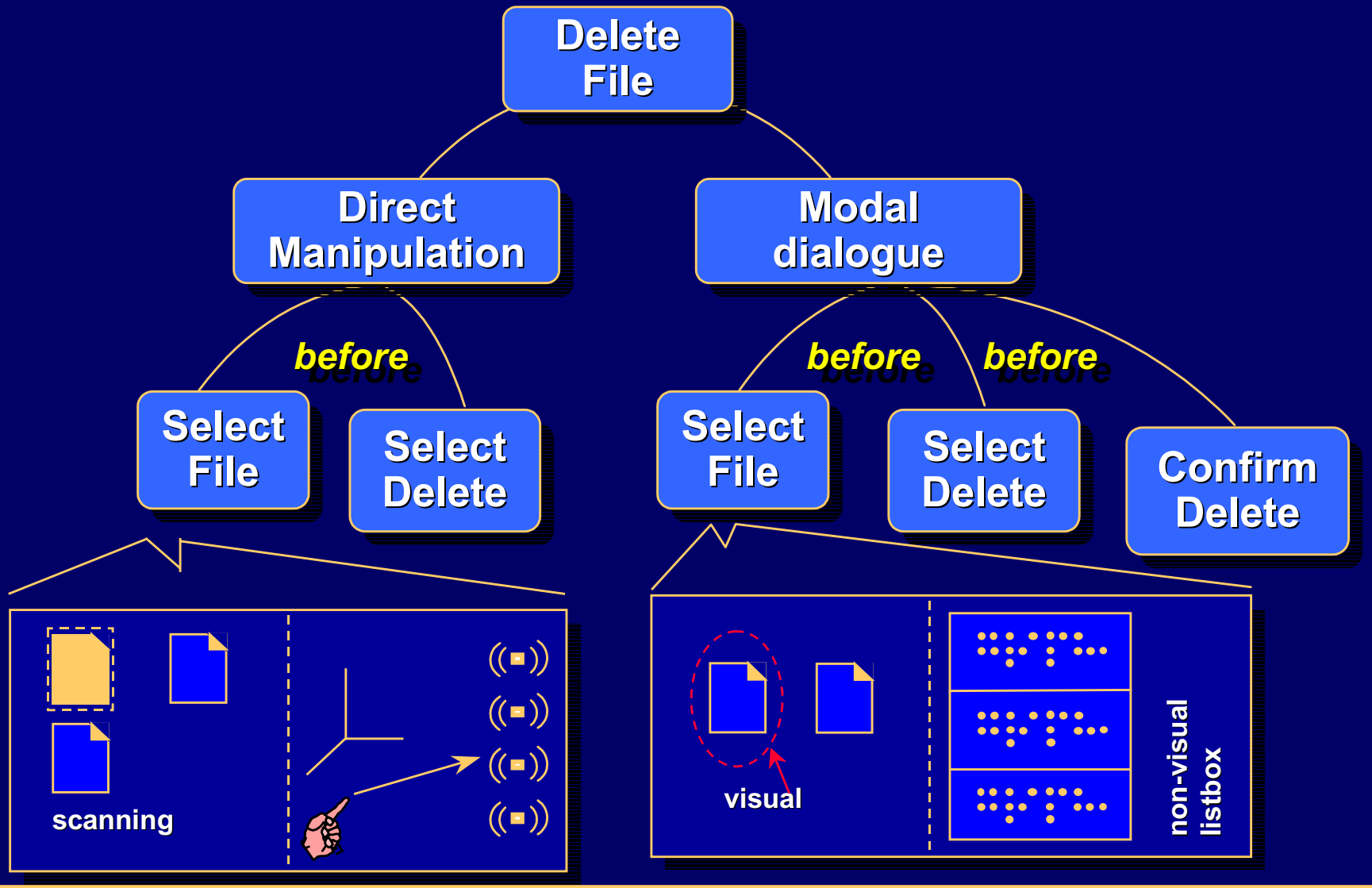
Unified Design outcomes

- Polymorphic task hierarchy
- Elements of the design space
  - ✔ Design alternatives
  - ✔ Design criteria
  - ✔ Design argumentation

Polymorphic task hierarchies - properties

- Hierarchical organisation
- Polymorphism
- Differentiation parameters (user and context)
- Task operators (not fixed)
  - *before* (sequencing)
  - *or* (parallelism)
  - *xor* (exclusive repetition)
  - * (single repetition)
  - + (absolute repetition)
  - *designers may employ more operators*

Polymorphic task hierarchies - example



The notion of style in polymorphic task hierarchies

- Each alternative decomposition is called a decomposition style, shortly a style.
- At any decomposition level, a style can be analysed following any other design method, apart from the unified design technique, such as:
  - ✔ Graphical / physical design
  - ✔ GOMS analysis
  - ✔ Traditional HTA
  - ✔ Event / response diagrams
  - ✔ UAN
  - ✔ State-based diagrams
  - ✔ Formal notations

Categories of design artefacts in polymorphic task decomposition

- User tasks
  - What the user has to do; user tasks are the centre of polymorphic decomposition
- System tasks
  - What the system has to do / respond (e.g. feedback); they are an integral part of the task decomposition
- Physical structure
  - Interface (sub-)components on which user actions are to be performed; always associated to user- or system- tasks

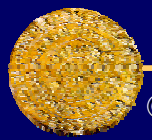
➡ Polymorphism can be applied, on the basis of user- and context-attributes, for any of the above categories of design artefacts.

Unified Design Rationale

- Two main objectives:
  - ✔ Seeking to find the reasoning behind alternative design options
  - ✔ Seeking to identify criteria for ranking design alternatives
  - ✔ Seeking to optimise the design space

Typical user-oriented questions driving the inquiry

- At the level of the target user group(s)
  - What is the user's preferred way of performing a particular task?
  - What is the user' s preferred style of interaction?
  - Which artefacts is the user familiar with?
  - What is the user' s reliable control act and contact site?
  - Does the user possess certain physical capabilities such as:
    - Ability to pull and push targets
    - Ability to repeat an action on demand
    - Ability to drag and drop objects
    - etc

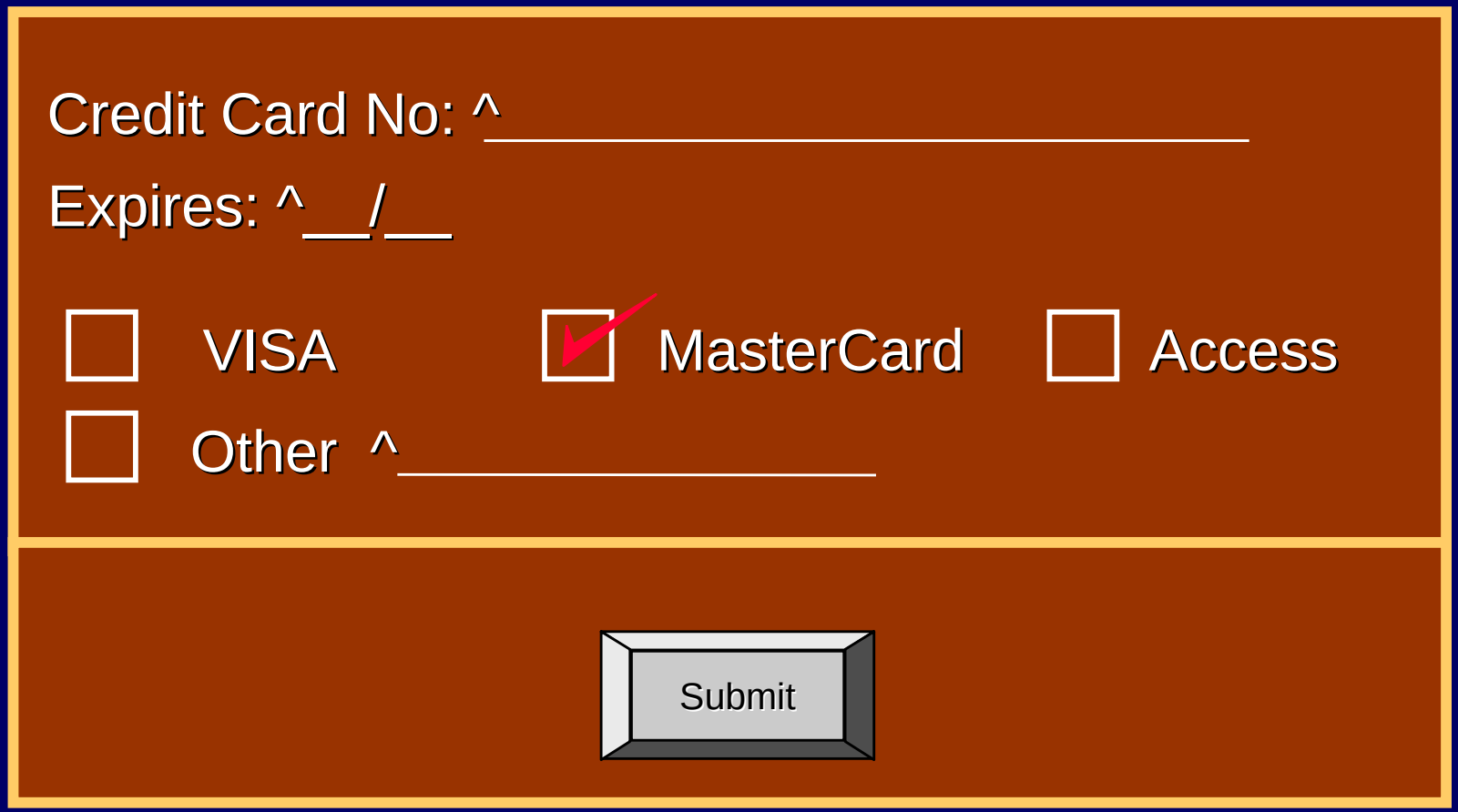


Typical platform oriented questions driving the inquiry

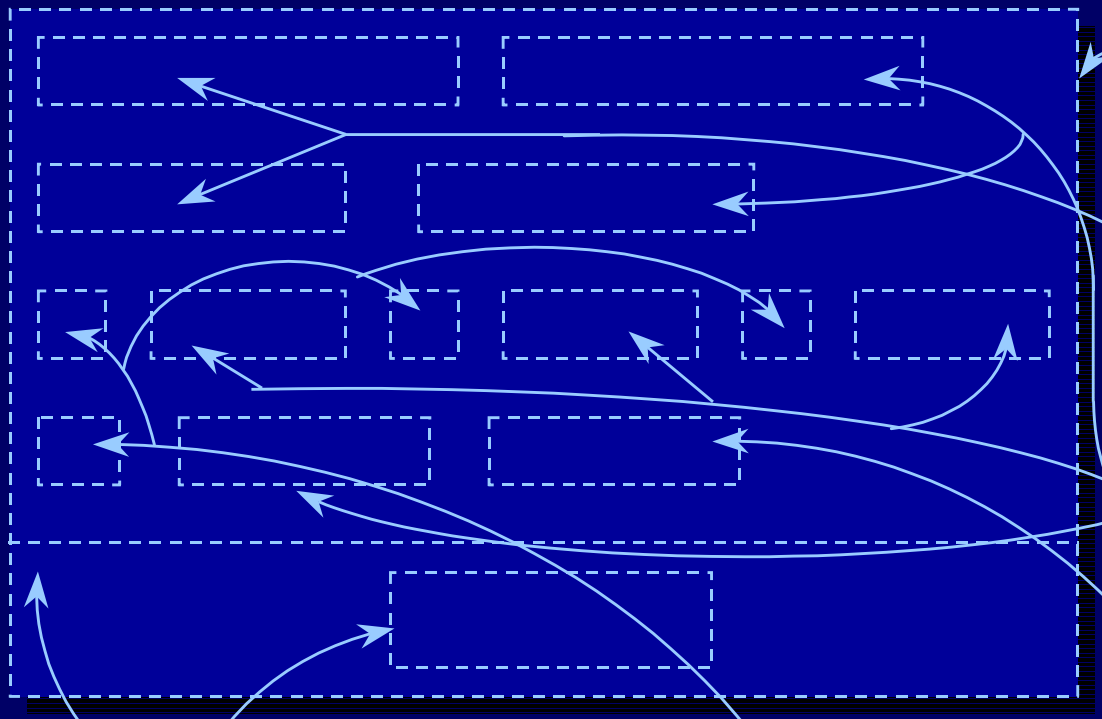
- At the level of the overall computational embodiment
  - Does the overall interaction metaphor fit the users and the tasks they are to accomplish with the interface?
  - Are there any equivalent computational embodiments, and under what conditions can they be selected?
  - Does the underlying platform support the required range of interactive behaviours?
  - etc

Unified design in practice

Reengineering an artefact



Identifying roles and characterising objects of the physical artefact



logical grouping (abstract syntactic)

label associated to domain object (abstract syntactic)

number (abstract syntactic)

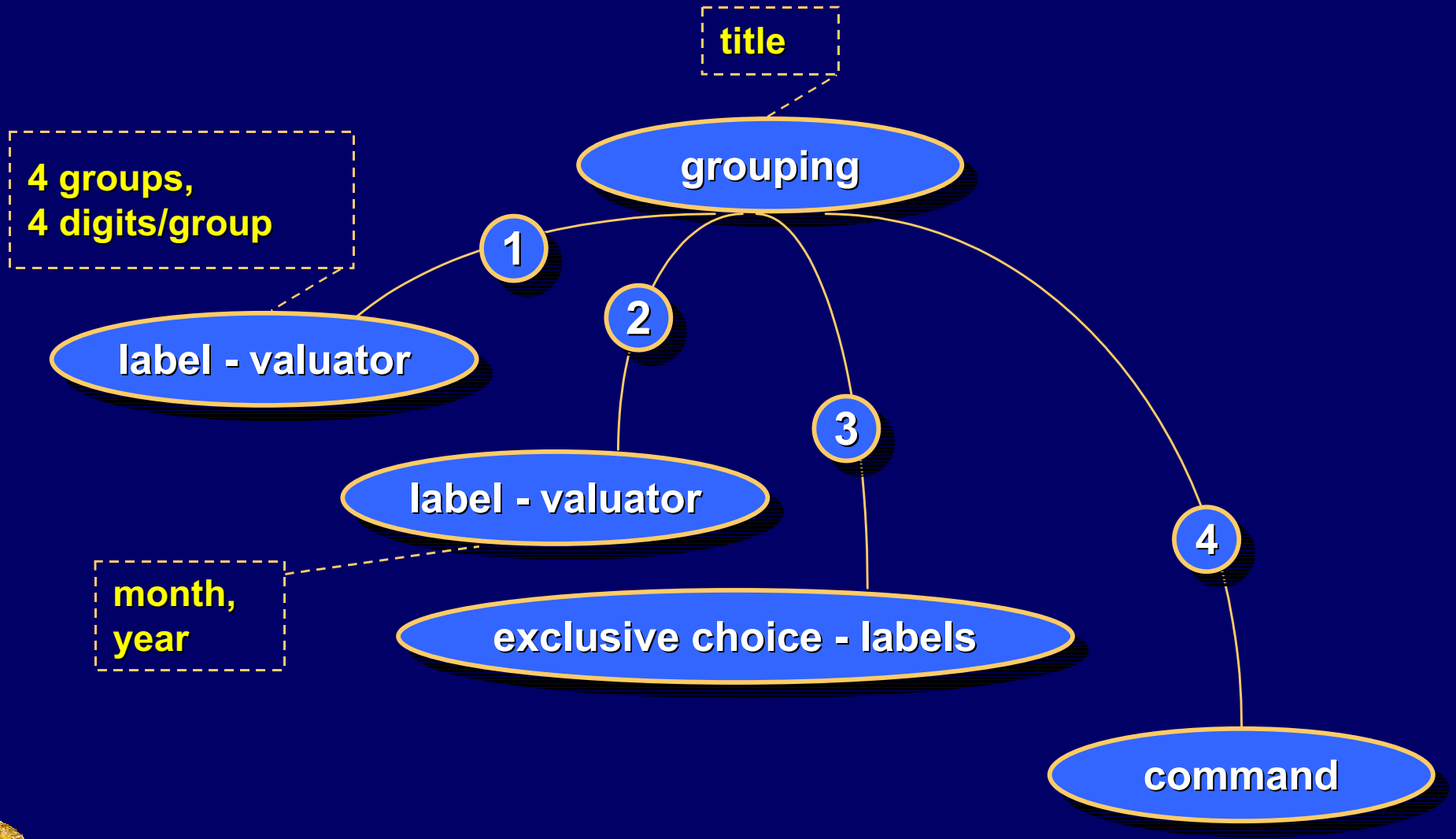
name (abstract semantic)

check-boxes (exclusive choice - abstract semantic)

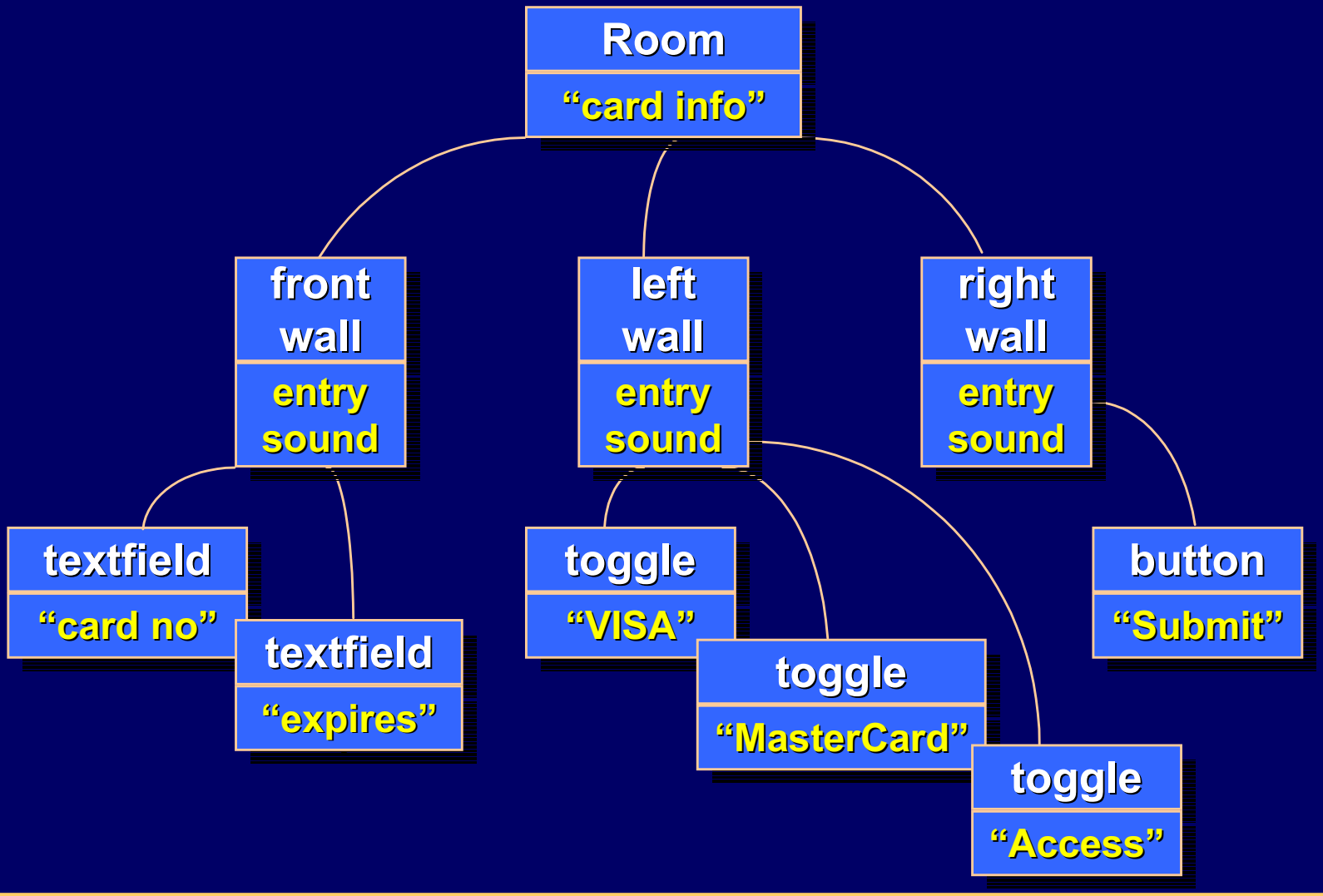
group separator (generalised syntactic)

command (abstract syntactic)

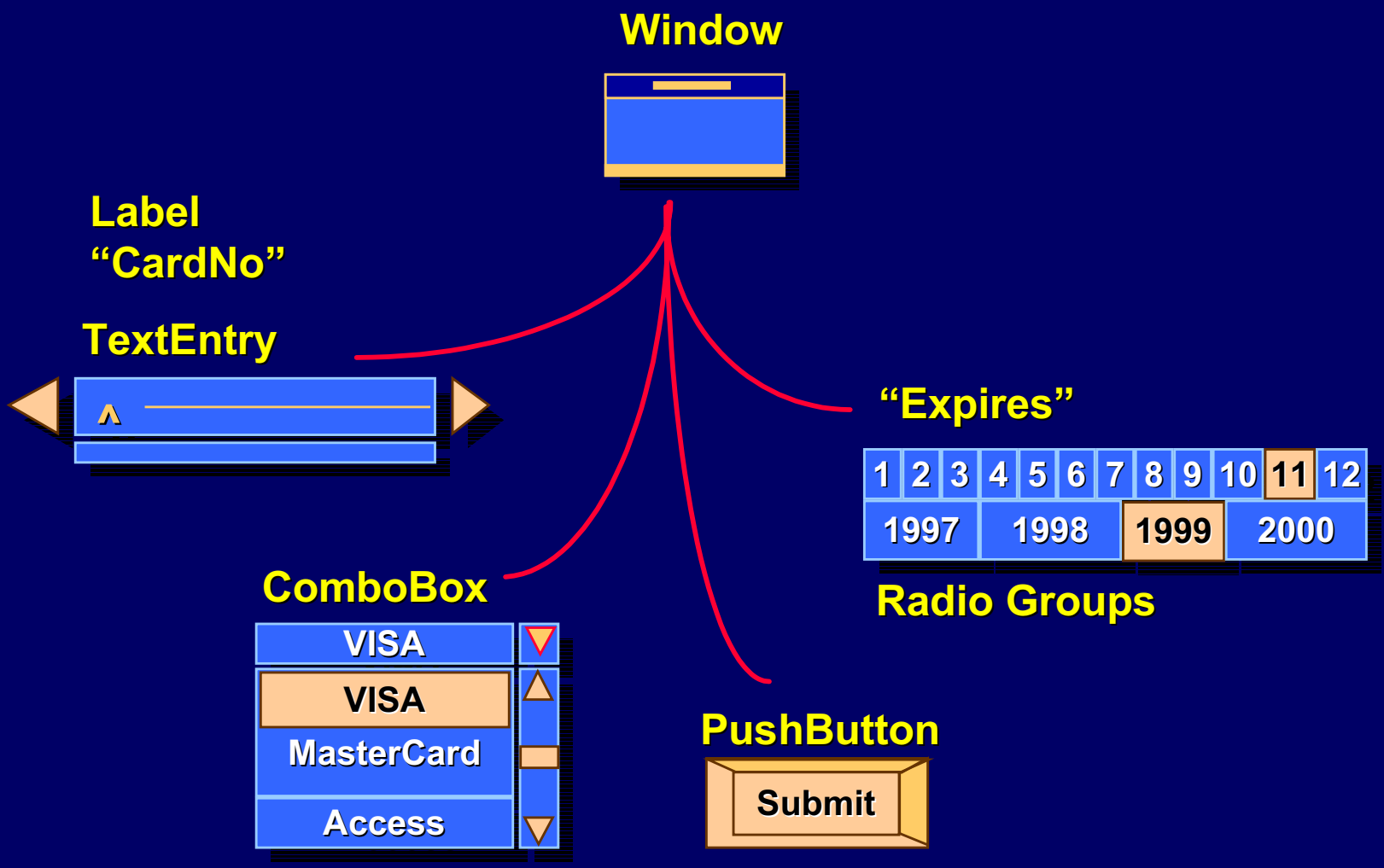
Resulting higher-level object model



Designing an alternative physical scenario - non-visual interaction



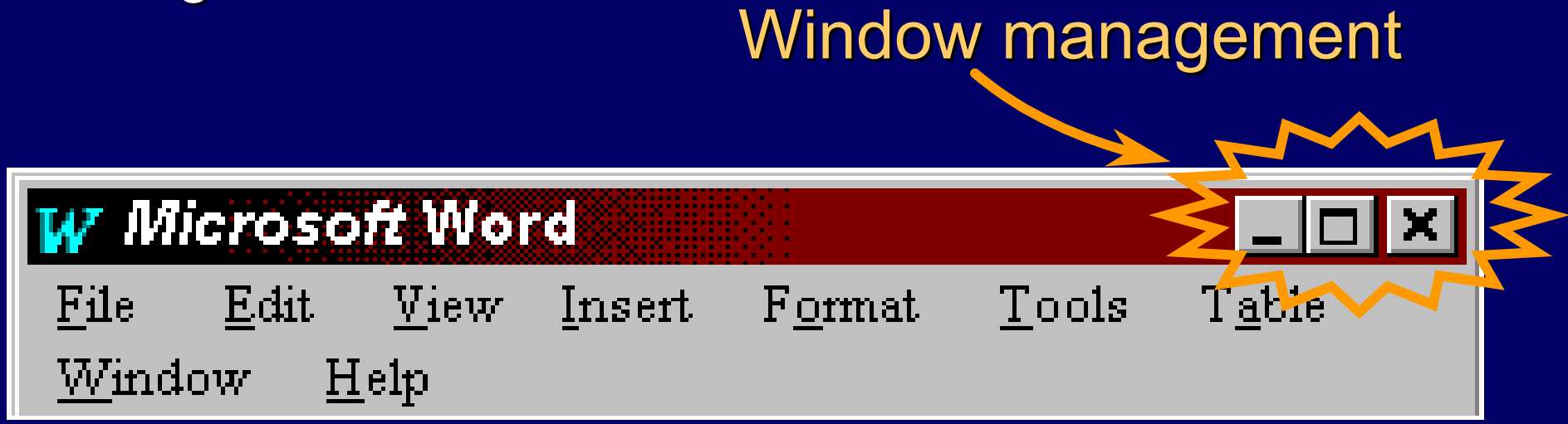
Designing an alternative physical scenario - windowing interaction



Exercises on Unified Design

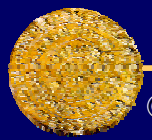
Example 1: Design question

- Given a motor-impaired user who can access a windowing application through switches and scanning, identify alternative artefacts for performing window management functions

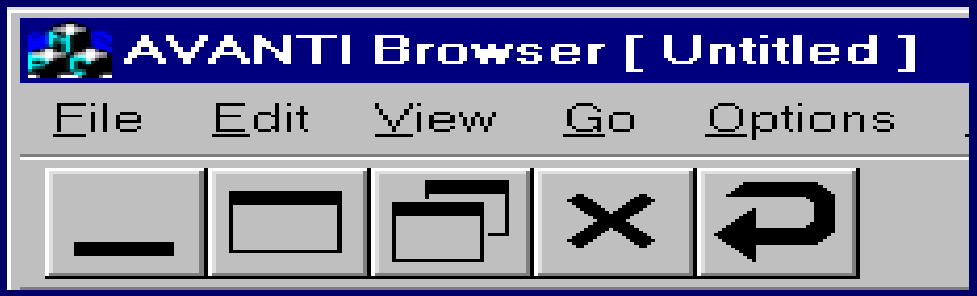


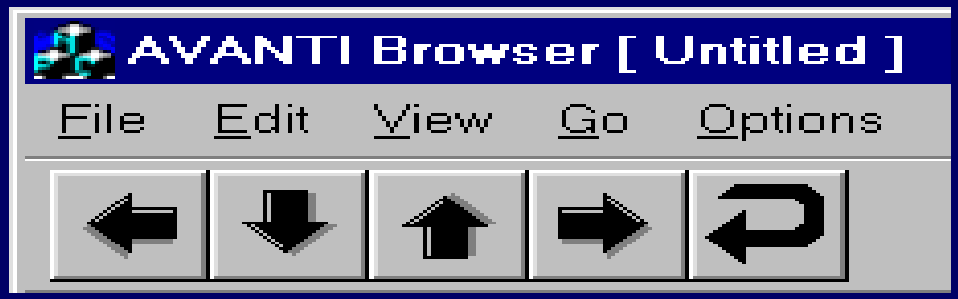
A possible scenario

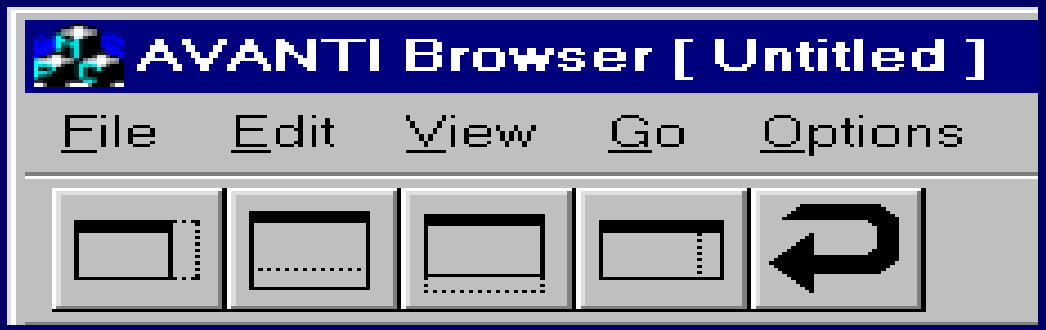
- Group window management functions in logical groups
- Identify a suitable container object class for each logical function group
- Provide artefacts that allow the user to carry out window management



Realising the scenario



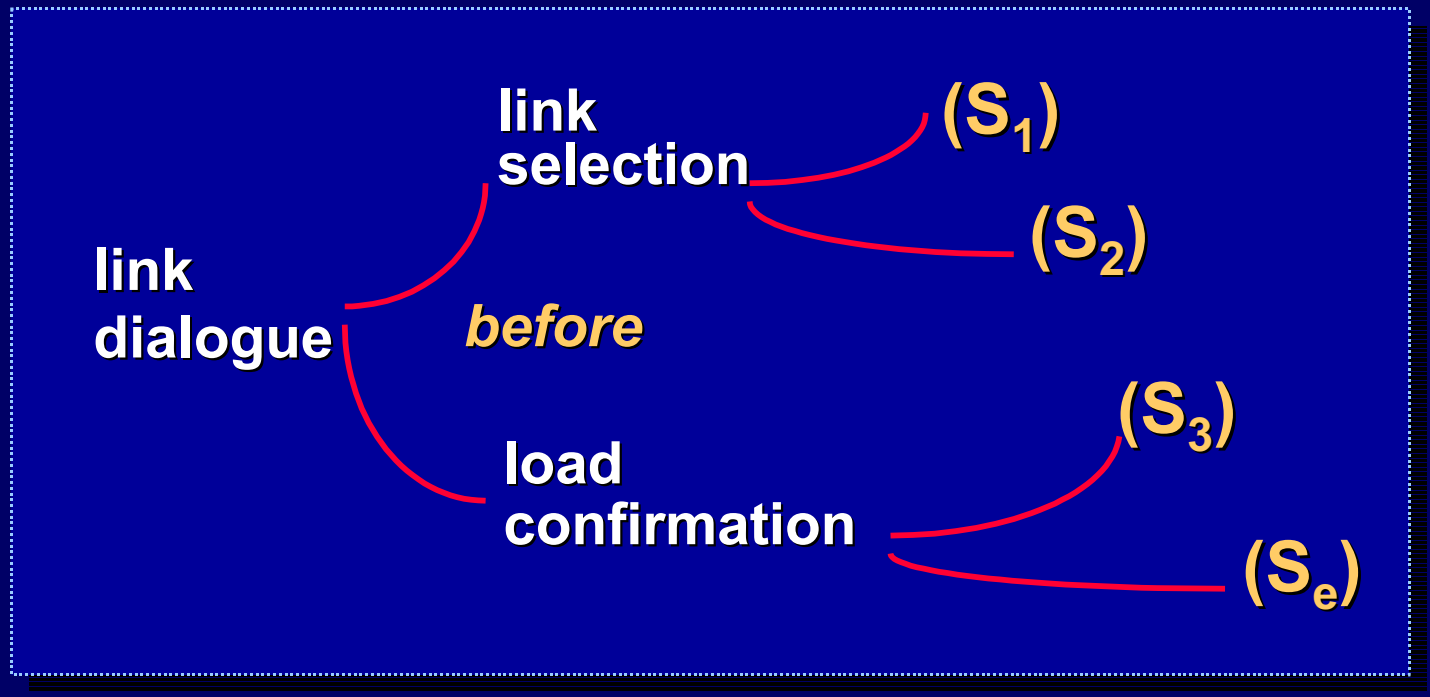




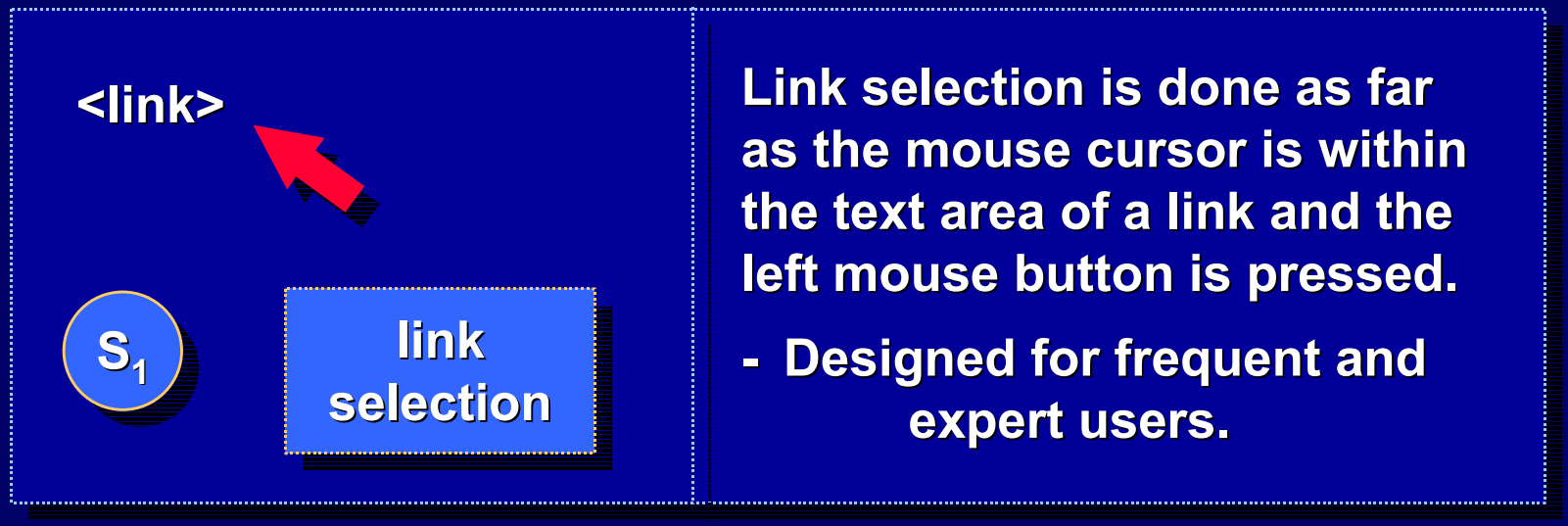
Example 2: Design question

- Identify plausible alternatives (e.g. polymorphic decomposition) for performing link selection in a conventional Web browser

Polymorphism on the link selection task



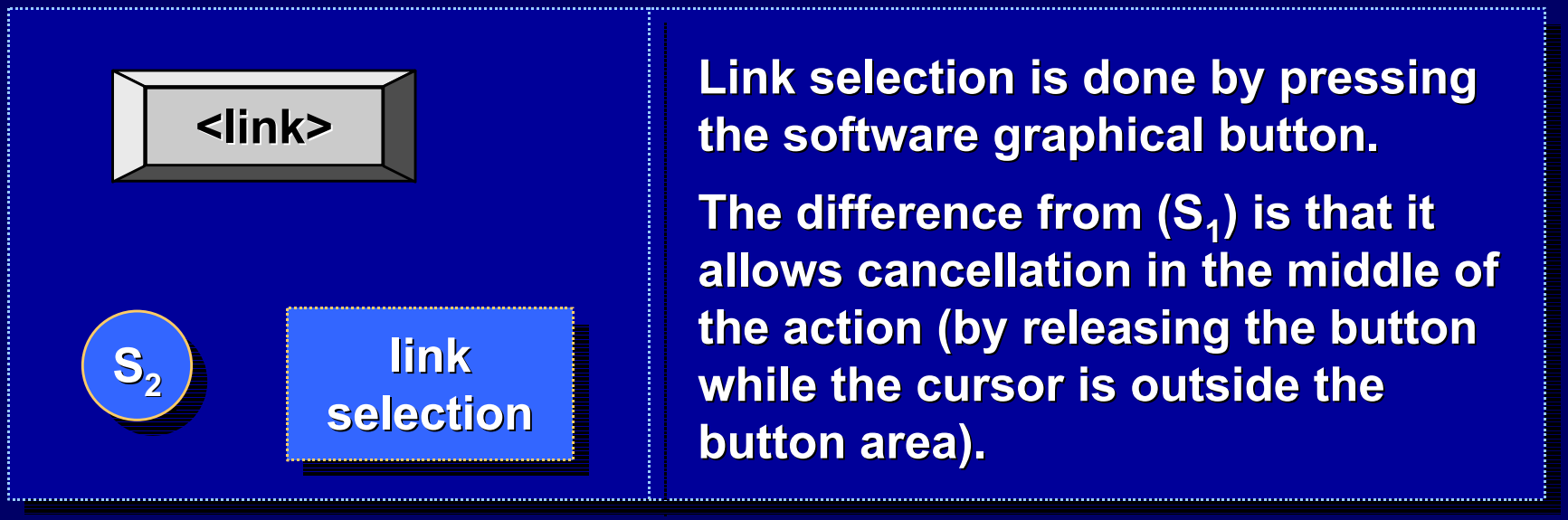
Link selection task - style 1



Link selection is done as far as the mouse cursor is within the text area of a link and the left mouse button is pressed.

- Designed for frequent and expert users.

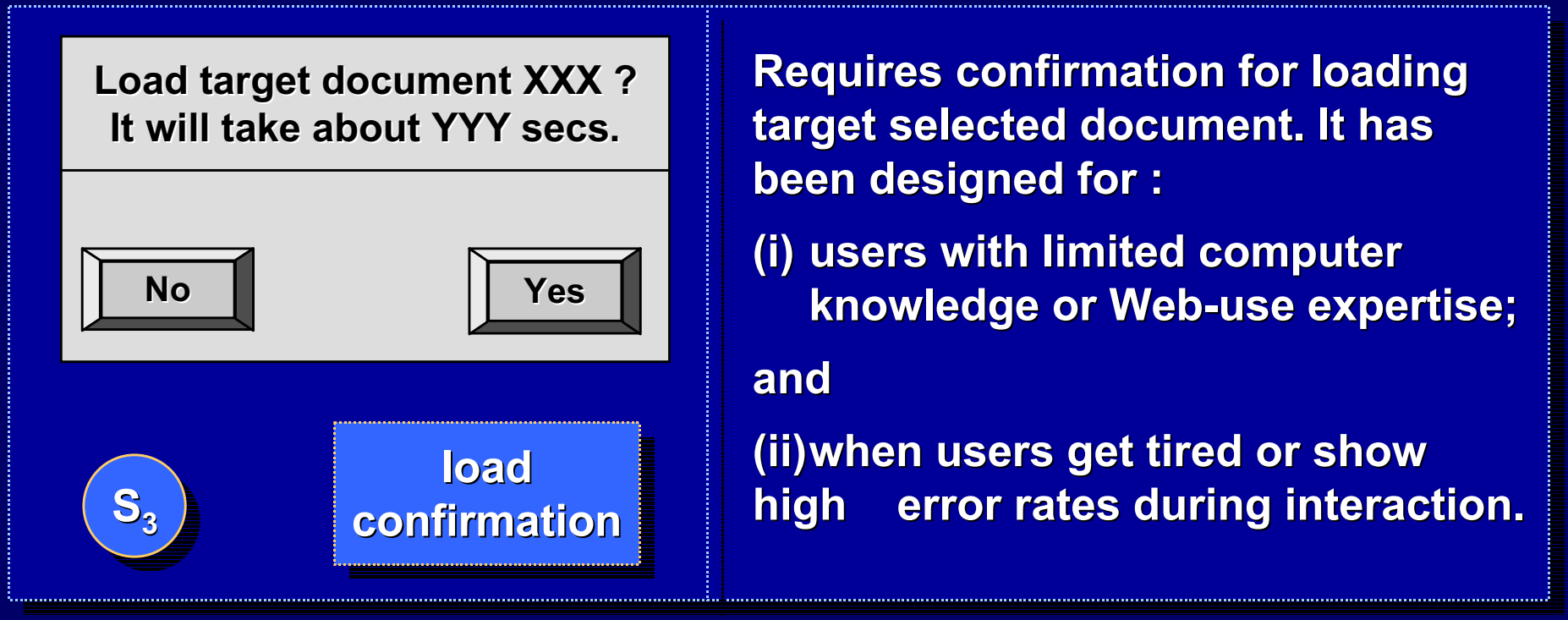
Link selection task - style 2



Link selection is done by pressing the software graphical button.

The difference from (S_1) is that it allows cancellation in the middle of the action (by releasing the button while the cursor is outside the button area).

Link selection task - style 3



Requires confirmation for loading target selected document. It has been designed for :

(i) users with limited computer knowledge or Web-use expertise;

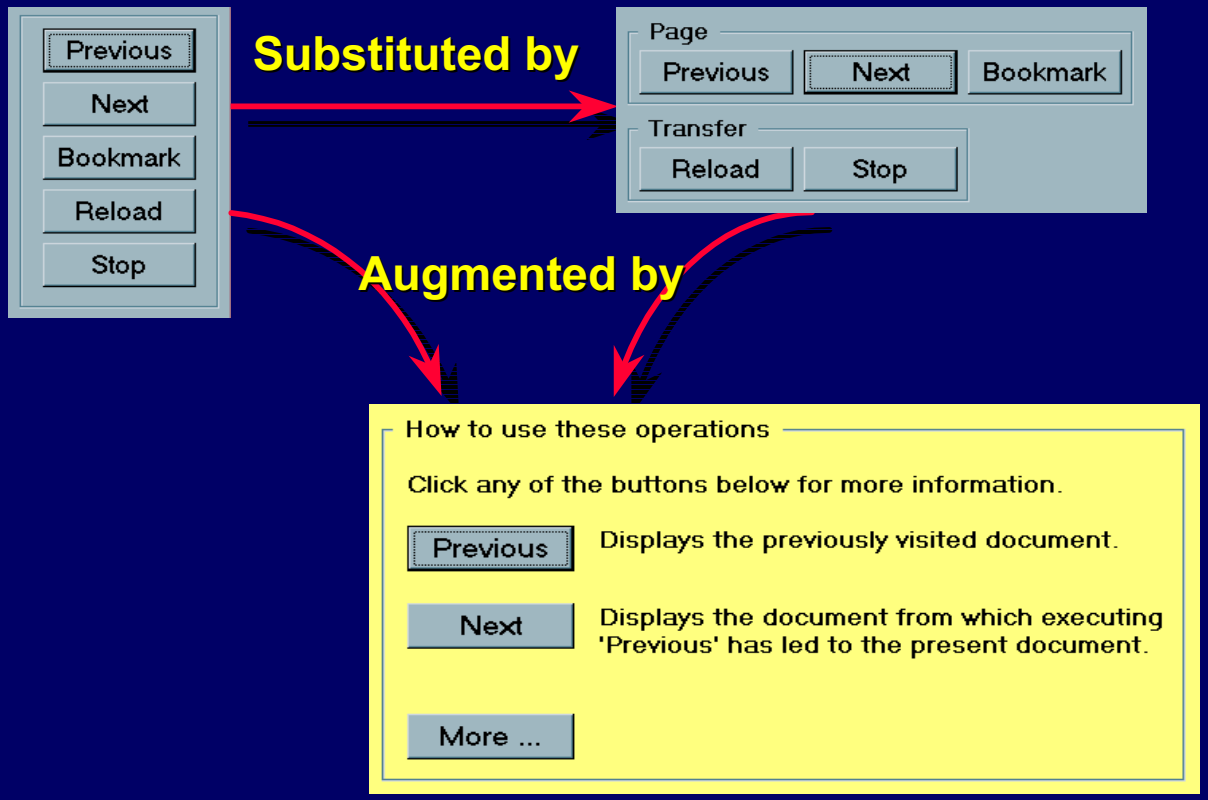
and

(ii) when users get tired or show high error rates during interaction.

Relationships among alternative styles for link selection task

- S_1 mutually exclusive with S_2.
- S_3 mutually exclusive with S_e.
- S_3 substitutes S_e dynamically (if high error rates are detected).
- S_e substitutes S_3 dynamically (if in a satisfactory interaction history, link activation has been always followed by positive confirmation).

Polymorphism for page loading control task (1/2)



- The top-left style (S_1) is the style to be initially active (casual / novice values on application expertise).

Polymorphism for page loading control task (2/2)

- The top-right style (S_2) is to dynamically substitute the previous style, in case that, during monitoring, it is observed that the user has used the operations successfully and became familiar with them.

  The new style groups logically operations with a title, and prepares the ground for the more advanced group called "options" to be included in future interaction sessions with this user (when the particular end-user completes a number of uses that will be considered as an advance to the "average" application expertise).

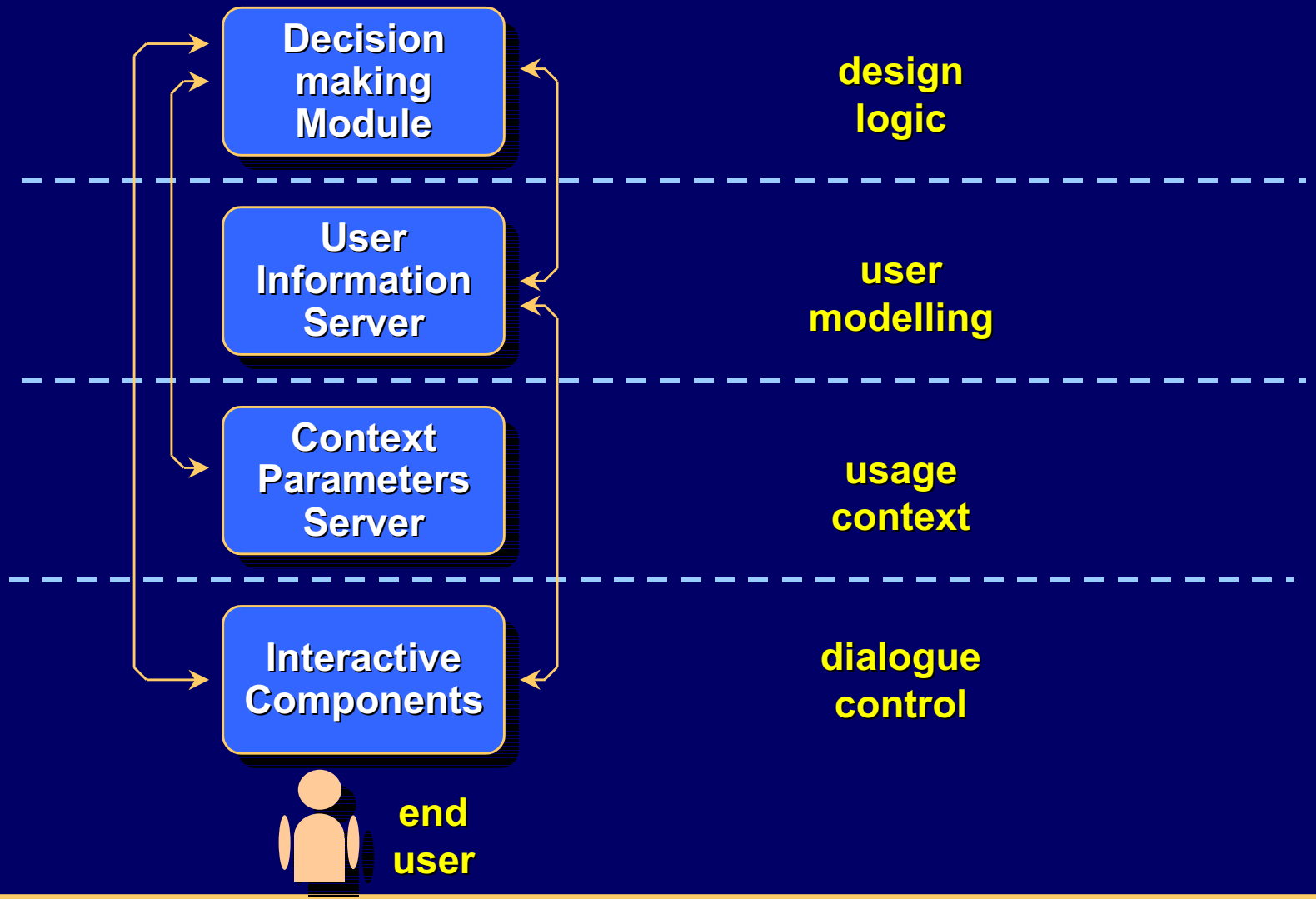
- The bottom style (S_3) augments the particular active style, and it provides adaptive prompting / helping for carrying out the operations in case that inability to perform the task or high error rates are dynamically detected.

The Unified user interface software architecture

Research question ...

- Provide a functional description of software components which would enable the user interface to reason towards and undertake the required adaptations, either prior or during interactive episodes, so as to preserve accessibility and exhibit intelligible interactive behaviour.

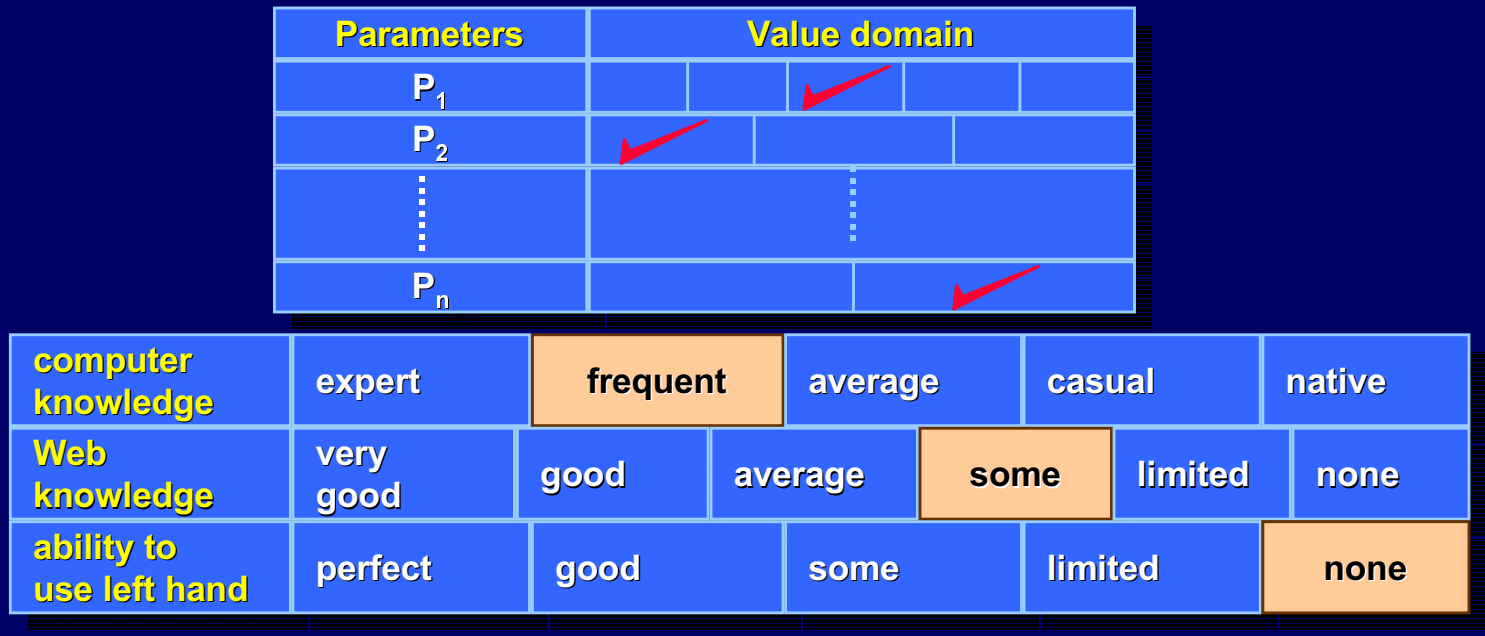
Overview of the transformable interface architecture



User Information Server

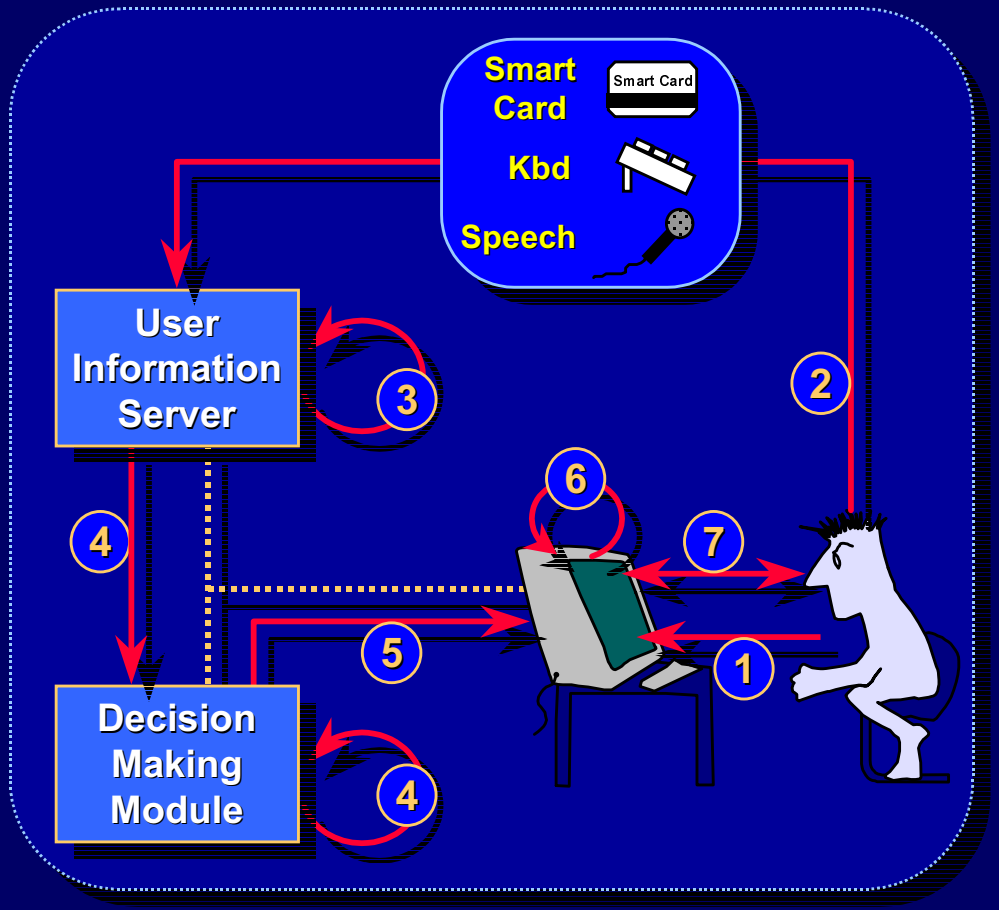
- Encompasses individual profiles of end-users.
- Initial user identification, upon start-up, is always required; in case of desk-top systems, with dedicated users, all interactive applications may gain a fixed "user id".
- A user profile is a collection of attribute-value pairs, however, additional knowledge-based modules may be employed with non-linear user representation structures.
- Independently of the user modelling approach locally adopted (within the UIS), the user profiles communicated to the rest of the architectural components, should always be in the form of attribute-value pairs.

A user profile example



- User abilities (sensory, motor, mental)
- User knowledge (domain specific)
- User role (domain specific)
- User interests (domain specific)
- User age (child, teenager, young, middle-aged, old)
- User preferences (hard-coded first priority dialogue style decisions)

Role of user profile in automatically adapted interaction



1. user approaches terminal
2. PIN typing, speech recognition or smart card
3. system recognises the user, activates the individual model
4. the end-user model (*parameters*) is taken to the decision module which draws interaction decisions
5. the decisions are sent to the unified interface
6. the unified interface activates the associated dialogue patterns, according to the parameter values
7. interaction starts with the activated dialogue components

Decision Making Component

- Three prime roles:

1. To encapsulate the design logic for activating (i.e. selecting) the most appropriate styles, on the basis of decision parameter values (i.e. user- and context-parameters).
2. To reason about alternatives on the basis of information received from other architectural components of a unified user interface
3. To provide the information needed to realise suitable adaptations

Context Parameters Server

- It encapsulates information regarding usage environment and machine parameters.
- The choice of those context parameters which are important within a particular interactive application is the responsibility of interface designer(-s).
- Dynamically changing context parameters may results in style substitution, or in the modification of dialogue parameters for activated styles.

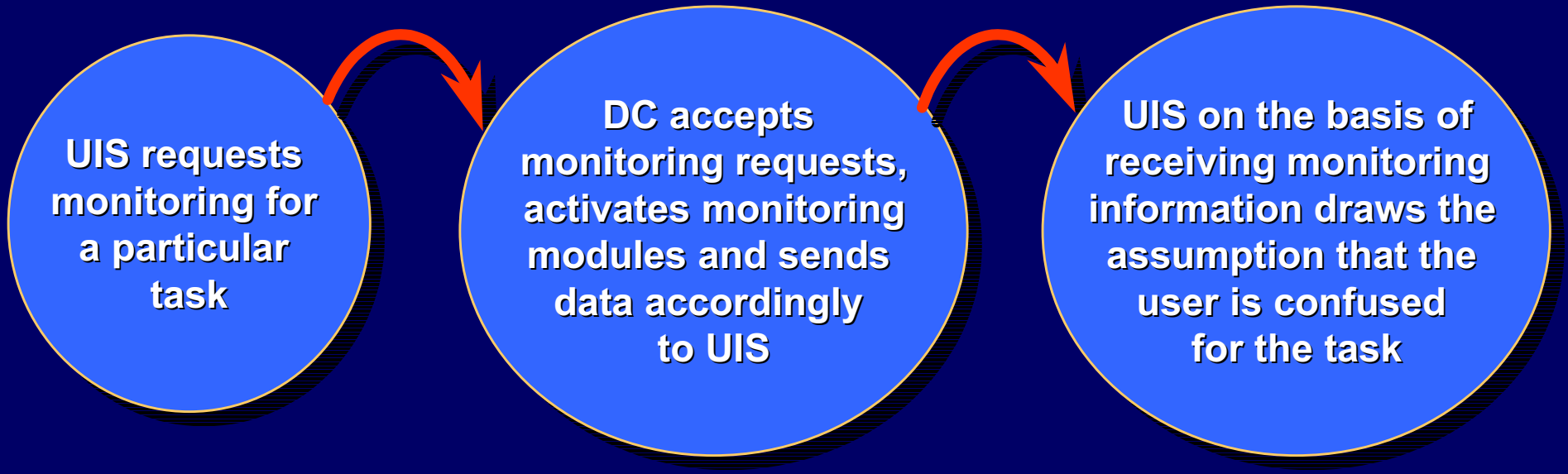
Interactive components

- These are the implemented dialogue patterns
- Encapsulated mechanism(s) that undertake the application of adaptation decisions (e.g. activation)
- Encapsulated monitoring mechanism(s) (integrated or attached)

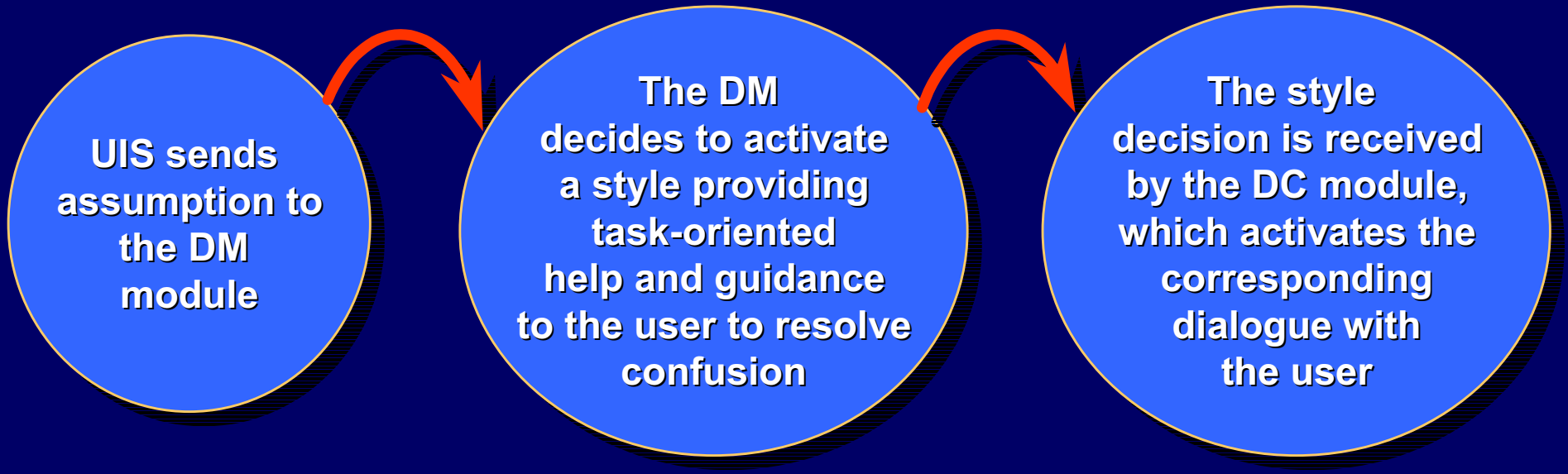
Note:

- The unified user interface architecture is orthogonal to the architectural abstraction underpinning a target platform

A communication scenario among components for adaptation (1/2)

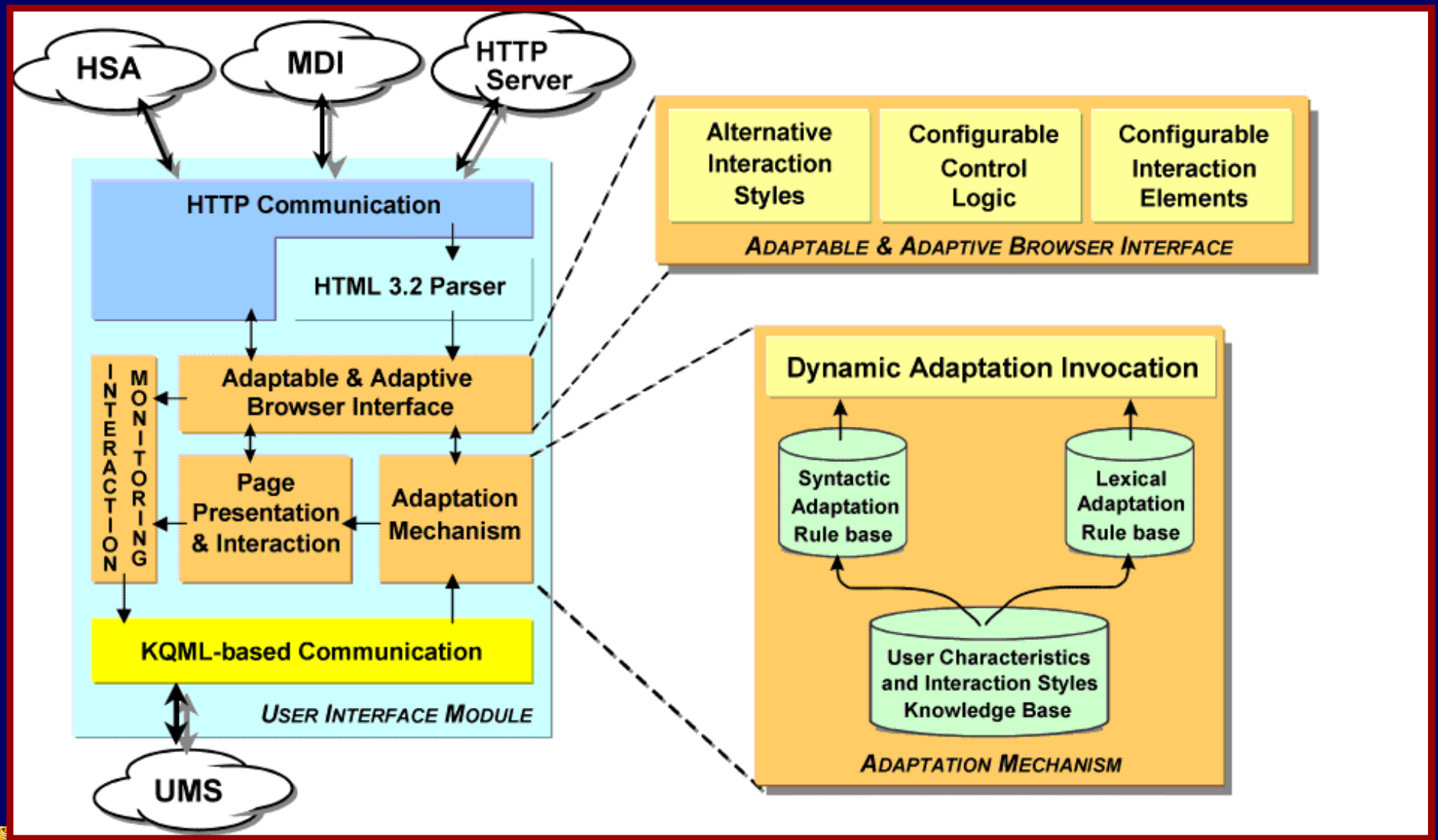


A communication scenario among components for adaptation (2/2)



Part IV: The AVANTI Case Study

Overview of the AVANTI Browser (1/3)



Overview of the AVANTI Browser (2/3)

- Adaptation-specific characteristics:
  - rule-based decision mechanism
  - adaptations mainly at the lexical and syntactic levels of interaction
  - information considered: user-, context-, and machine-characteristics
  - embedded models: user abilities skills and preferences, task model, dialogue model, application model
  - external models: dynamic user model

Overview of the AVANTI Browser (3/3)

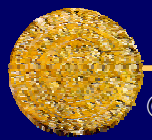
Placement in taxonomies

- Totterdell and Rautenbach
  - ✔ adaptive (apparent learning)
- Kuhme et al.

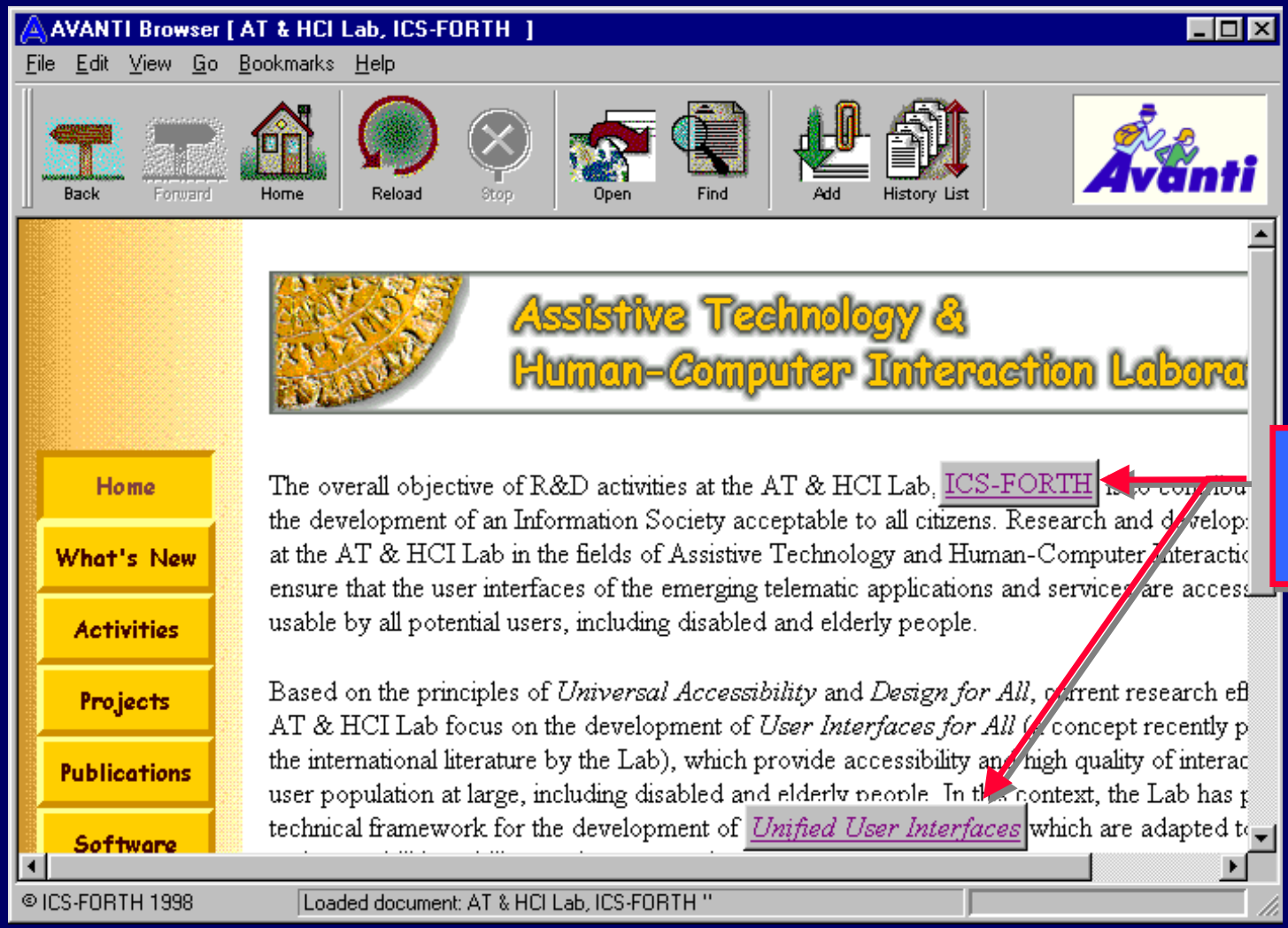
	System	User
Initiative	●	●
Proposal	●*	
Decision	●+	●
Execution	●	●

* Proposals made implicitly

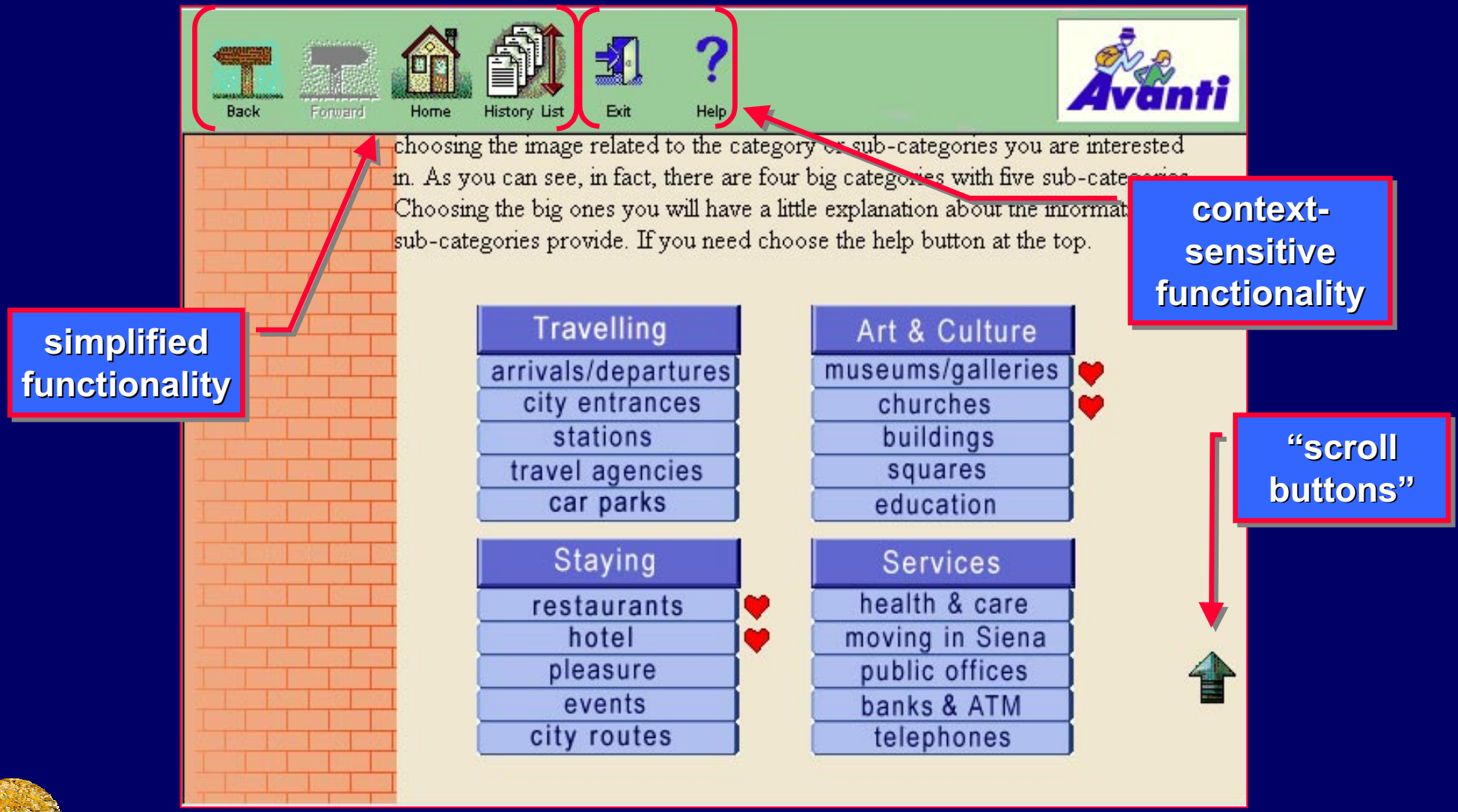
+ System decisions can always be overridden by the user



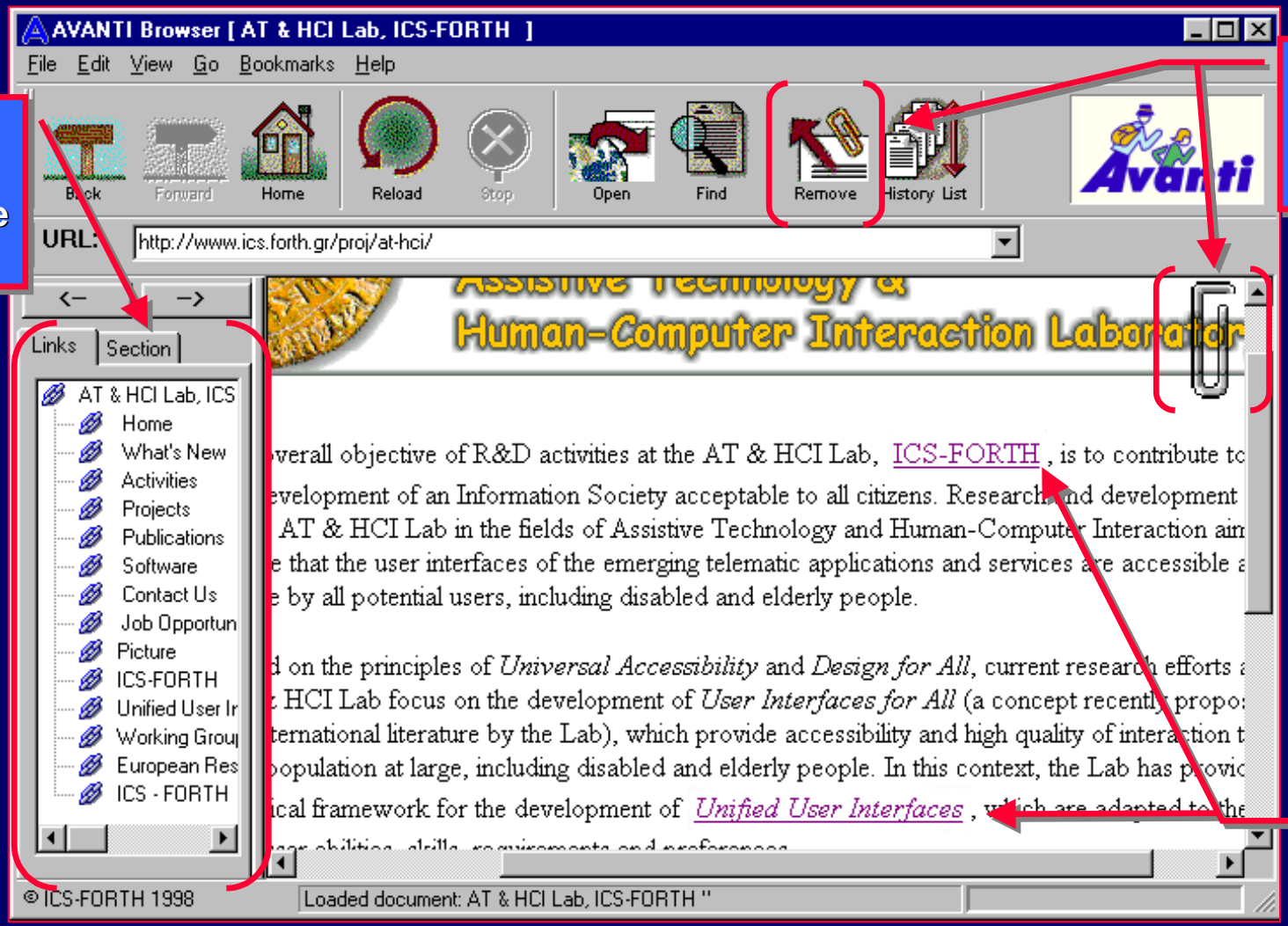
Instances of adaptability (1/6)



Instances of adaptability (2/6)



Instances of adaptability (3/6)

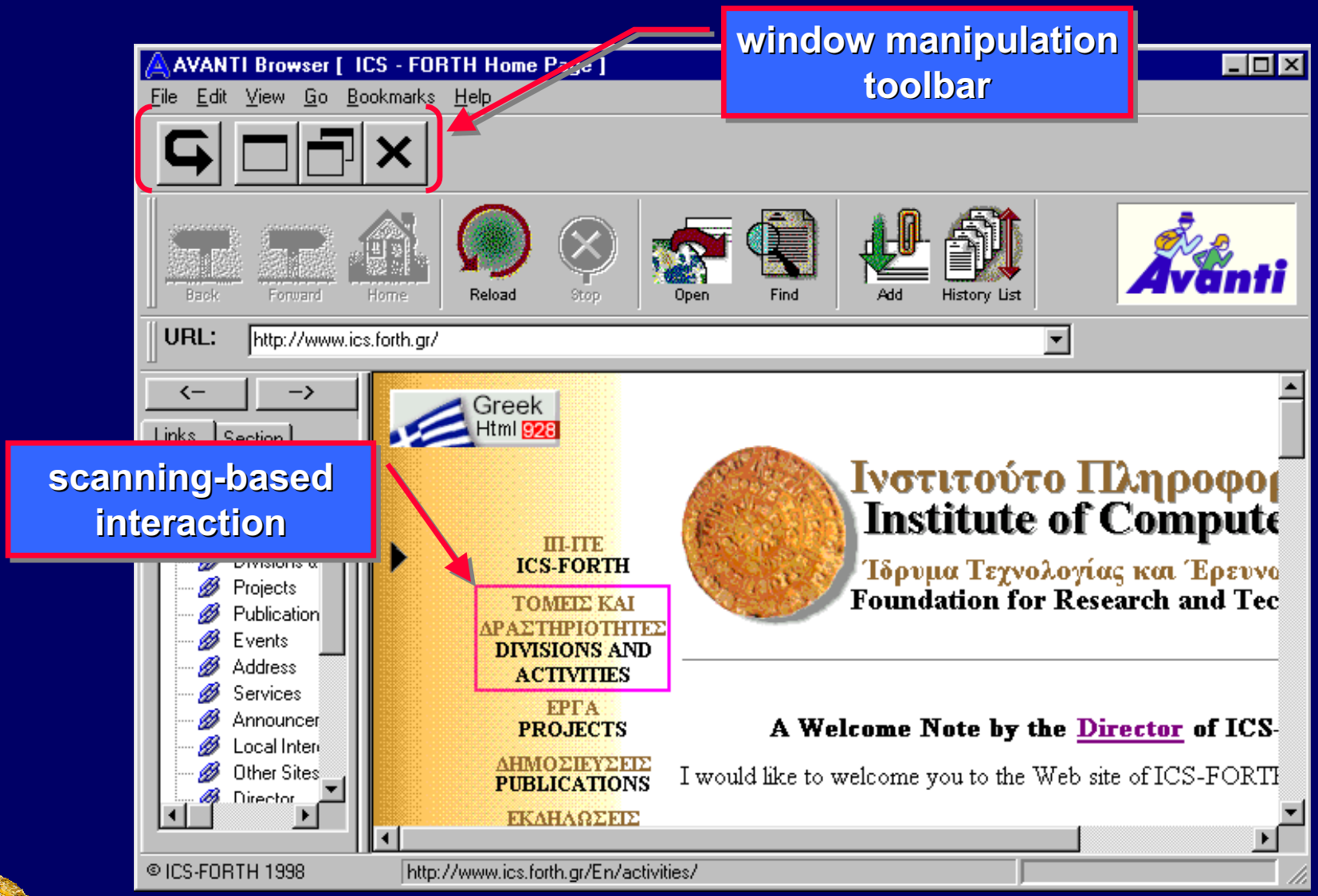


link replication and structure overview

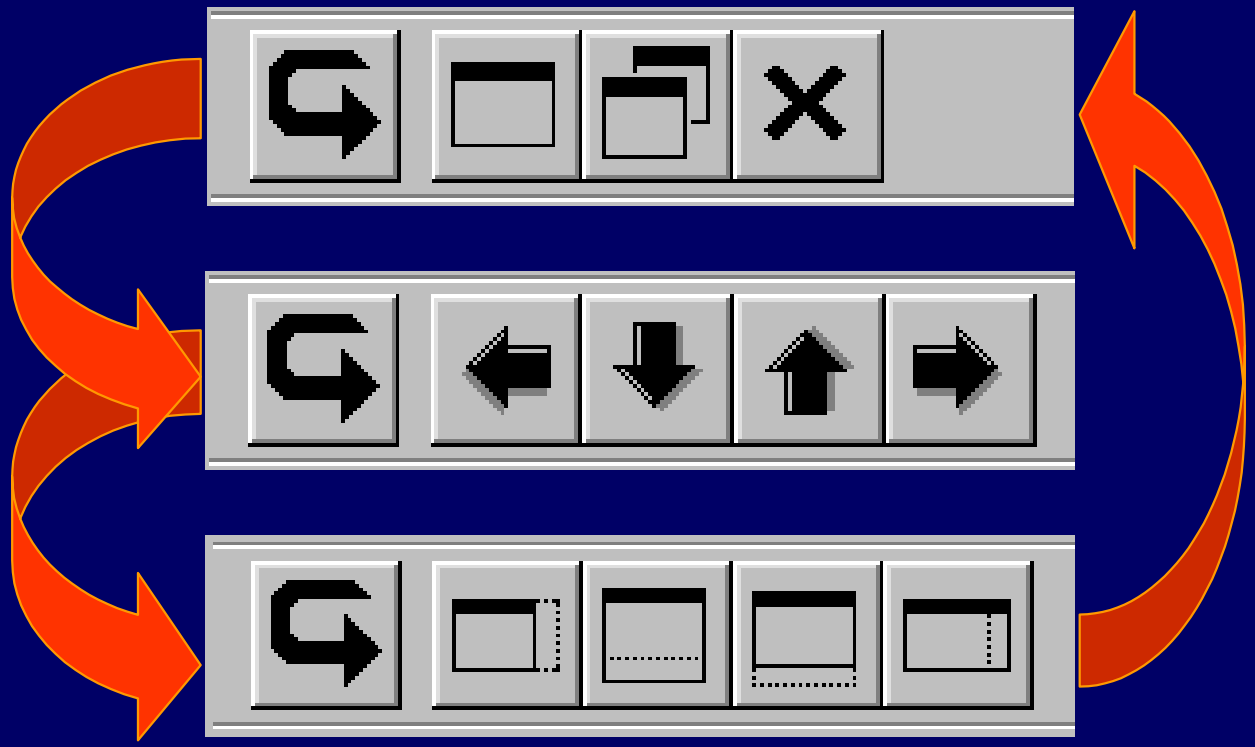
feedback on operation completion

links as text

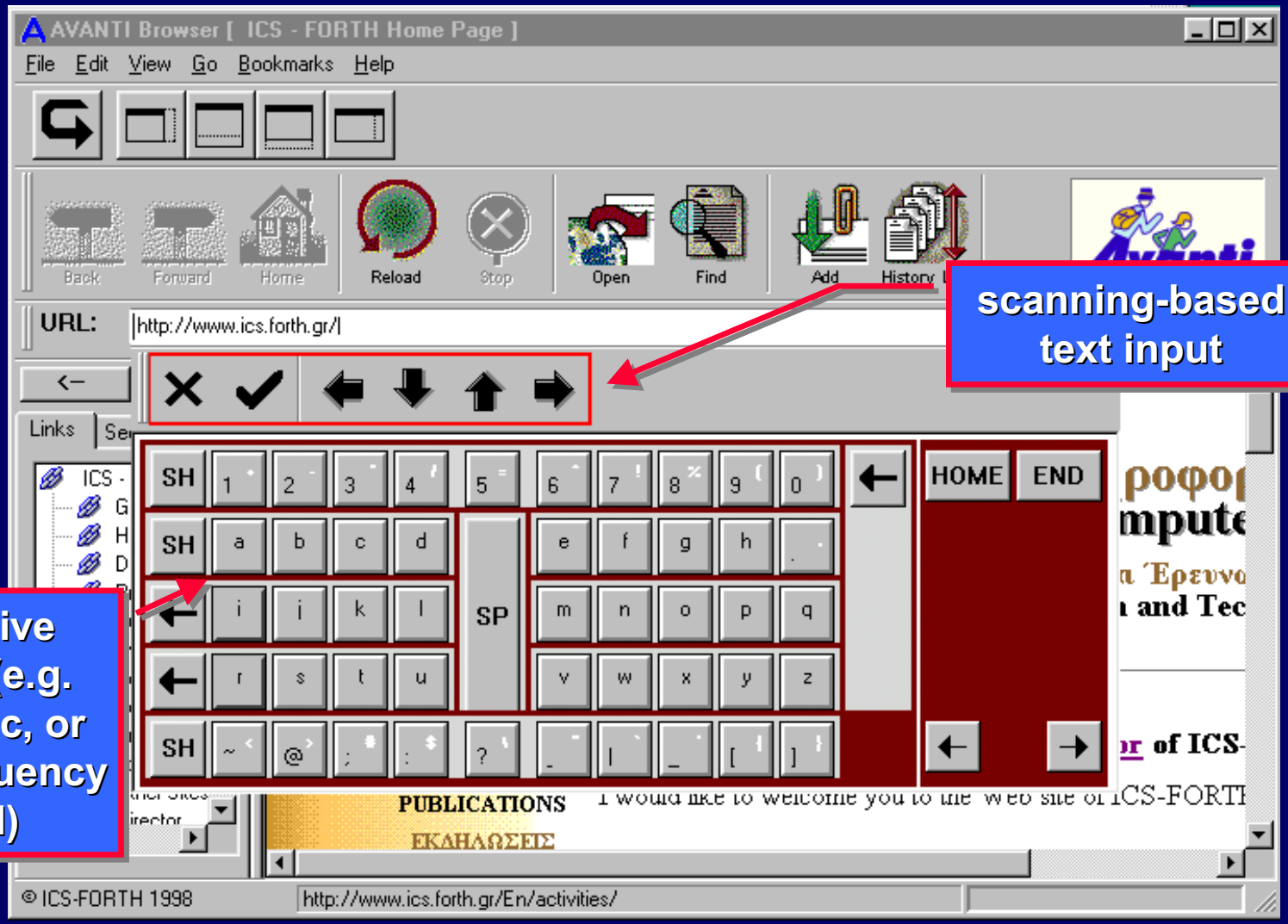
Instances of adaptability (4/6)



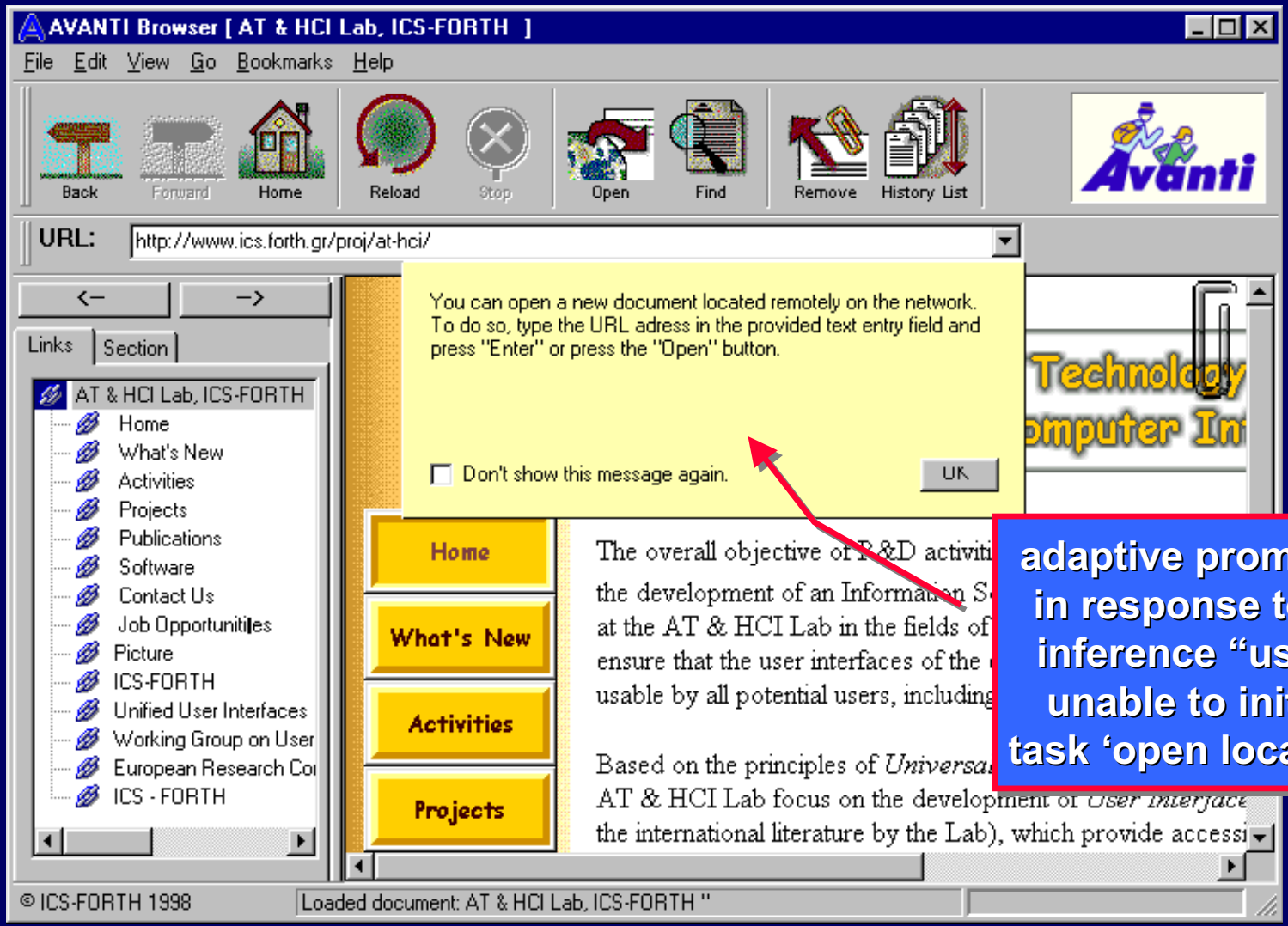
Instances of adaptability (5/6)



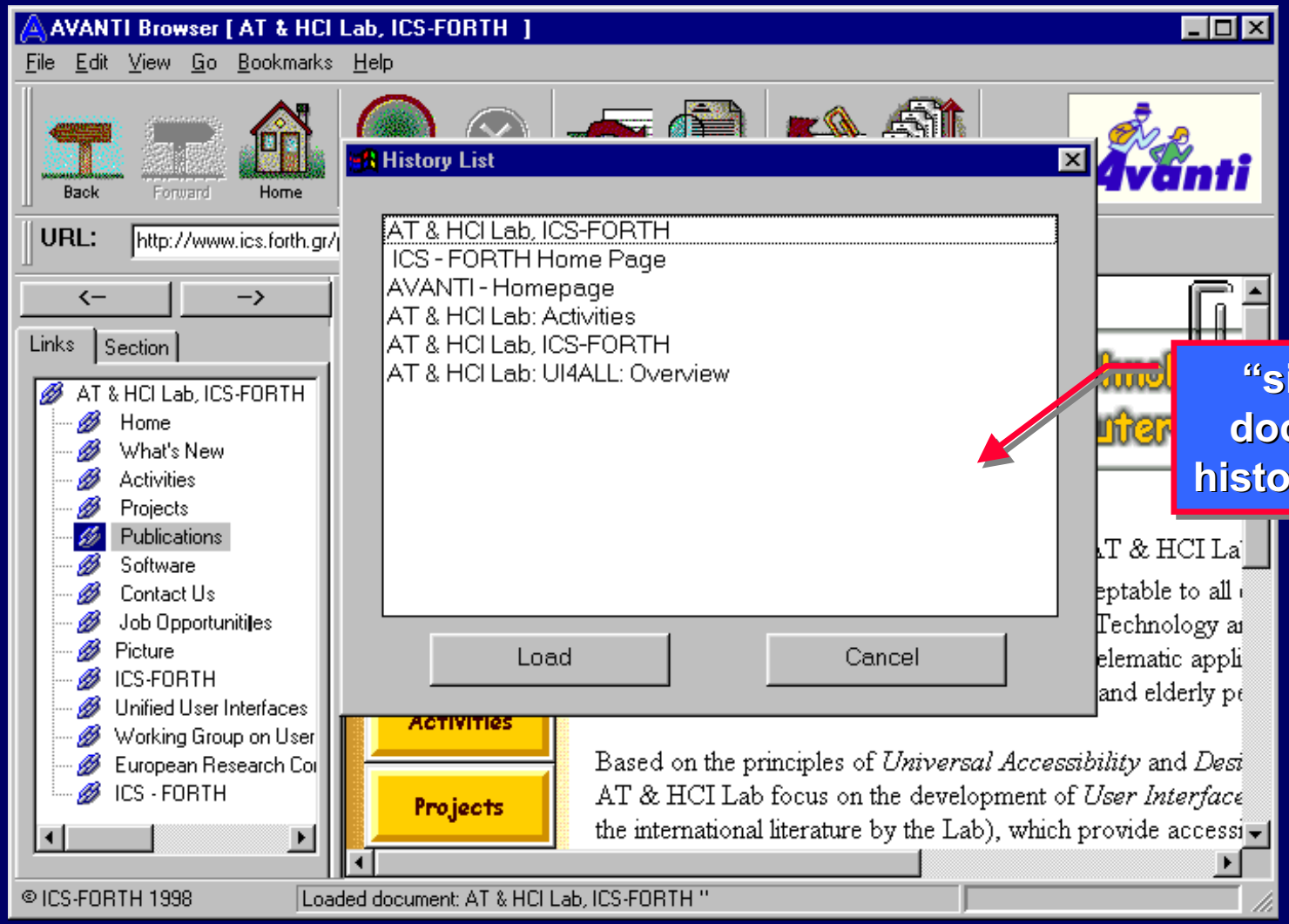
Instances of adaptability (6/6)



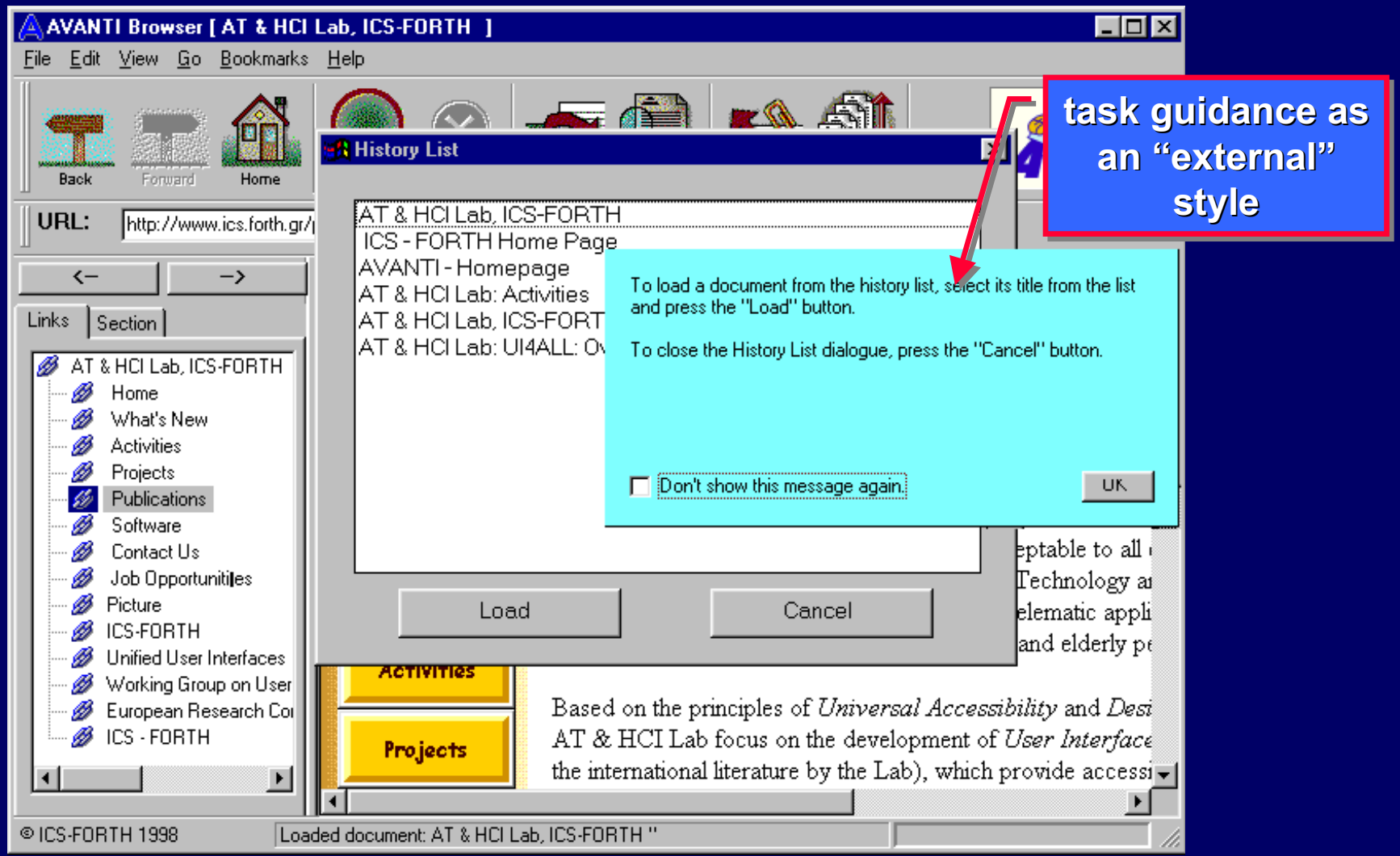
Instances of adaptivity (1/4)



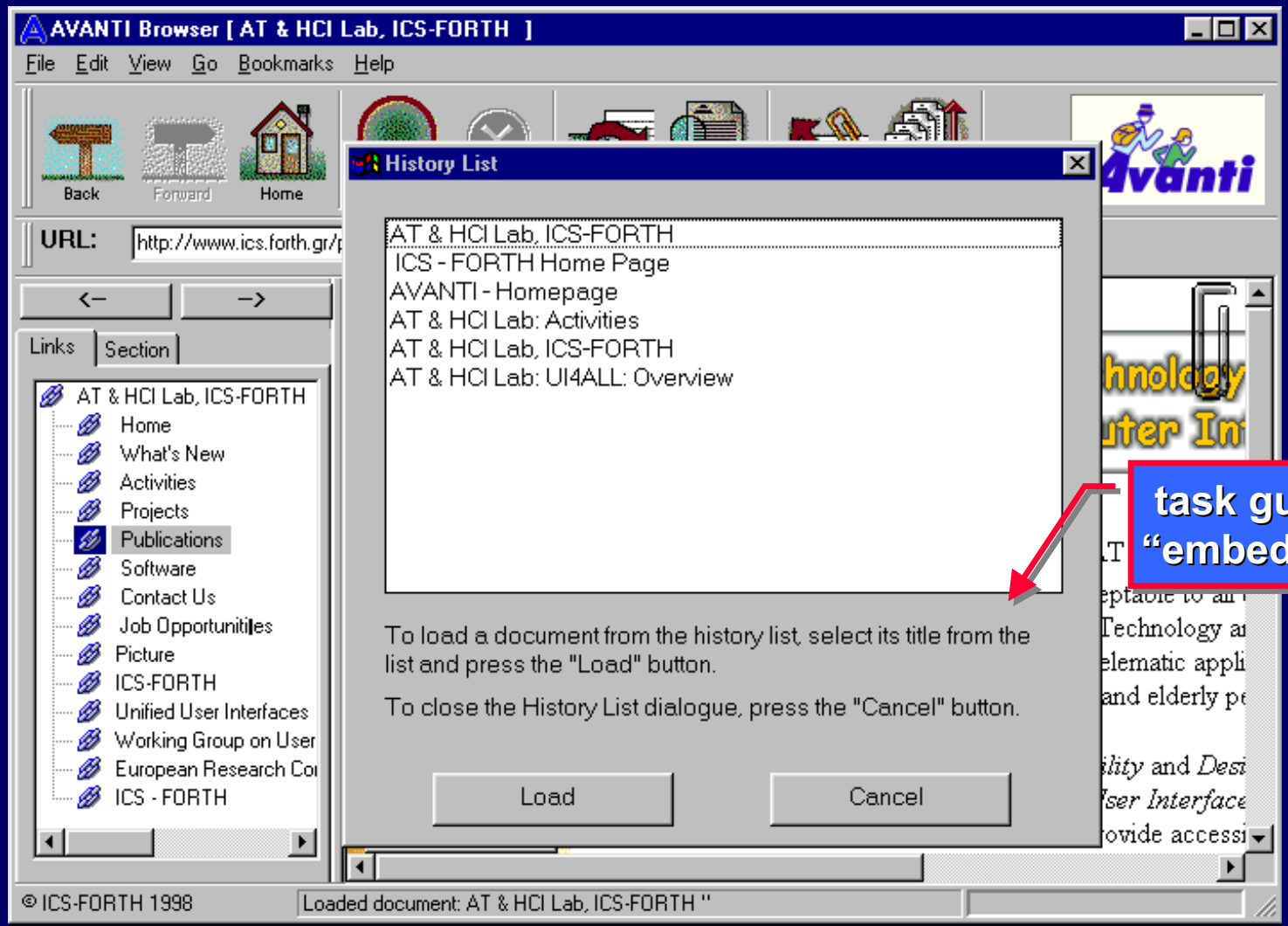
Instances of adaptivity (2/4)



Instances of adaptivity (3/4)



Instances of adaptivity (4/4)



Part V: Summary and Conclusions

Summary (1/3)

- Compelling need to address diversity in HCI
- The study of diversity in HCI raises implications on several levels
  - ✔ theory
  - ✔ design code (e.g. methods and tools)
  - ✔ practice

Summary (2/3)

- A core element to the study of diversity is the notion of adaptation
- The study of adaptation has become more complex, due to:
  - new application domains
  - novel interaction platforms
  - variety in the context of use
- The implications on HCI are on both:
  - the prevailing design code of practice
  - the architectural models that have influenced the development of interactive systems

Summary (3/3)

- The tutorial has presented
  - a design method to cope with the practicalities of adaptation in HCI design - The Unified Design Method
  - an architectural proposition comprising the functional components of an interactive system exhibiting adaptable and adaptive behaviour - The Unified User Interface Architecture
- Practical insights gained through collaborative R&D efforts
  - TIDE-ACCESS (focus on applications for speech-motor and language-cognitive impaired users)
  - ACTS-AVANTI (briefly presented in this tutorial)

Concluding remarks ...

- Adaptations have a strong situational character
- They require effective means to study context
- New design methods enriched with concepts from more developmental and social science approaches to HCI
- The Unified Design Method offers one particular perspective which is worth considering when tasked with the development of interactive systems which need to exhibit adaptable and adaptive control

Acknowledgments (1/2)

Part of this R&D work has been carried out in the context of the TIDE TP1001 ACCESS project "Development platform for unified ACCESS to enabling environments", partially funded by the European Commission (DG XIII).

The ACCESS consortium comprises:

- ✓ **Consiglio Nazionale delle Ricerche**, *Italy (Prime Contractor)*
- ✓ **Foundation for Research and Technology - Hellas**, *Greece*
- ✓ **National and Capodistrian University of Athens**, *Greece*
- ✓ **Royal National Institute for the Blind**, *UK*
- ✓ **University of Hertfordshire**, *UK*
- ✓ **SELECO Spa**, *Italy*
- ✓ **MA Systems & Control**, *UK*
- ✓ **Hereward College**, *UK*
- ✓ **National Research & Development Centre for Welfare and Health**, *Finland*
- ✓ **VTT Information Technology**, *Finland*
- ✓ **Pikomed**, *Finland*

Acknowledgments (2/2)

The AVANTI Web Browser has been developed in the context of the ACTS AC042 AVANTI project "**A**dapti**V**e and **A**daptable i**N**teractions for multimedia **T**elecommunications appl**I**cations", partially funded by the European Commission (DG XIII).

The AVANTI consortium comprises:

- ✓ **ALCATEL Siette**, *Italy (Prime Contractor)*
- ✓ **Consiglio Nazionale delle Ricerche**, *Italy*
- ✓ **Foundation for Research and Technology - Hellas**, *Greece*
- ✓ **GMD**, *Germany*
- ✓ **University of Sienna**, *Italy*
- ✓ **MA Systems & Control**, *UK*
- ✓ **MATHEMA**, *Italy*
- ✓ **VTT Information Technology**, *Finland*
- ✓ **ECG**, *Italy*
- ✓ **University of Linz**, *Austria*
- ✓ **TELECOM Italia**, *Italy*
- ✓ **EUROGICIEL**, *France*