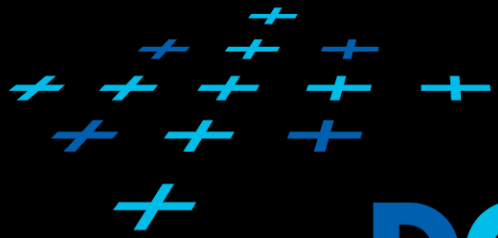


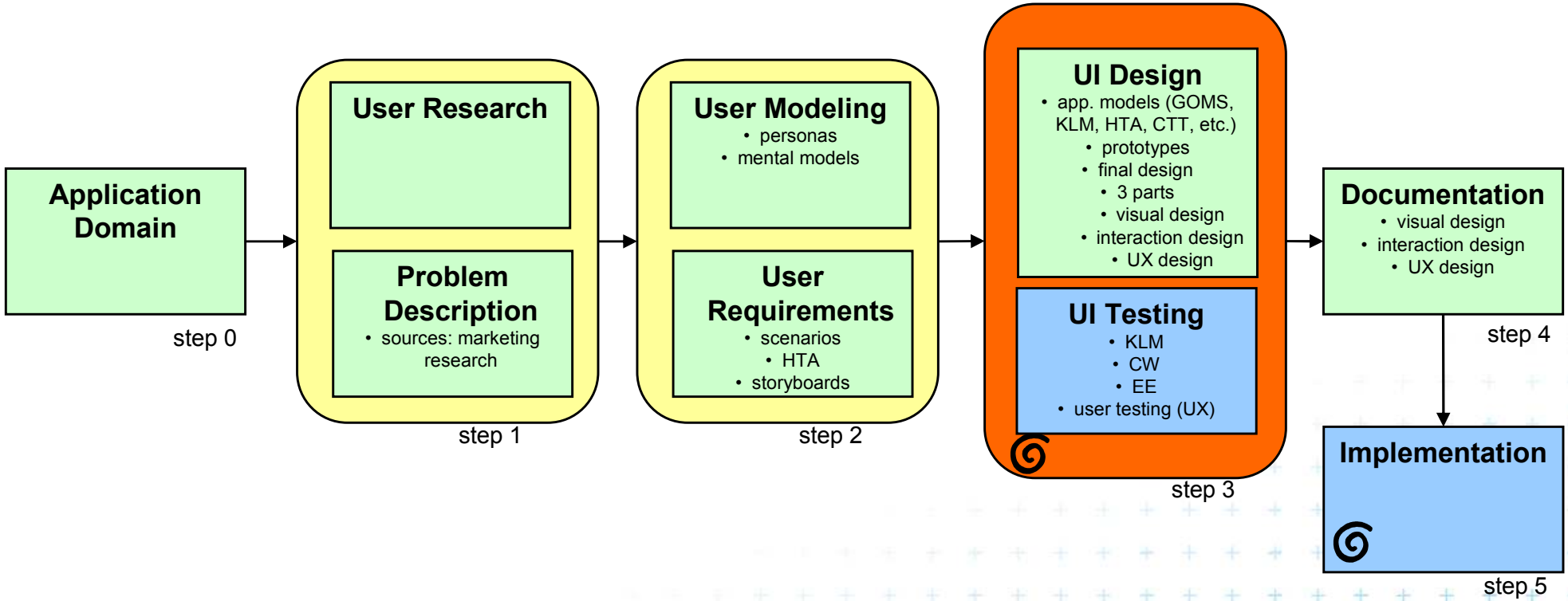
DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

MODELS AND ARCHITECTURE

LECTURE 10

User interface design - big picture



What is interaction?

communication

user ⟺ system

but is that all … ?

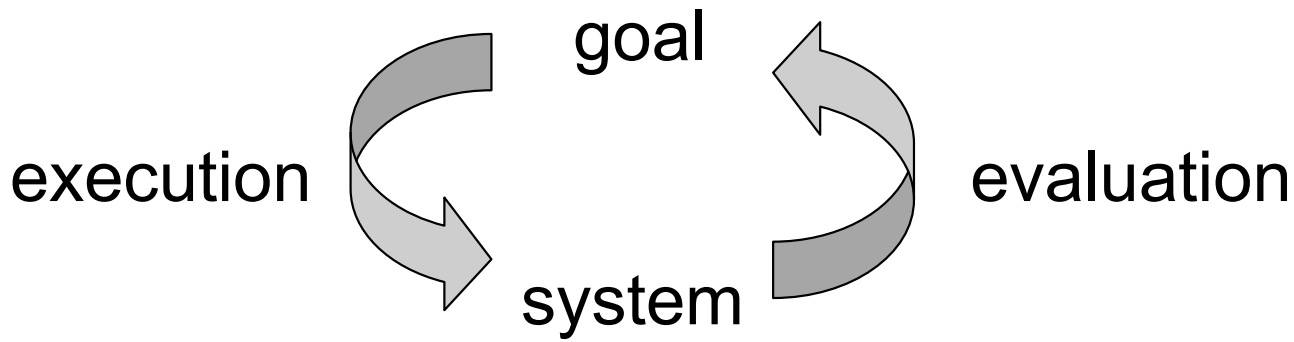
HCI = Human Computer Interaction

- Understanding of processes in both parts
- How to describe activities in these parts?
- What are links between these parts?
- How the communication is performed?

HCI

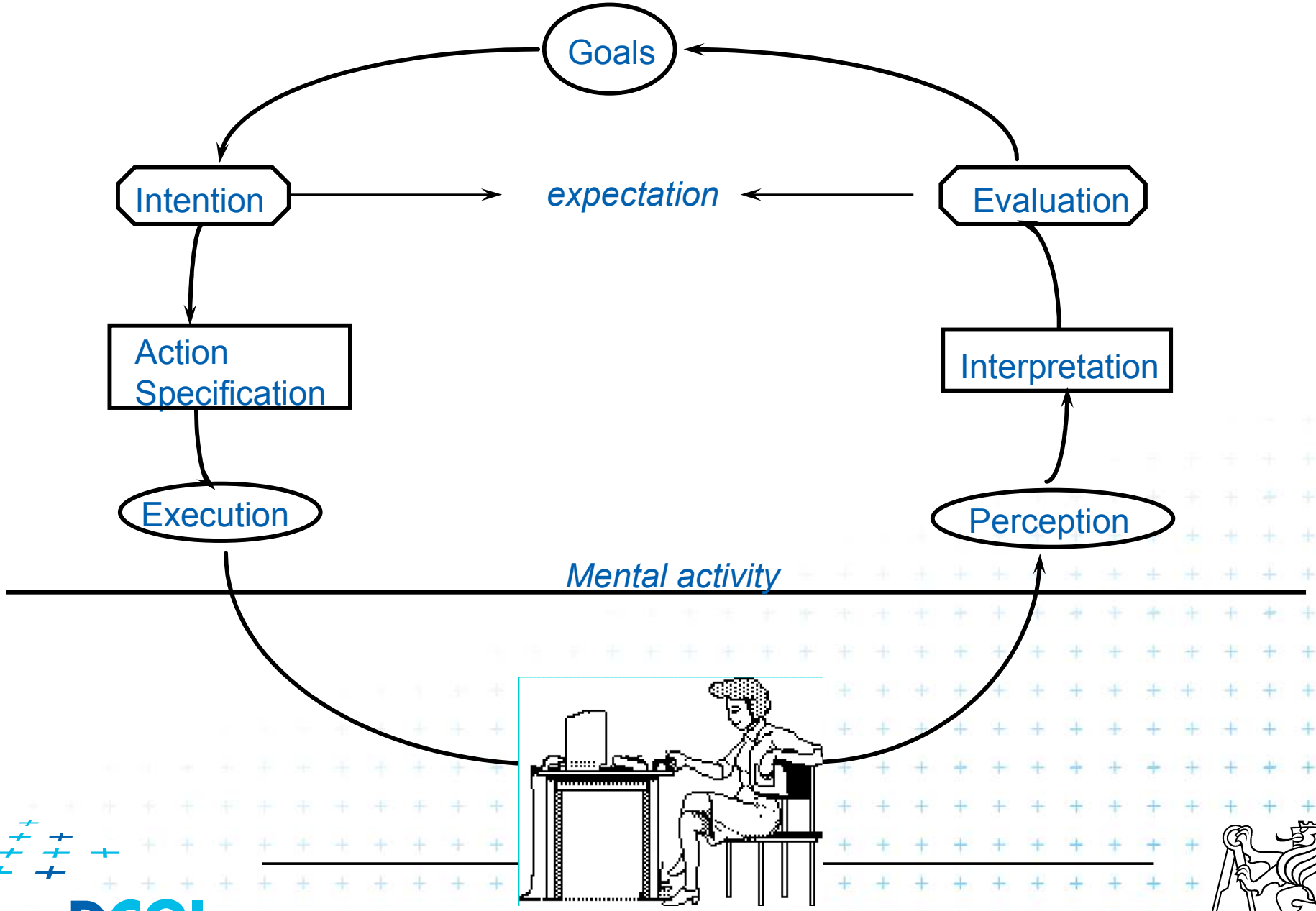
- Which formalisms can be used?
- Some formalisms have been already discussed
- Human side: HTA (task related)
- Computer side: STN (dialog structure described)
- What is missing?
- E.g.: info about implementation issues (what the structure of software product will look like)
- E.g.: how to prepare concept of the UI design
- Etc…

Execution/evaluation loop

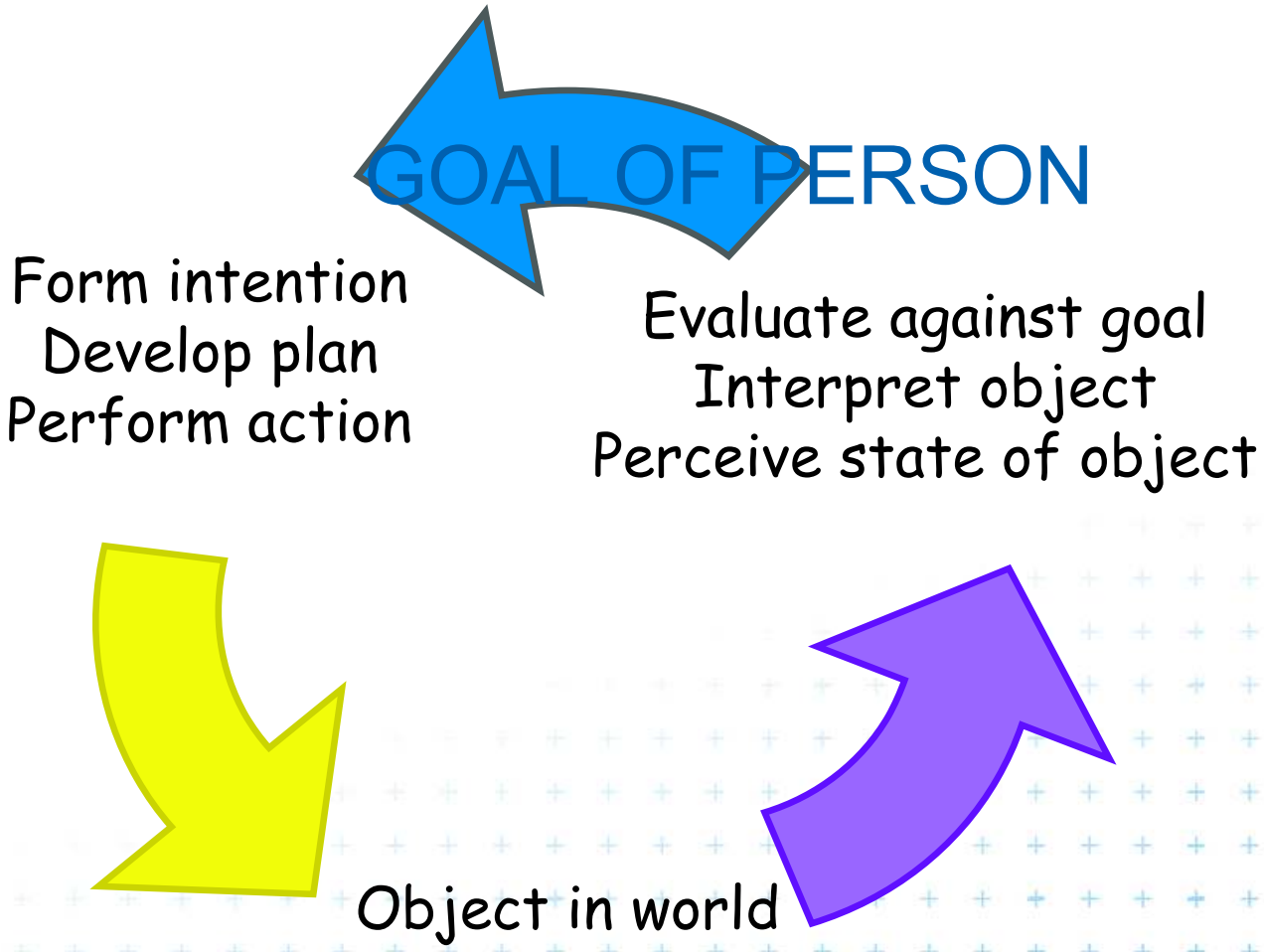


- user establishes the goal
- formulates intention
- specifies actions at interface
- executes action
- perceives system state
- interprets system state
- evaluates system state with respect to goal

The stages of user activities when performing a task

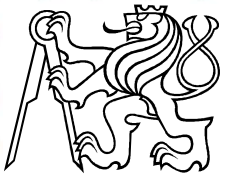


Seven Stage Action Model [Norman, 1990]



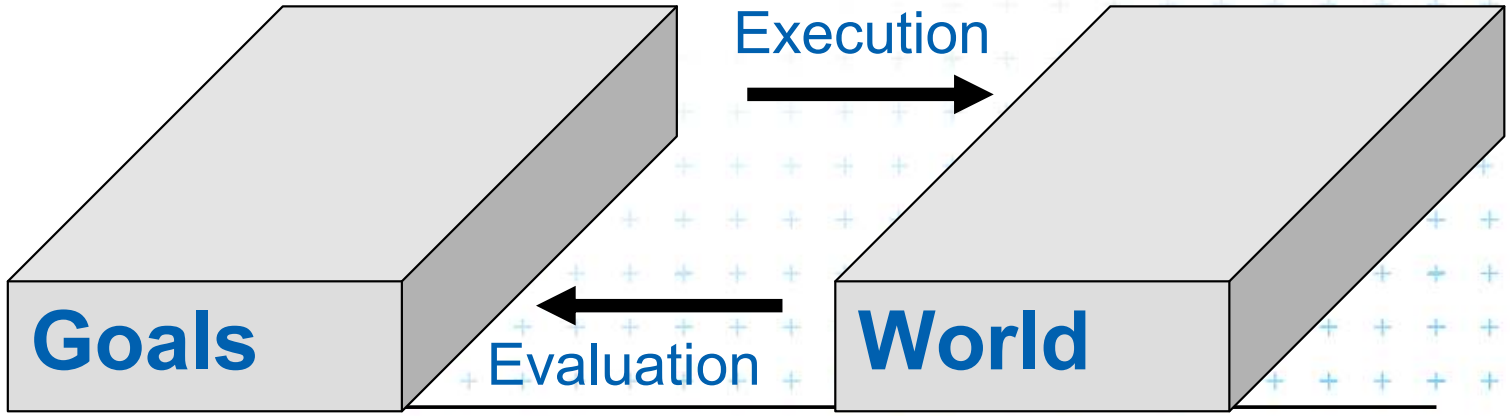
Exercise #1

- Use the Seven Stage model to represent the activities associated with withdrawing £20 from a cashpoint machine



Gulfs of execution and evaluation

- Gulf of execution:
  - How does the user translate intentions into action?
- Gulf of evaluation:
  - How does the user understand the effects of actions and does s/he tell when her/his goals are satisfied?



Bridging the gulfs

- Execution
  - Mappings:
    are actions designed so the users make the connection between the effects they intend to achieve and the actions provided by the system?
- Evaluation
  - Feedback:
    is information about the system state provided in a way that allows users to determine whether goals are satisfied?

Using Norman's model - Summary

Some systems are harder to use than others

Gulf of Execution

user's formulation of actions

≠ actions allowed by the system

Gulf of Evaluation

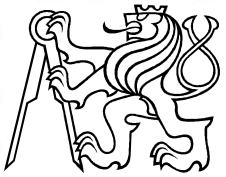
user's expectation of changed system state

≠ actual presentation of this state

UI and user behavior

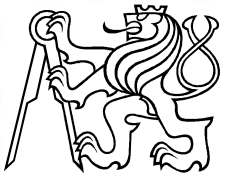
Concept of UI design

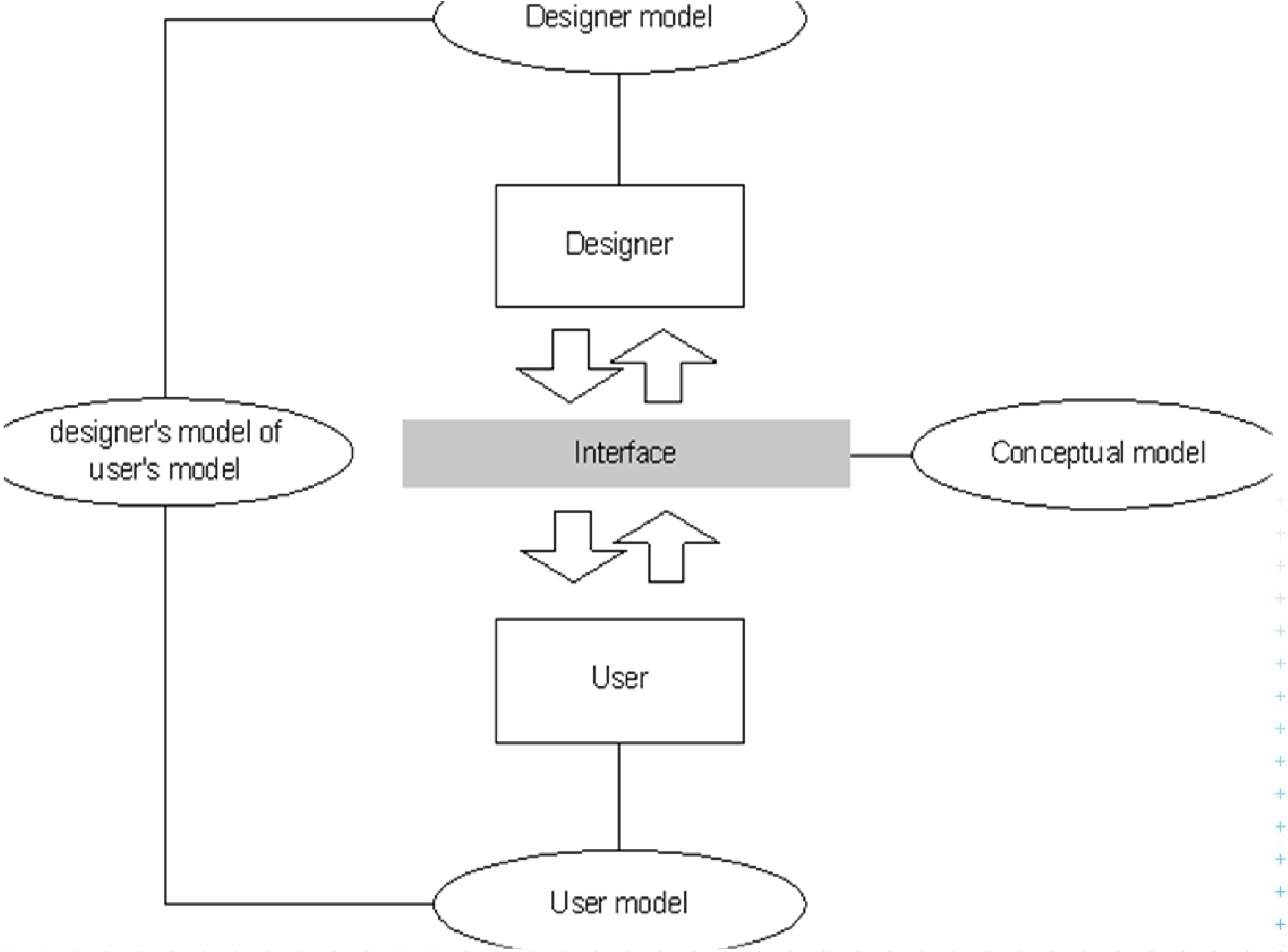
- Mental and conceptual models
- What kind of information they will deal with?
- Roughly speaking: they “force” the designer to take into account the user’s way of thinking



Mental and Conceptual model

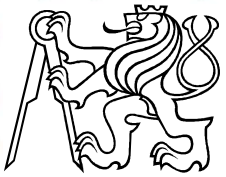
- Need to first think about how the system will appear to users (i.e. how they will understand it)
- A **conceptual model** is the designer's intended mental model for the user of the system: a set of ideas how it is organized and how it operates (Norman calls it a design model)
- We form a mental model of the function, which we use to predict the likely result of our actions, and hence choose what to do for a desired outcome
- Example: heating in a "cold" home




Designer model
Designer
designer's model of user's model
Interface
Conceptual model
User
User model

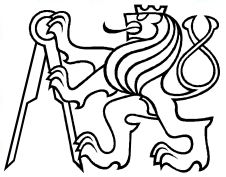
For either idea….

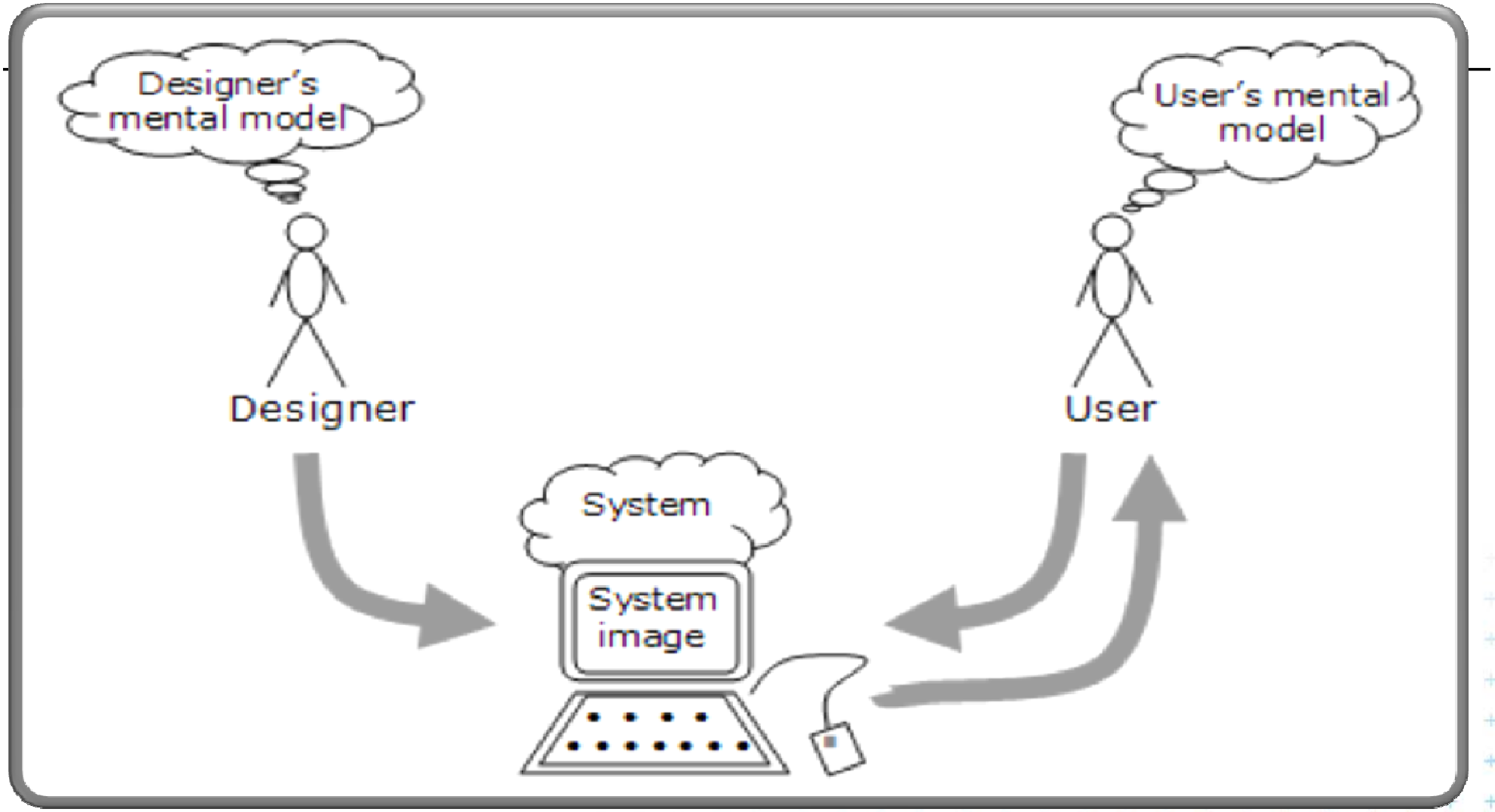
- You want to understand what the user already has in their head
- Then, you want to:
  - Maybe build your system in response to this
  - Work to create a different model in their head as they use your system
- Always keep in mind when making design decisions how the user will understand the underlying model



Mental Models

- A person's image about an artefact
- The user's image about the machine
- Making sense of the world from inside





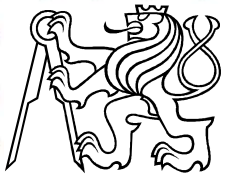
Mental-model in HCI

Mental Models

- An internal representation of a system that can be interrogated and manipulated.
- Mental models are concrete.
- Mental models can be run.
- Mental models are constructed from experience.
- Mental models are generally incomplete and inaccurate, but serve a purpose

Example Camera

- What are the components and how do they fit together?
- How does it work?
- What causes what?
- How do you use it?
- How do you use your understanding when something goes wrong?



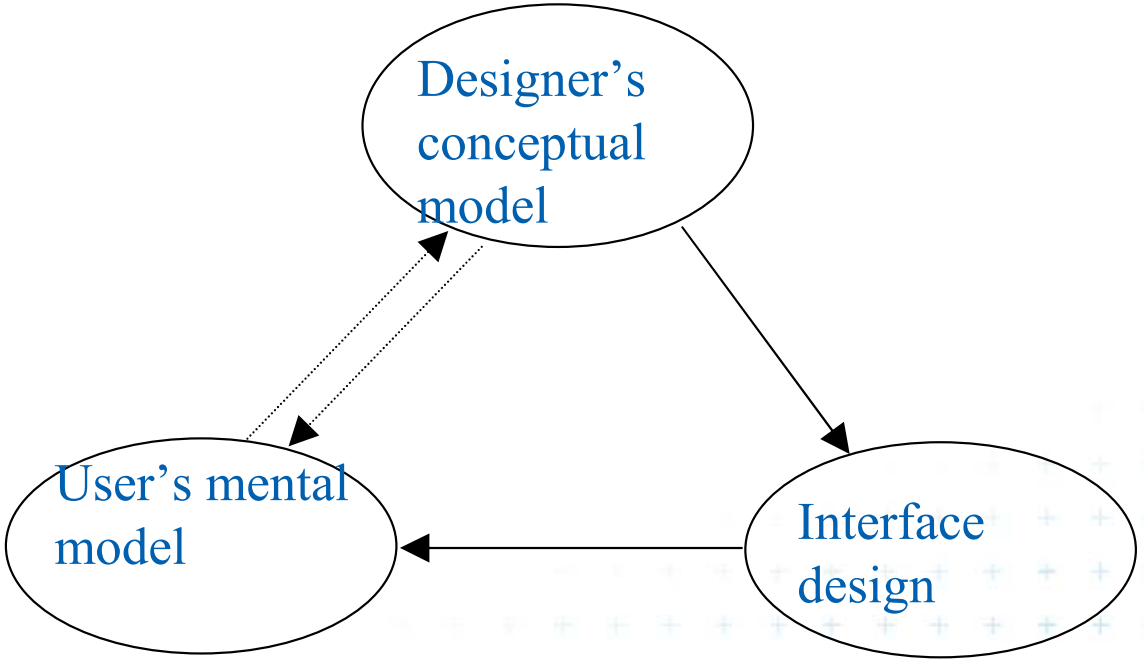
The Contents of a Model

- Kieras (1982) categorised the kinds of knowledge that people have about a device as follows.
  - Label or name of the device
  - Function or purpose (what goals can be accomplished)
  - Controls and indicators
  - Inputs, outputs and connections
  - Power sources and requirements
  - External layout and appearance
  - Internal layout and appearance
  - External behaviour (input-output function)
  - How to operate the device to accomplish goals
  - Procedures for troubleshooting and maintenance
  - Internal structure and mechanisms (how it works)

Mental Models

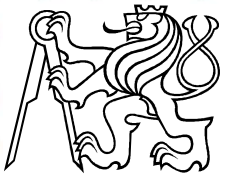
- A runable mental model = how-it-works knowledge + how-to-use-knowledge !
- How-it-works knowledge may be at various levels of detail.
- Strategic knowledge includes various strategies — e.g. inference, prediction, diagnosis. These are transferable skills.

How designers influence the user's mental model

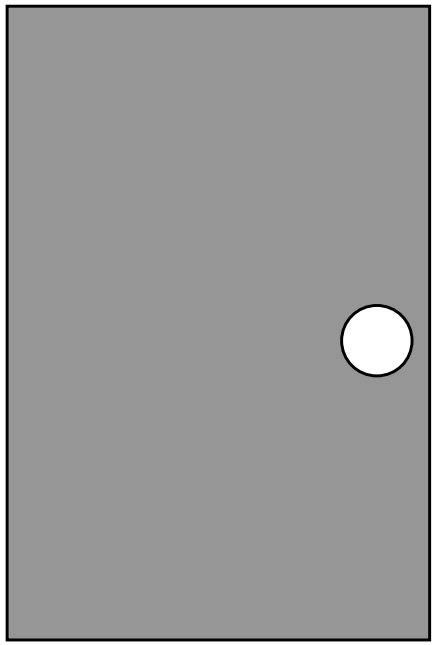


Mental models - Examples

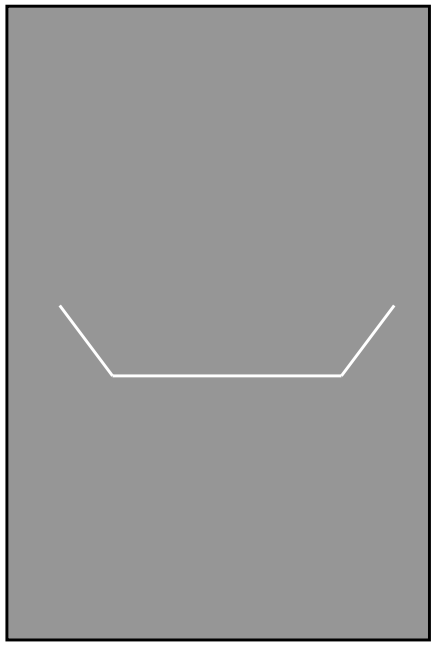
- Examples from everyday life
- User expects some behavior
- E.g. doors in coridor
- User tries to figure out how some device works



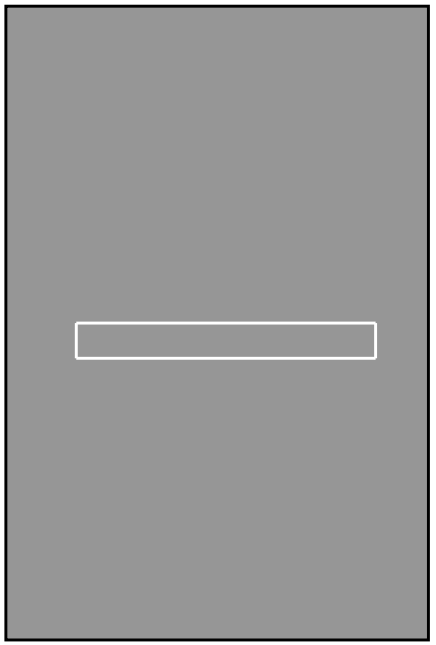
Doors (Good or Bad Affordances?)

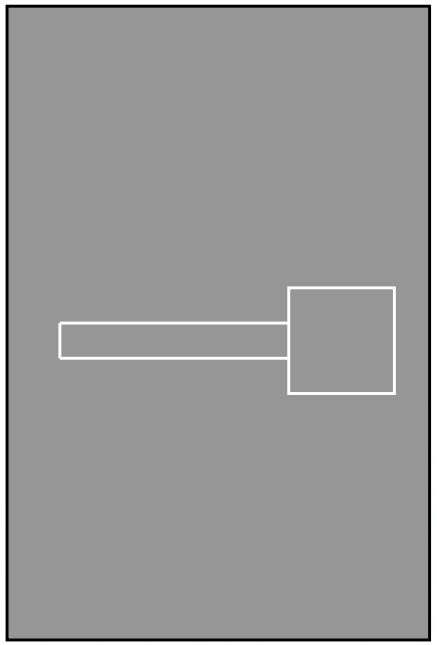


Push or pull?



Which side should I push?





Flat metal plate on one side tells me

Mapping

What Knob Goes Where?

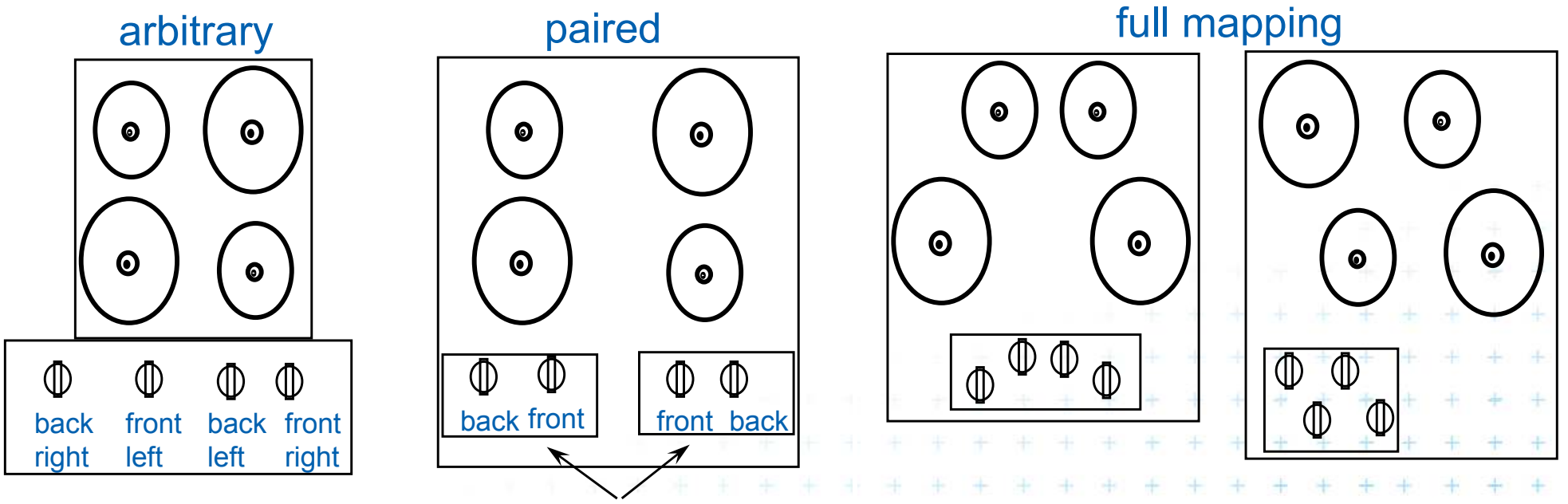
Mapping

Exploiting Natural Mapping

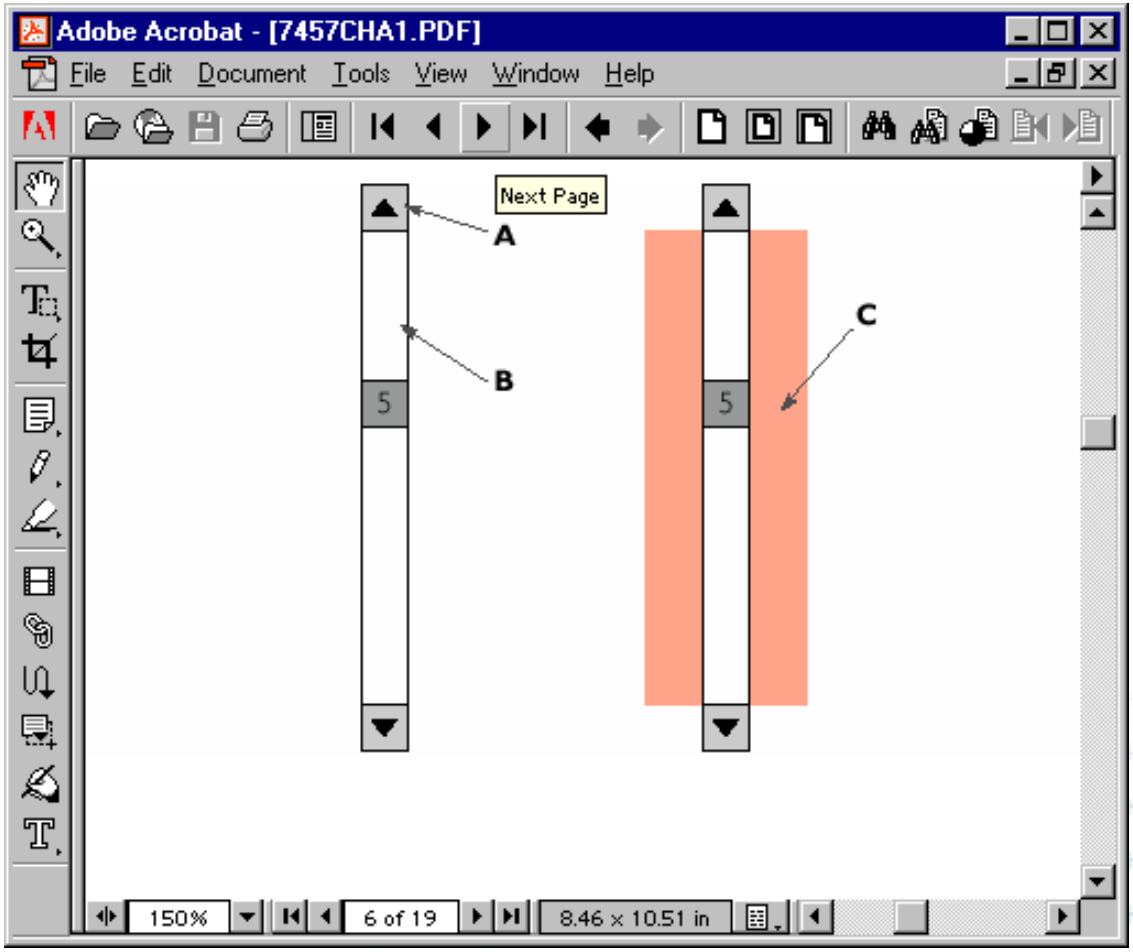
Affordance

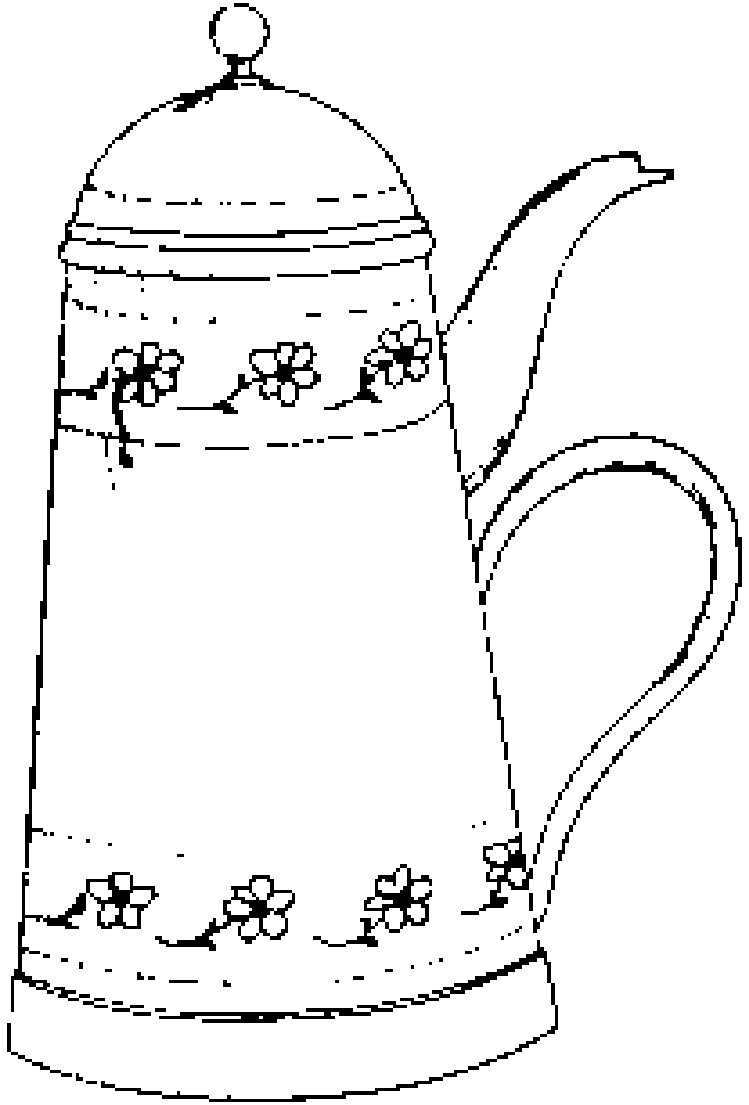

OFF
OFF
OFF
OFF
OFF
OFF
HI
LO
HI
LO
HI
LO
HI
LO
HI
LO
250
100
200
150
ON
200 230 260
400 450 500
OVEN
HOTPLATES
GRILL
Centenary

Mapping controls to physical outcomes



Visual affordances of a scrollbar



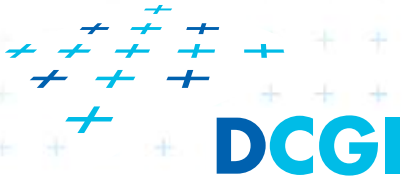


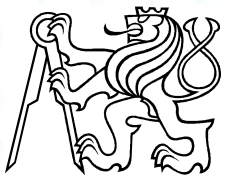
Slide adapted from Saul Greenberg

Transfer Effects

- People transfer their expectations from familiar objects to similar new ones
  - positive transfer: previous experience applies to new situation
  - negative transfer: previous experience conflicts with new situation

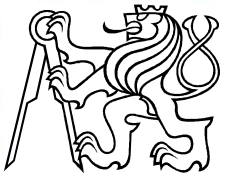
UI implementation issues – Architecture of a System





Interface layers / logical components

- linguistic: lexical/syntactic/semantic
- Seeheim
- Arch/Slinky

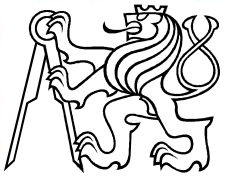


Monolithic Reference Architecture

Problem with monolihic architecture

- Difficult to modify/maintain
- Every single change is propagated to other parts of the system
- When this problem became an urgent one?
- Solution:

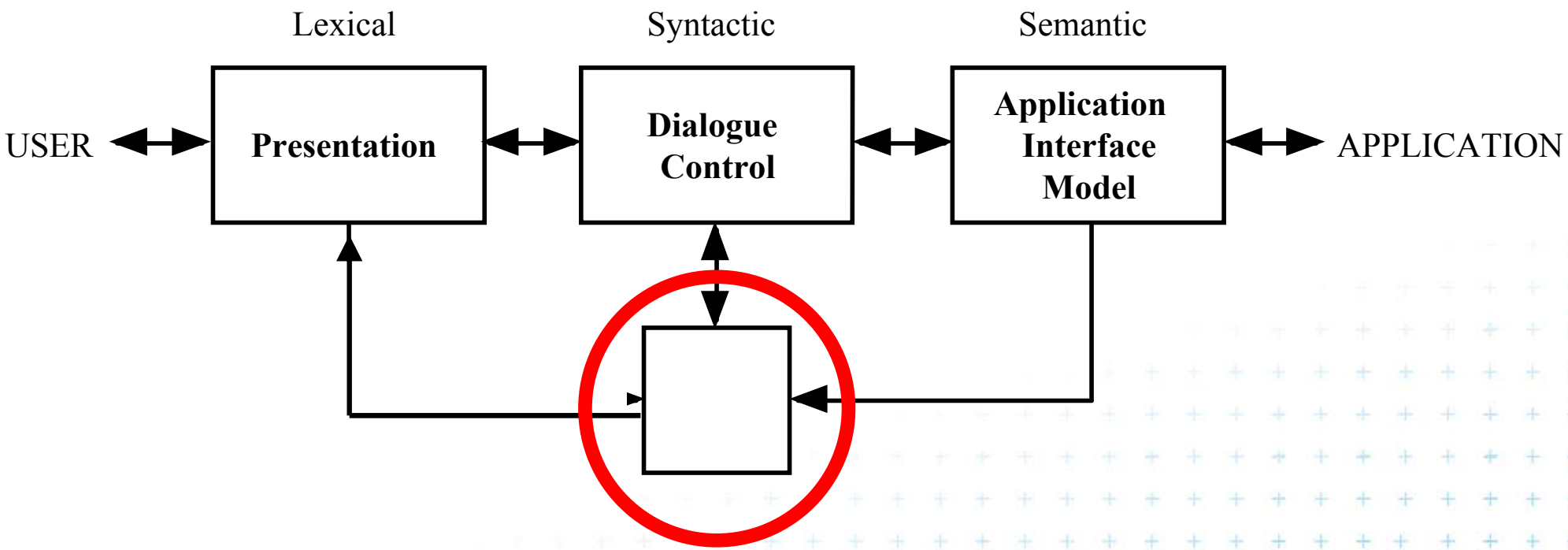
  - separation of UI part from application
- Existence of several models (no universal model exists)



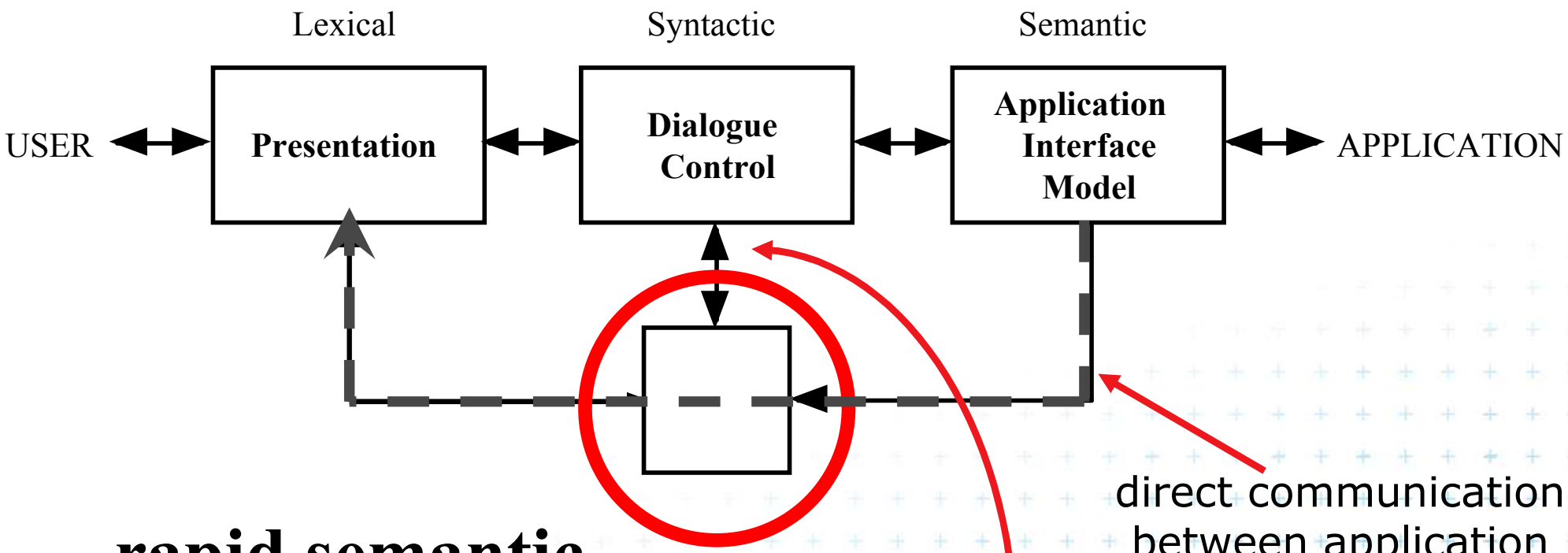
semantic feedback

- different kinds of feedback:
  - lexical – movement of mouse
  - syntactic – menu highlights
  - semantic – sum of numbers changes
- semantic feedback often slower
  - use rapid lexical/syntactic feedback
- but may need rapid semantic feedback
  - freehand drawing
  - highlight trash can or folder when file dragged

Seeheim model



the bypass/switch



rapid semantic feedback

direct communication between application and presentation

but regulated by dialogue control

DCGI

conceptual vs. implementation

Seeheim

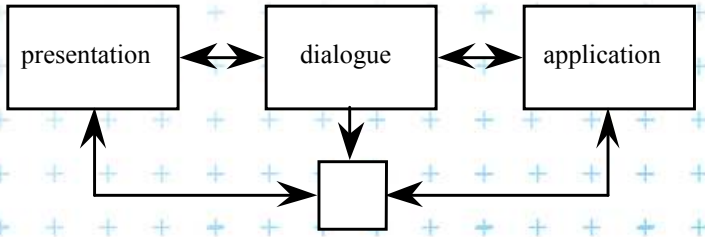
- arose out of implementation experience
- but principal contribution is conceptual
- concepts part of ‘normal’ UI language

… because of Seeheim …

… we think differently!

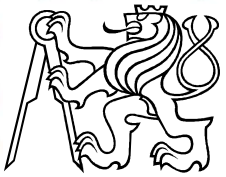
e.g. the lower box, the switch

- needed for implementation
- but not conceptual

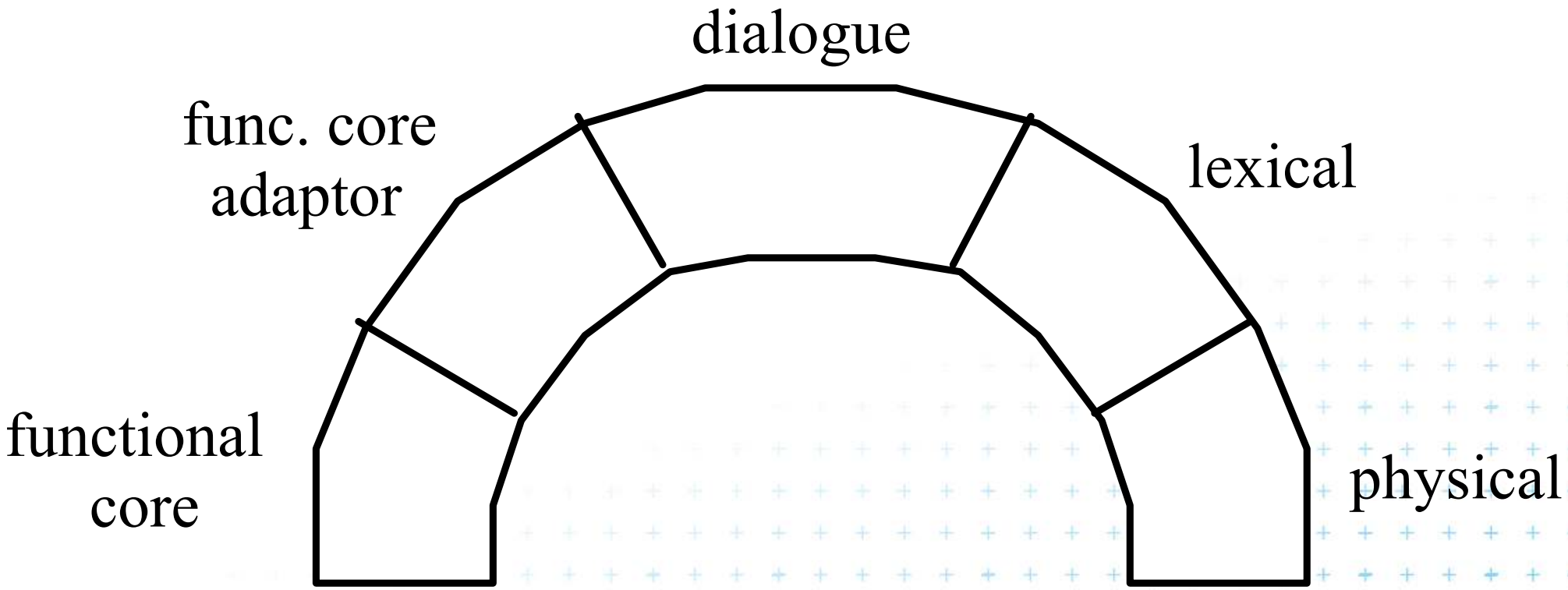


Seeheim problem

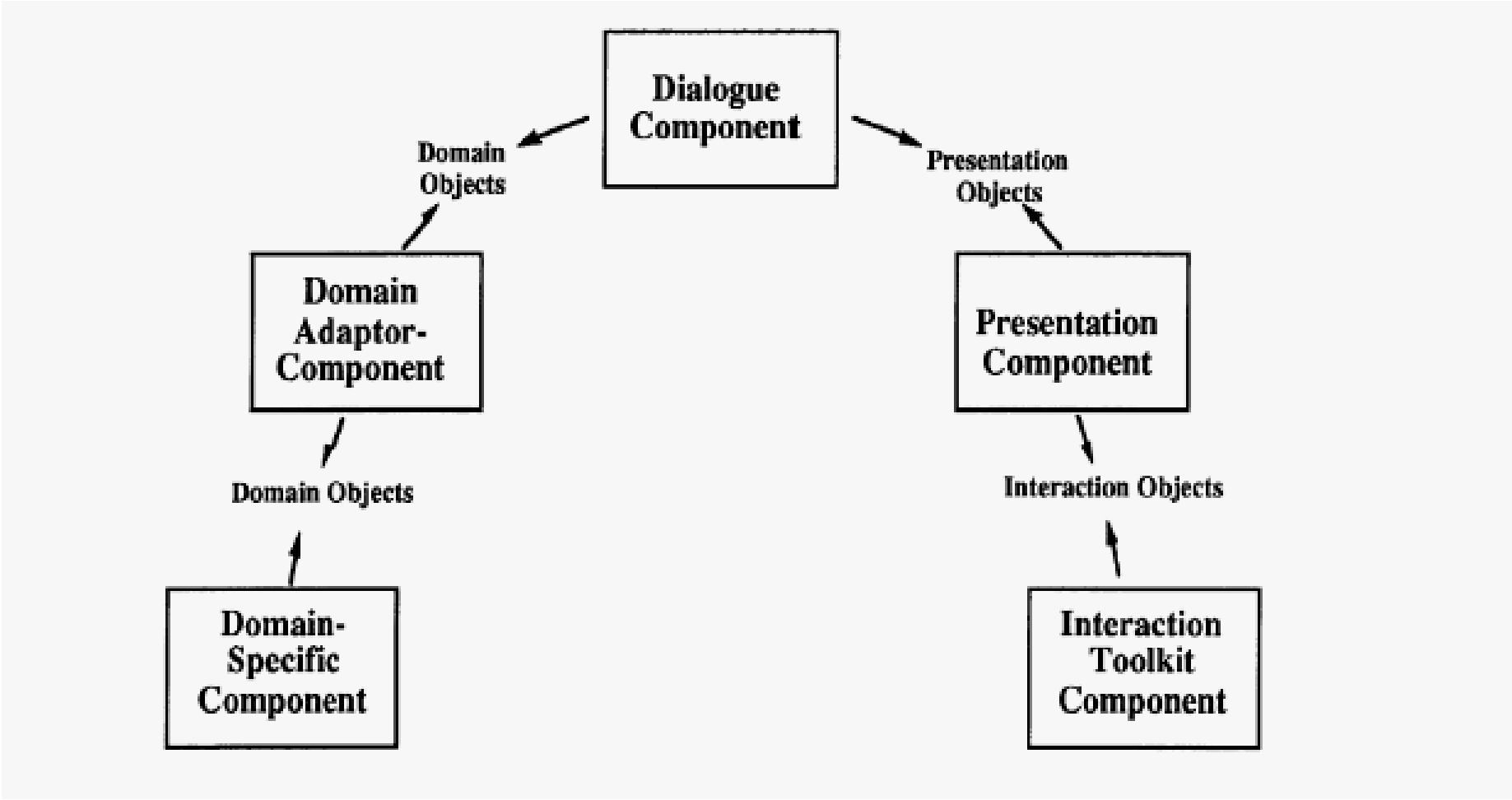
- Low granularity
- Result: problem with software modification



more layers!

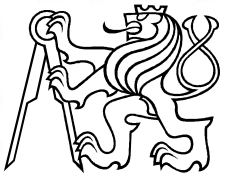

dialogue
func. core
adaptor
lexical
functional
core
physical

Arch model

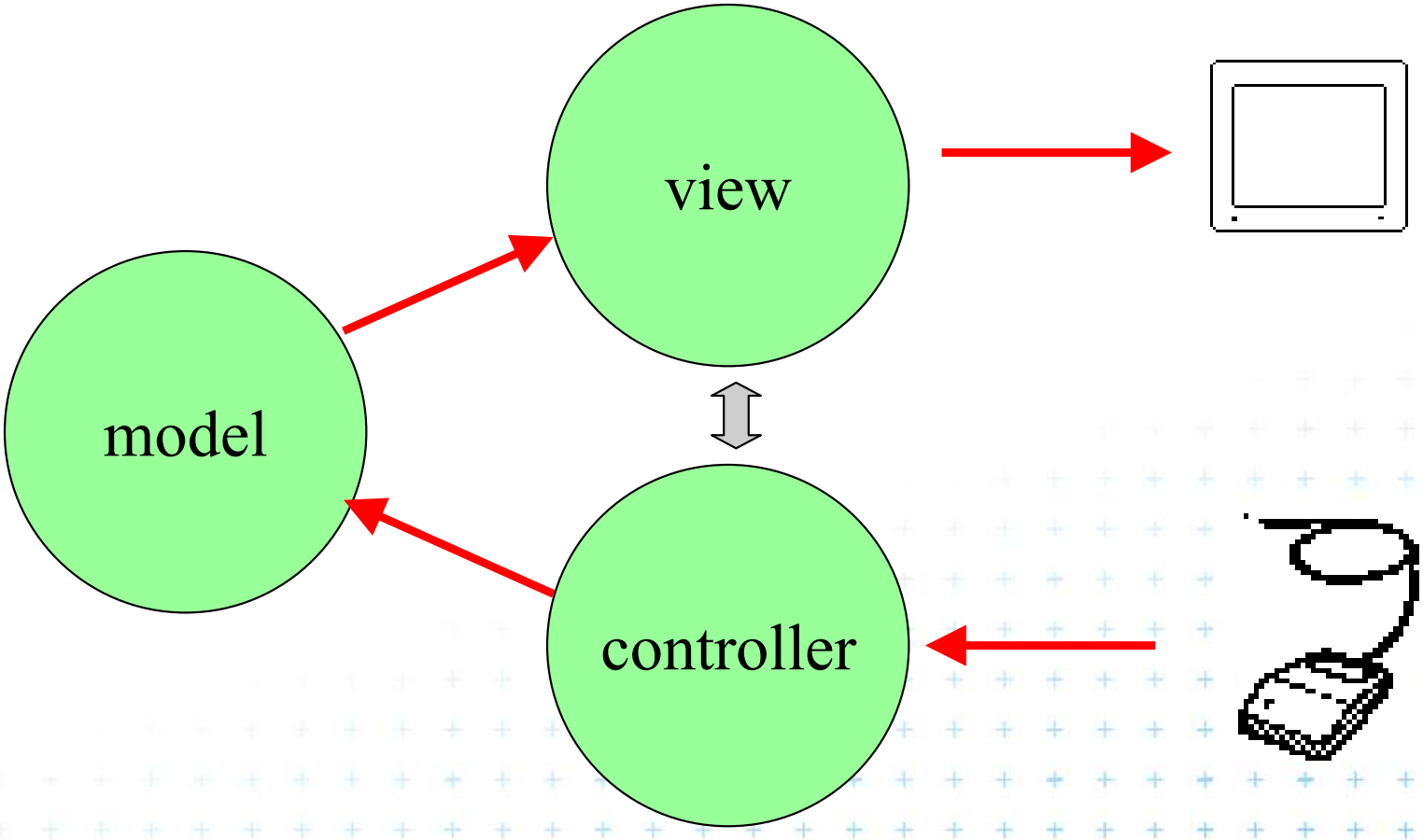

Dialogue Component
Domain Objects
Presentation Objects
Domain Adaptor-Component
Presentation Component
Domain Objects
Interaction Objects
Domain-Specific Component
Interaction Toolkit Component

Architecture: monolithic vs. components

- Seeheim has big components
- often easier to use smaller ones
  - esp. if using object-oriented toolkits
- Smalltalk used MVC – model–view–controller
  - model – internal logical state of component
  - view – how it is rendered on screen
  - controller – processes user input



MVC model - view - controller



MVC issues

- MVC is largely pipeline model:

  input → control → model → view → output

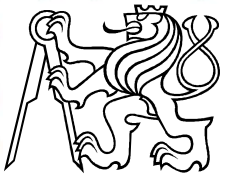
- but in graphical interface
  - input only has meaning in relation to output

  e.g. mouse click
  - need to know *what* was clicked
  - controller has to decide what to do with click
  - but view knows what is shown where!
- in practice controller ‘talks’ to view
  - separation not complete

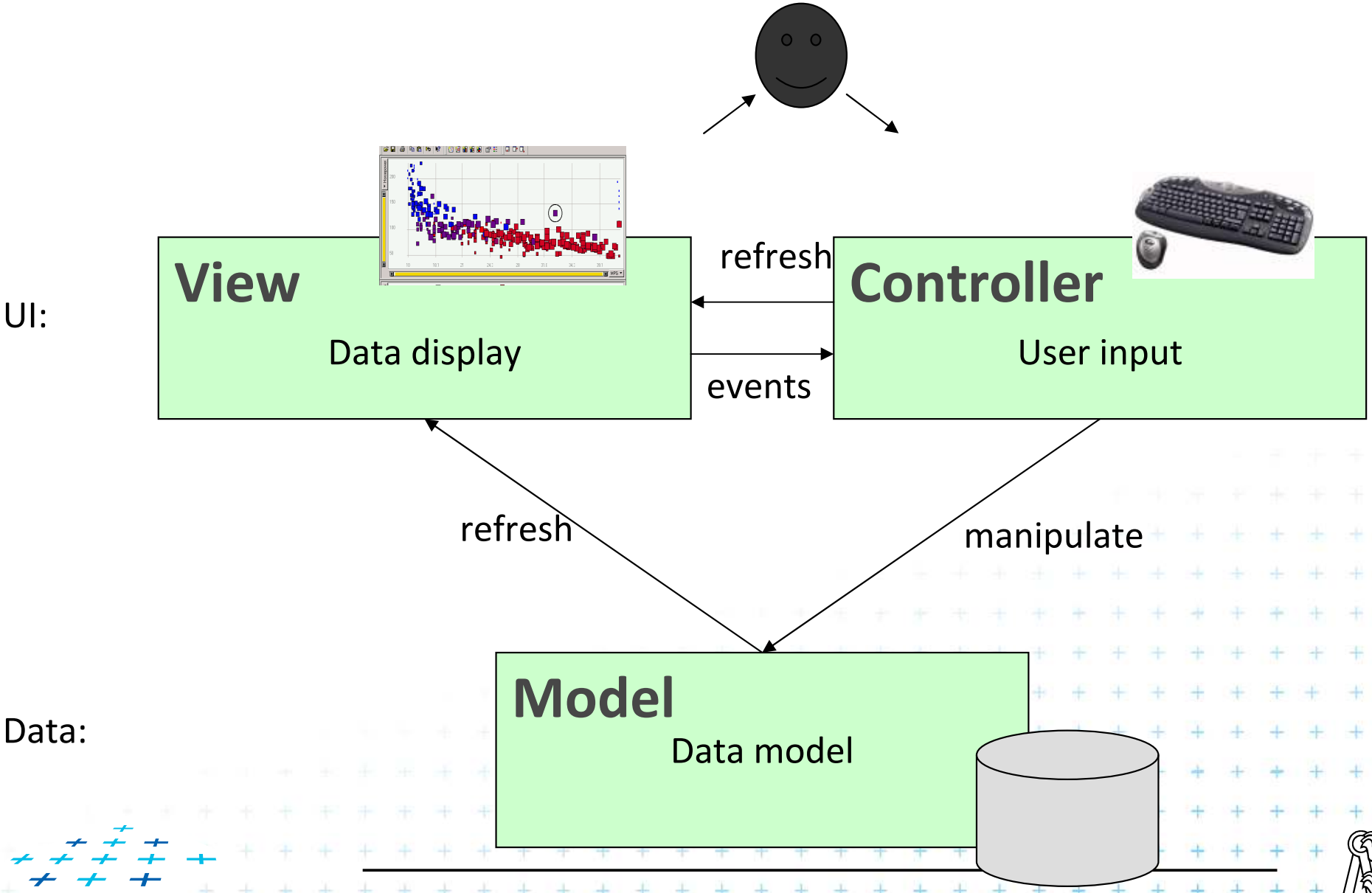
MVC

- The application is divided in three parts: the model, views and controllers
- The model is normally implemented so that it is independent of the views and the controllers
- There can be several views and controllers linked to one model

Example?

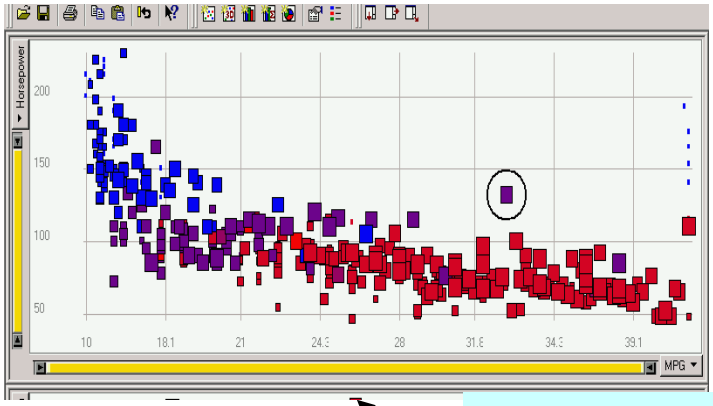


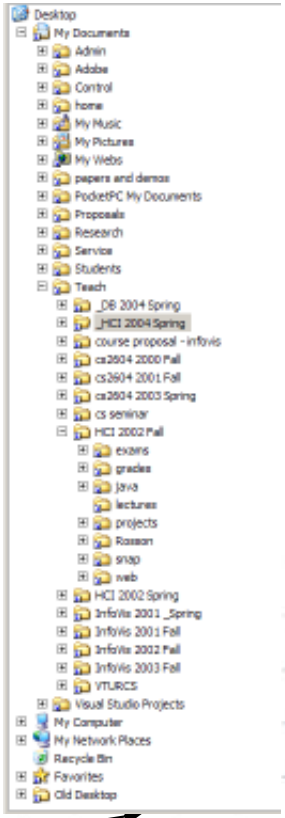
Model-View-Controller (MVC)



DCGI

Multiple Views





View

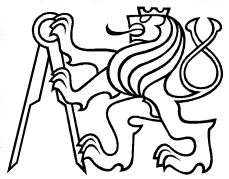
Controller

View

Controller

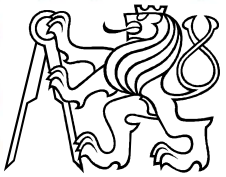
Model

DCGI



Implementation issues

- Techniques for dialogue controller
  - menu networks
  - grammar notations
  - declarative languages
  - graphical specification
  - state transition diagrams
  - event languages
  - constraints

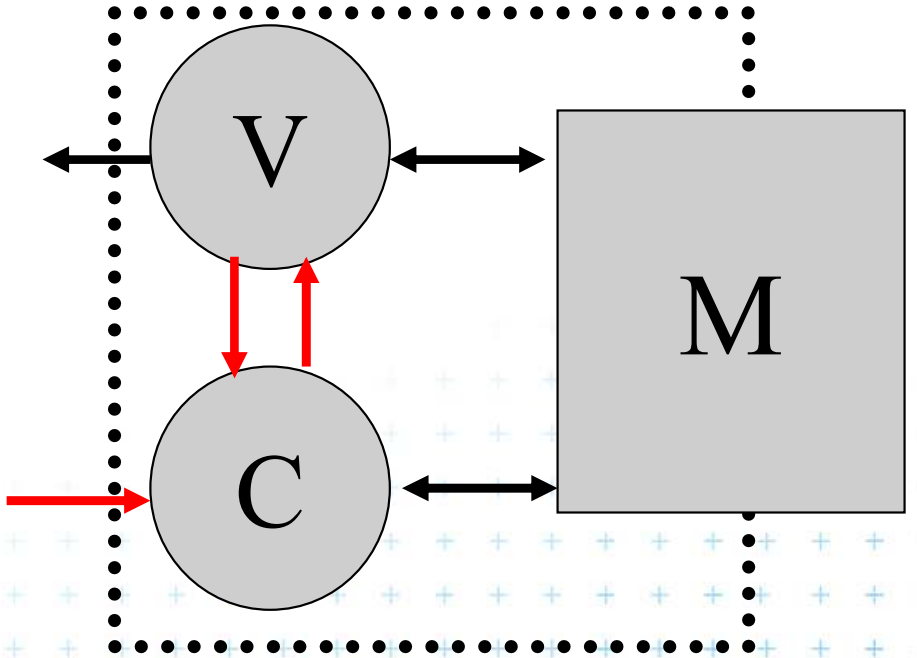


Controller Examples

- Textual commands
- Mouse (point and click) commands

MVC Dynamics

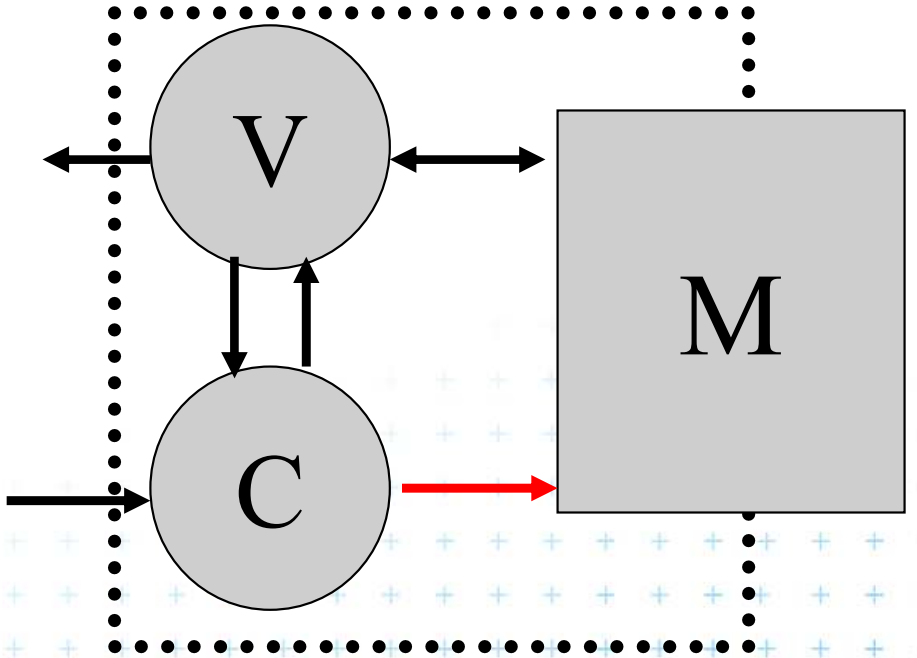
- 1. User input event routed by Window System to appropriate Controller.
- 2. Controller may require View to “pick” object of focus for event.



MVC Dynamics

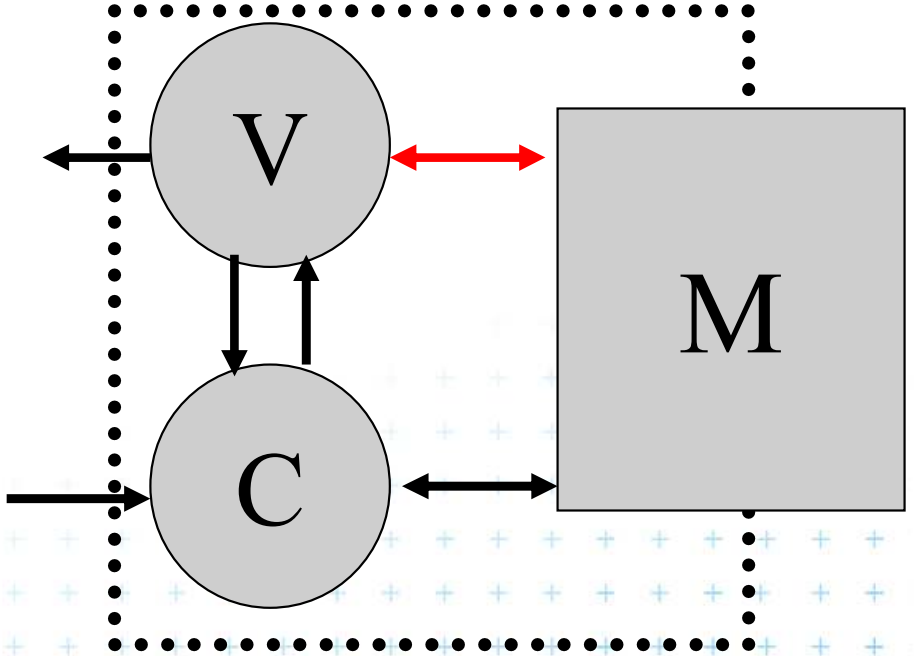
- 3. Controller requests method of Model to change its state.

- 4. Model changes its internal state



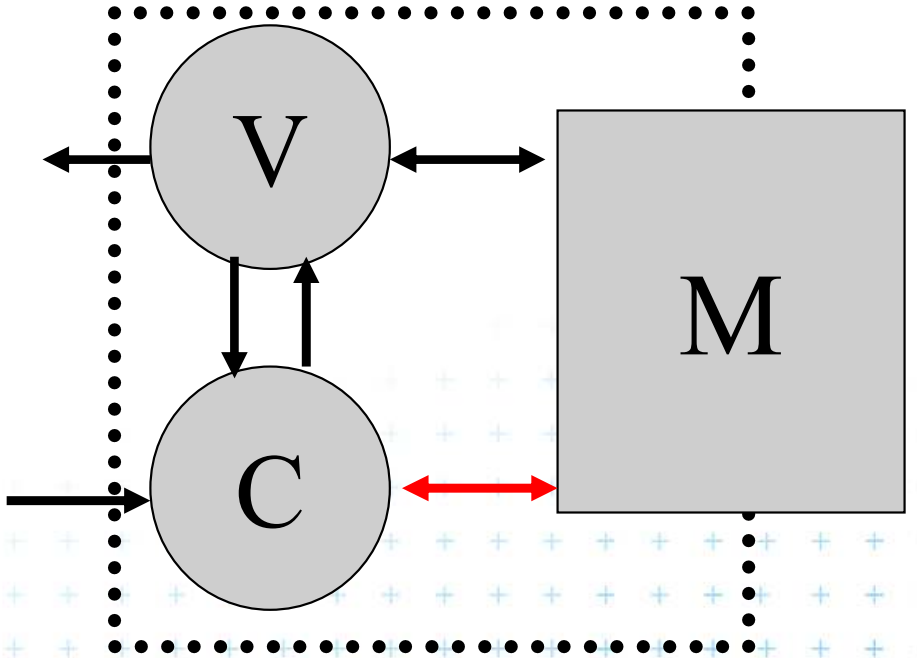
MVC Dynamics

- 5. Model notifies all dependent Views that data has changed.
- 6. View requests from Model current data values.



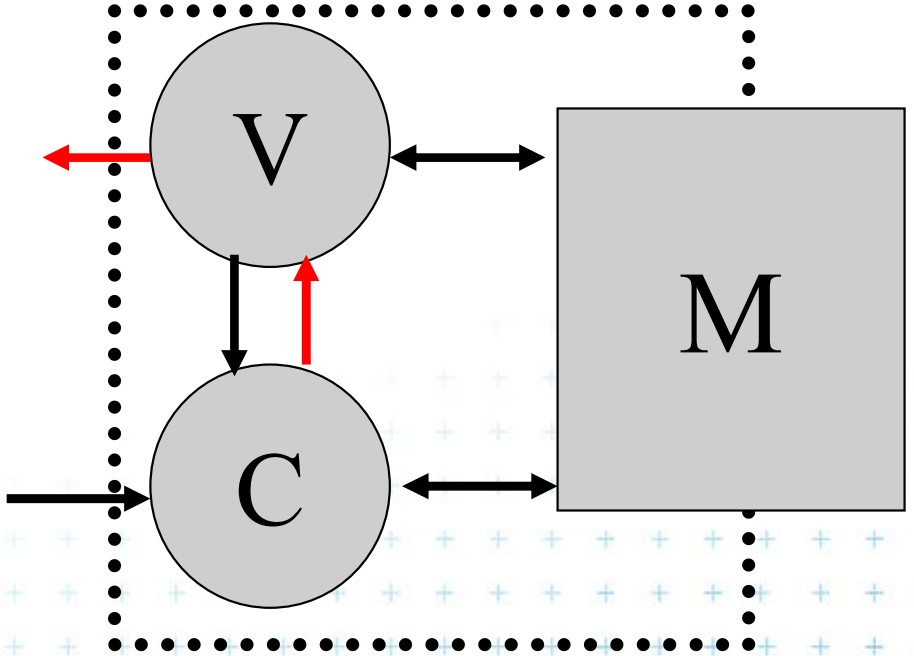
MVC Dynamics

- 7. Model notifies all dependent Controllers that data has changed.
- 8. Controller requests from Model current data values.



MVC Dynamics

- 9. Controller informs View if elements are disabled.
- 10. View requests redraw



Thank you for your attention