



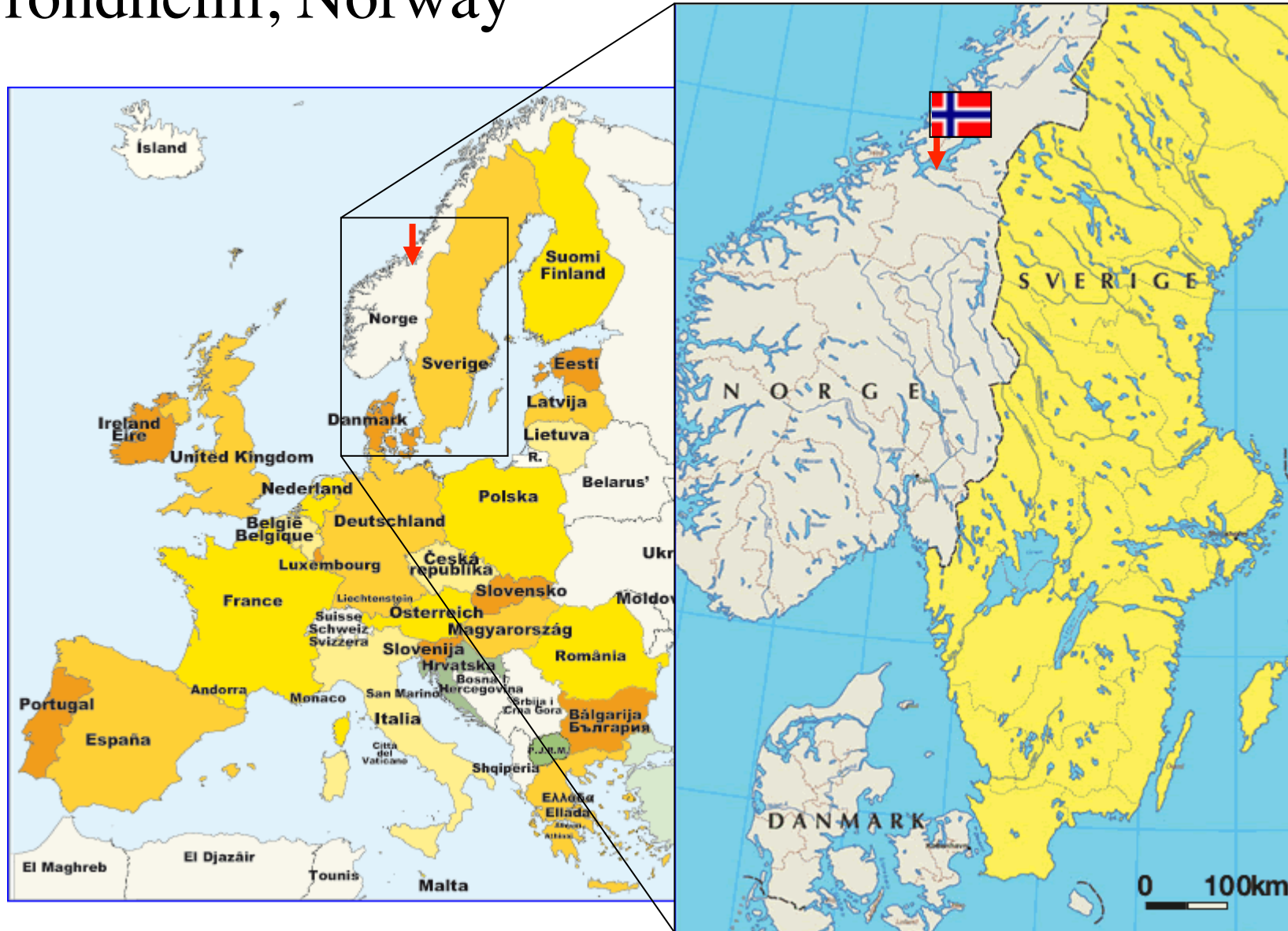
# NTNU

Norwegian University of  
Science and Technology

## User interface modeling Model-based UI design

Hallvard Trætteberg, Associate Professor  
Dept. of Computer and Information Sciences  
Norwegian Univ. of Science and Technology

# Trondheim, Norway

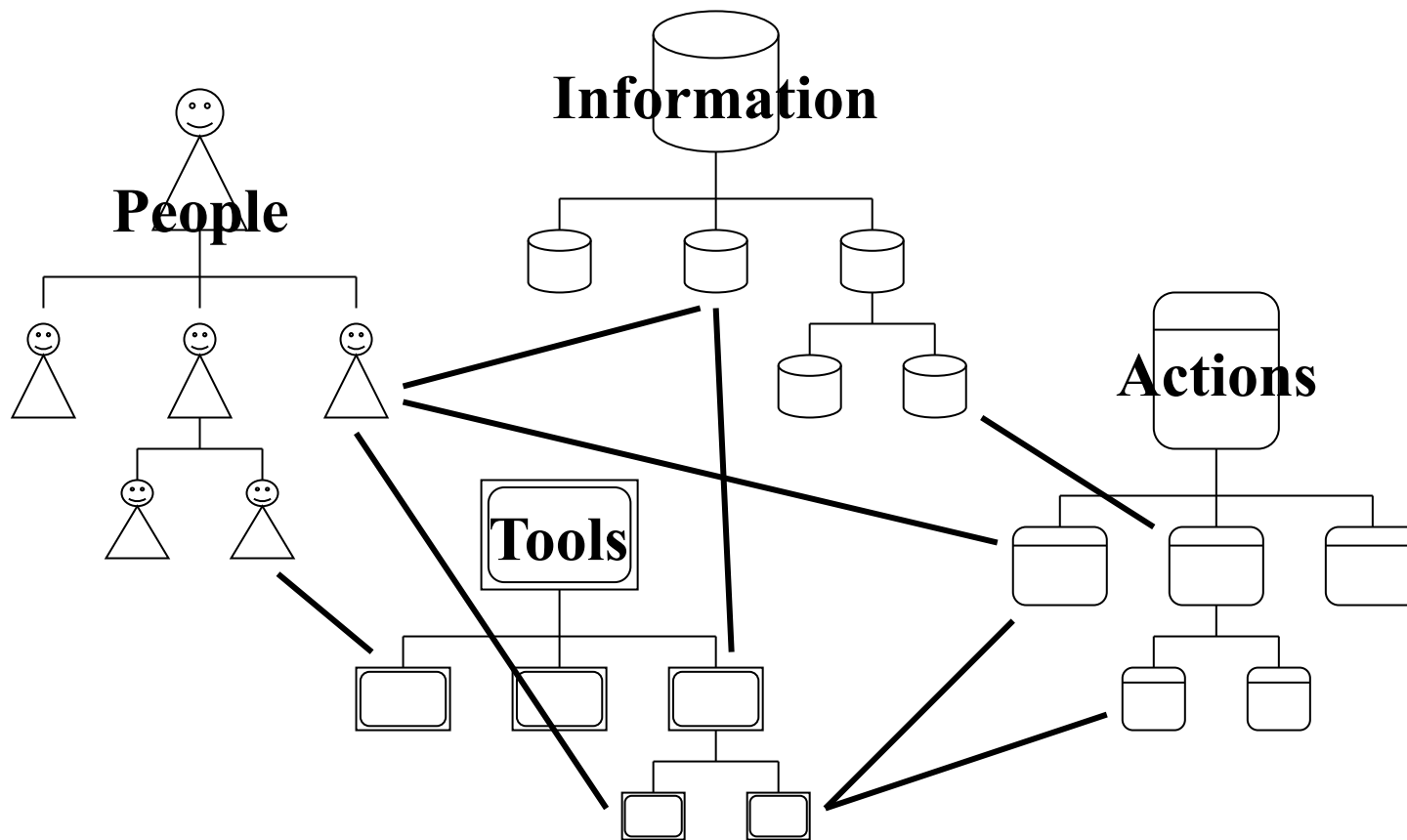


# User interface modeling

## Model-based UI design

1. Background and framework for classifying design representations
2. (Examples of) Models for development of UIs
3. Diamodl
4. ptui – ptolemy-based tool for development of UIs

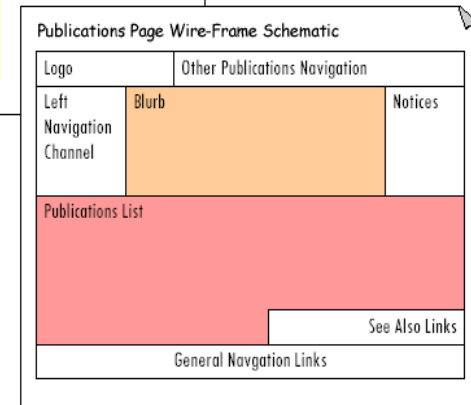
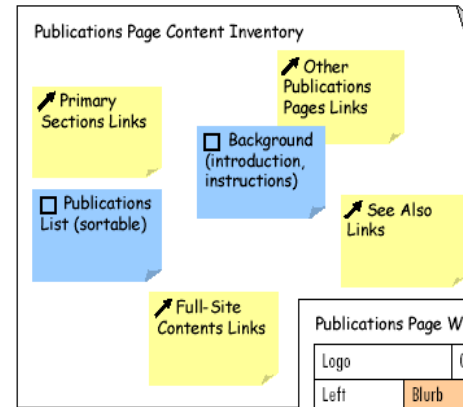
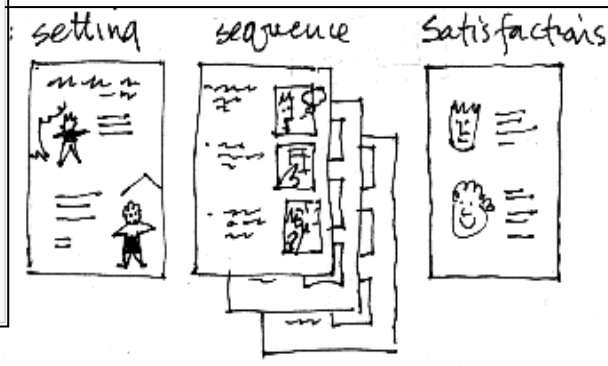
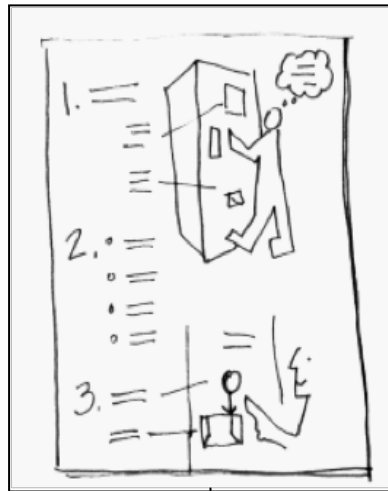
# Many models capture our knowledge about the world



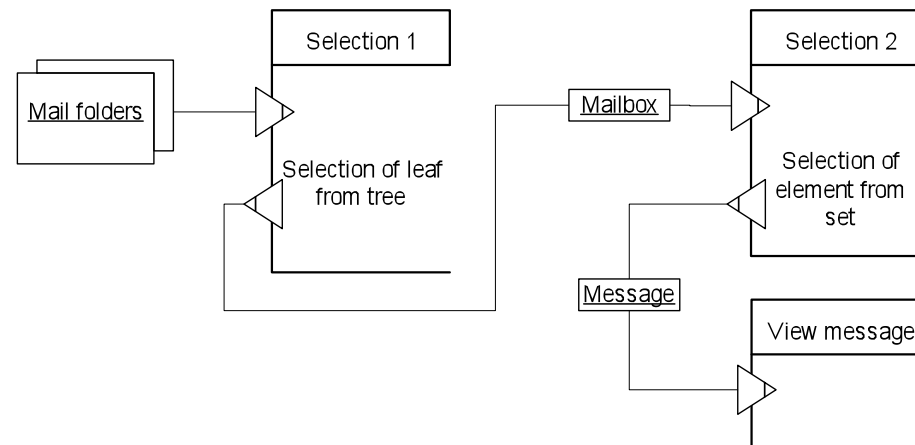
# Roles representations play

- Semantic
  - accurately and completely capture knowledge
- Communicative
  - support communication among designer and end-users
- Constructive
  - stimulate, guide and constrain further design
- Analytic
  - support interpretation and evaluation
- Engineers and designers focus on different roles

# From informal representations...

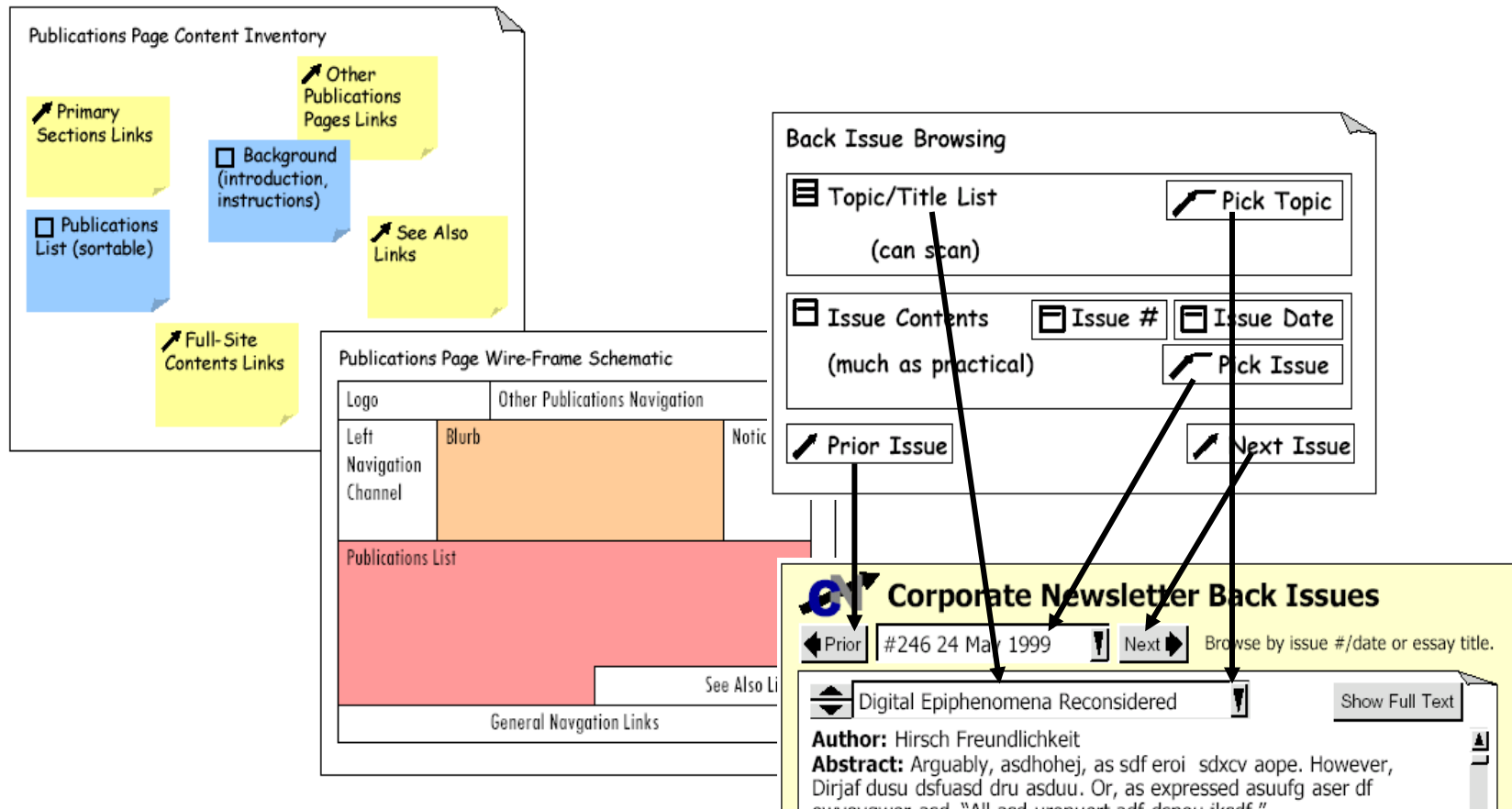


... to models



# Canonical Abstract Prototypes

## [Constantine] – semi-formal sketching



# What aspects of a UI do we want to capture?

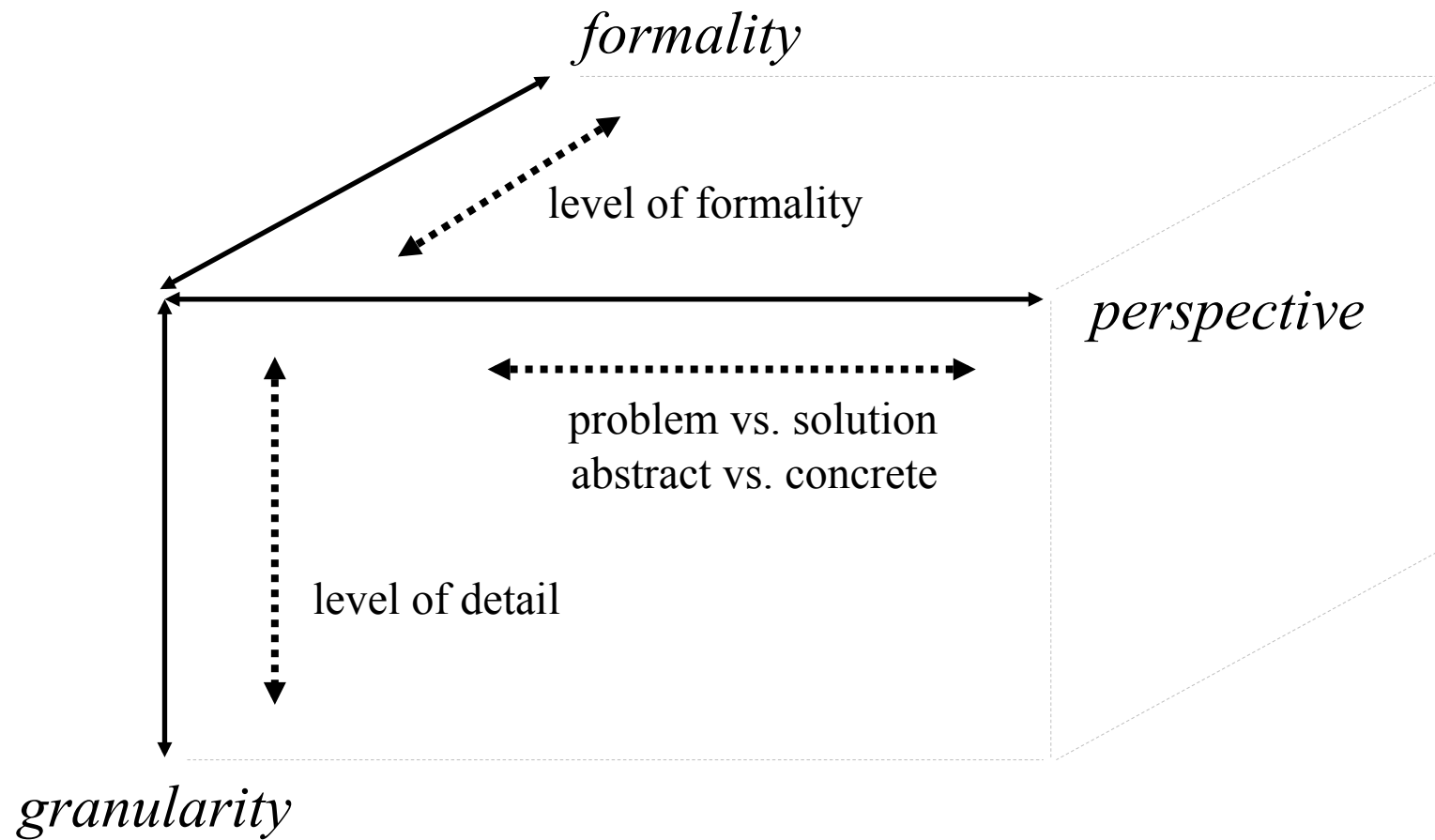
- **Structure**
  - hierarchical structure of interaction elements
- **Information**
  - what information is accessible in which parts of the UI
  - what is the relationship between information in various parts of the UI
- **Behavior**
  - when are the various interaction elements active
  - how are changes in the UI triggered by the user
- **Style**
  - non-functional aspects, like layout, use of colors, fonts etc.



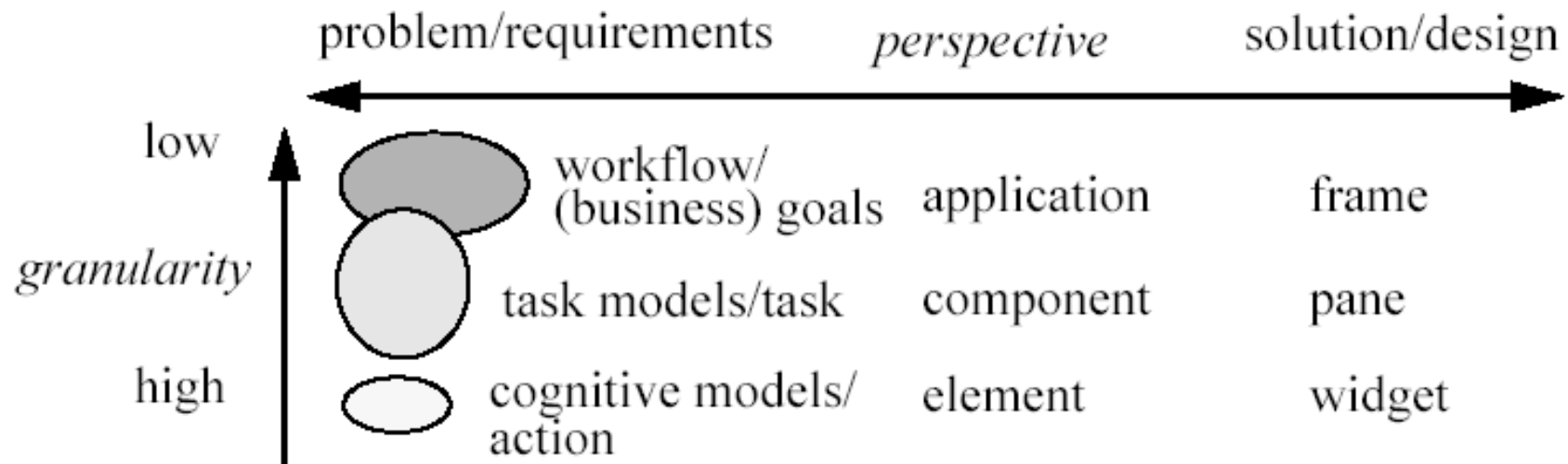
# Four phases of MBDUI

1. Model and generate
  - model your domain
  - generate UI from canned knowledge and pre-compiled rules
2. Computer-Aided Design of UI
  - abstract models/representations of UI
  - explicitly represent design knowledge
  - model editors and tools for applying design knowledge
3. Task-based UI design
  - can't design usable interfaces without knowing the user and tasks
  - base design of UI on task model (goals, structure and dependencies)
4. Contextualizing and adapting design models
  - focus on context of use
  - target multiple devices

# Design representation classification framework

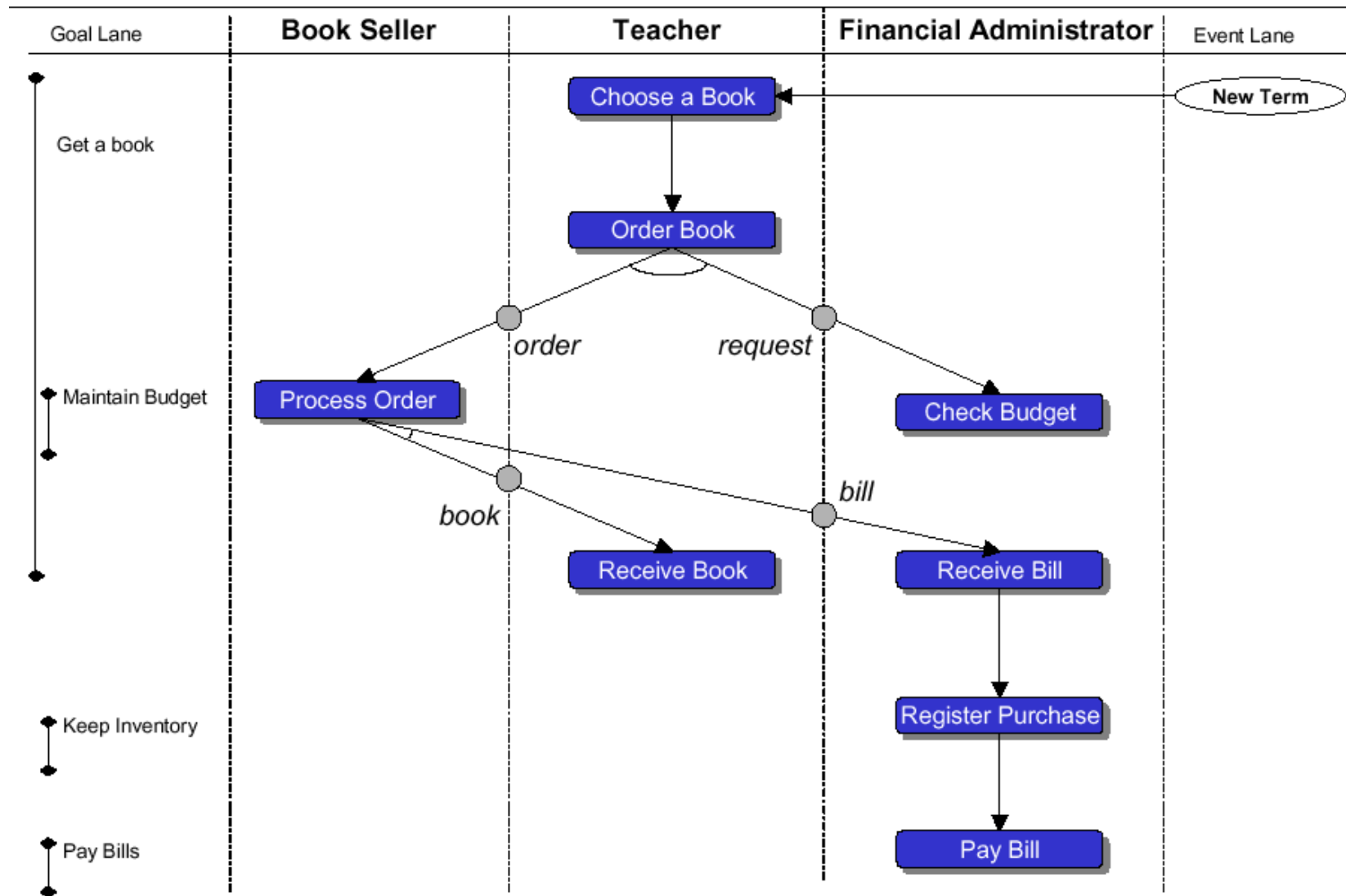


# Perspective and granularity dimensions

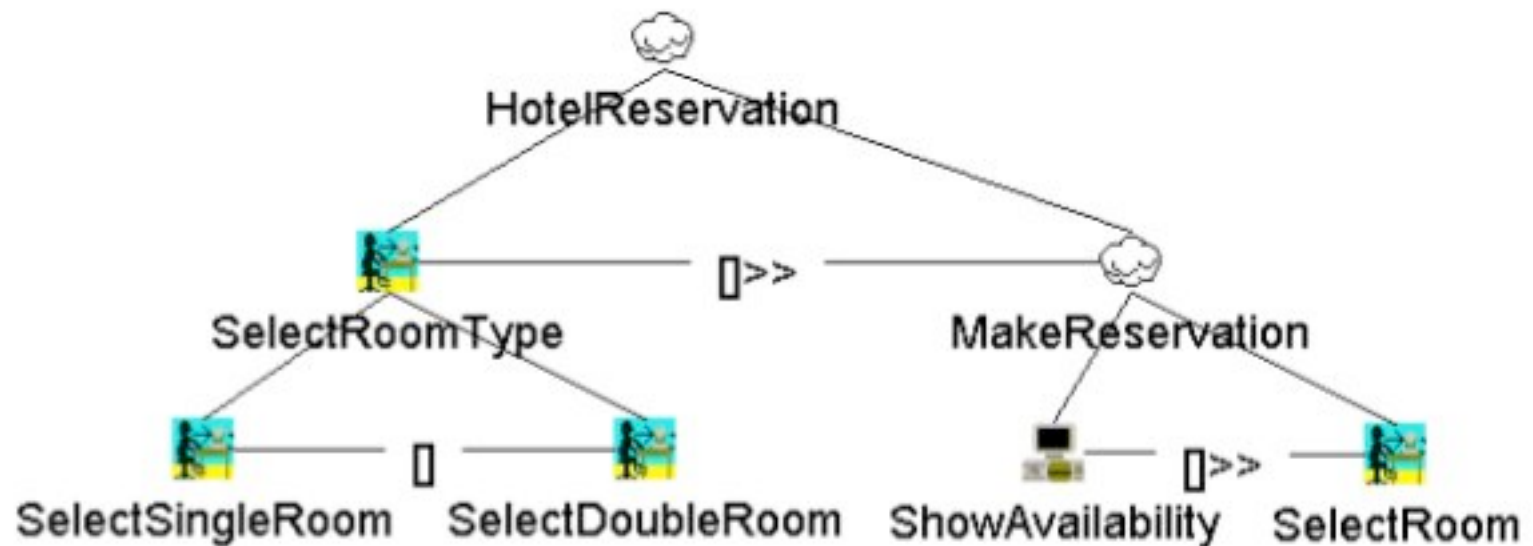


**Figure 12.** The level/granularity dimension interpreted across perspectives

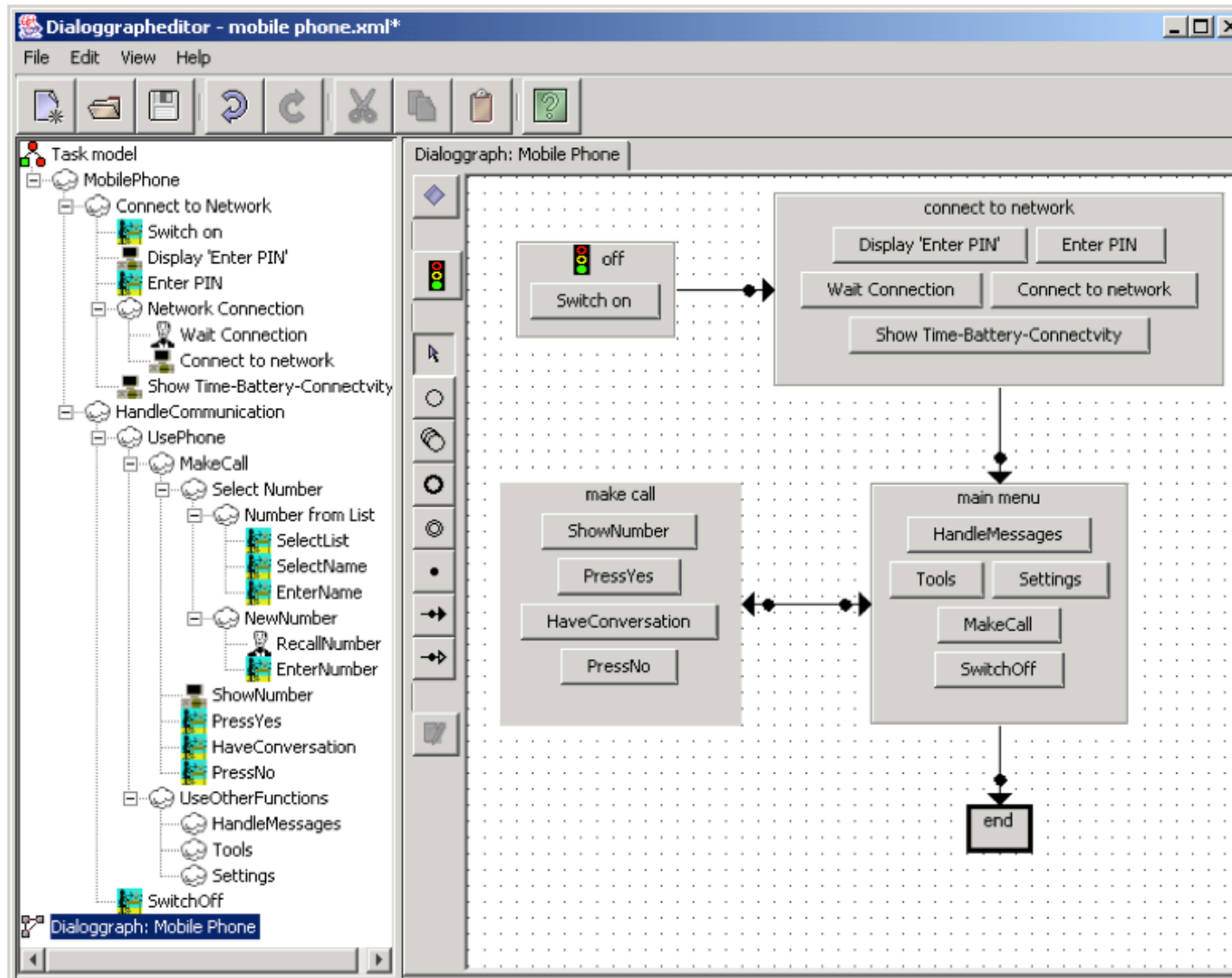
# Dutch [van der Veer] - task models as activity charts



# ConcurTaskTrees [Paternò] – task hierarchies with temporal operators



# Dialog graphs [Forbrig] – Relating tasks to dialog



# UsiXML [Vanderdonckt] – A family of XML-based notations for UI elements

The image displays two screenshots of the Eclipse IDE interface, illustrating the UsiXML project structure and the generation of UI elements.

**Top Screenshot:** Shows the Eclipse IDE with the Package Explorer on the left. The project structure is as follows:

- UsiXMLProject
  - src
    - JRE System Library [JavaSE-1.6]
  - model
    - sample.aui
    - sample.auidiag
    - sample.xwt

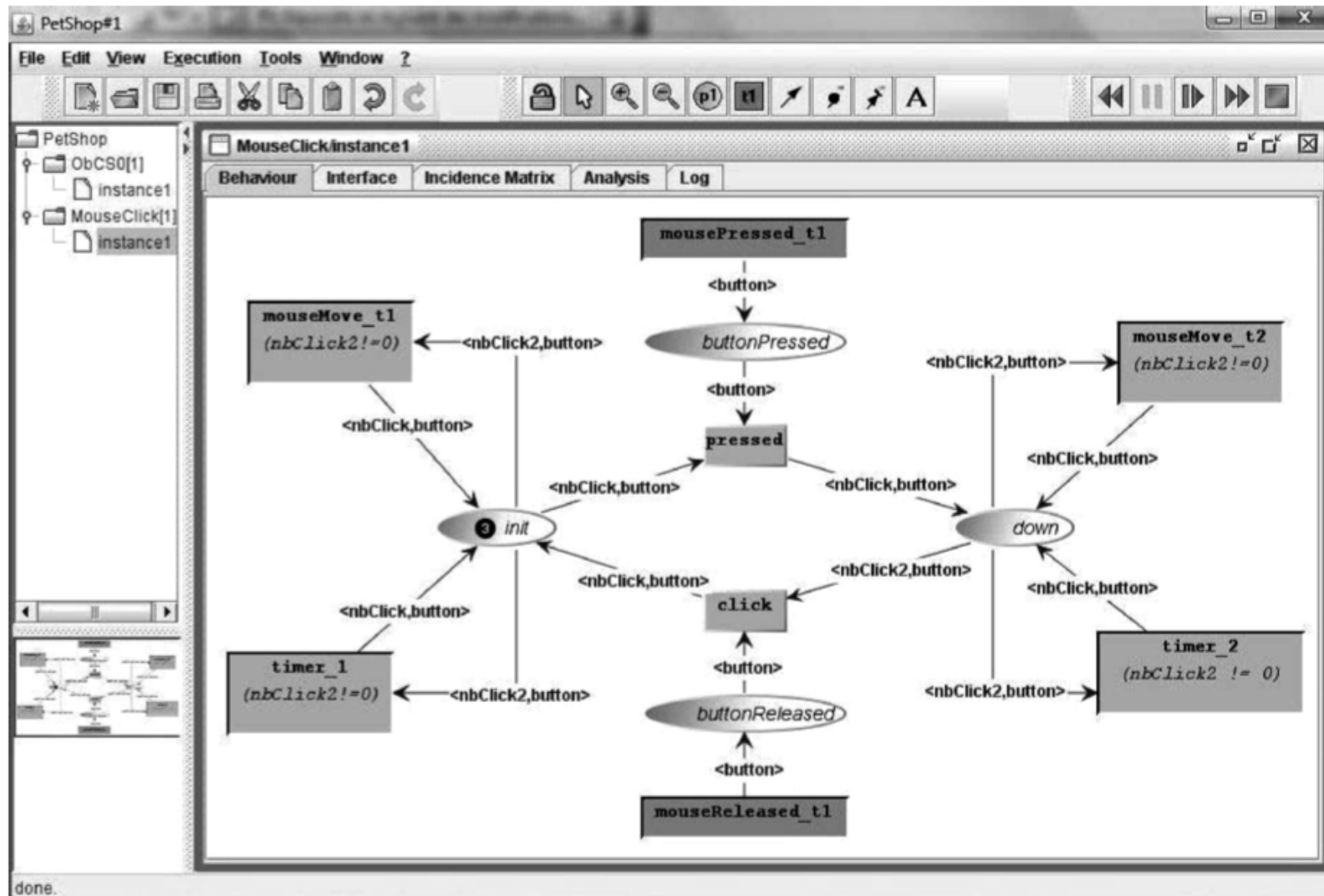
The main editor window displays the `sample.auidiag` file, which is a Registry form. The form contains the following elements:

- ShowEnterName
- GetUsername
- ShowEnterPassword
- GetPassword
- ShowSelectRole
- SelectRole
- Submit

**Bottom Screenshot:** Shows the Eclipse IDE with the Package Explorer on the left. The project structure is the same as in the top screenshot. The main editor window displays the `sample.xwt` file, which is an XML-based notation for the UI elements. The XML code is as follows:

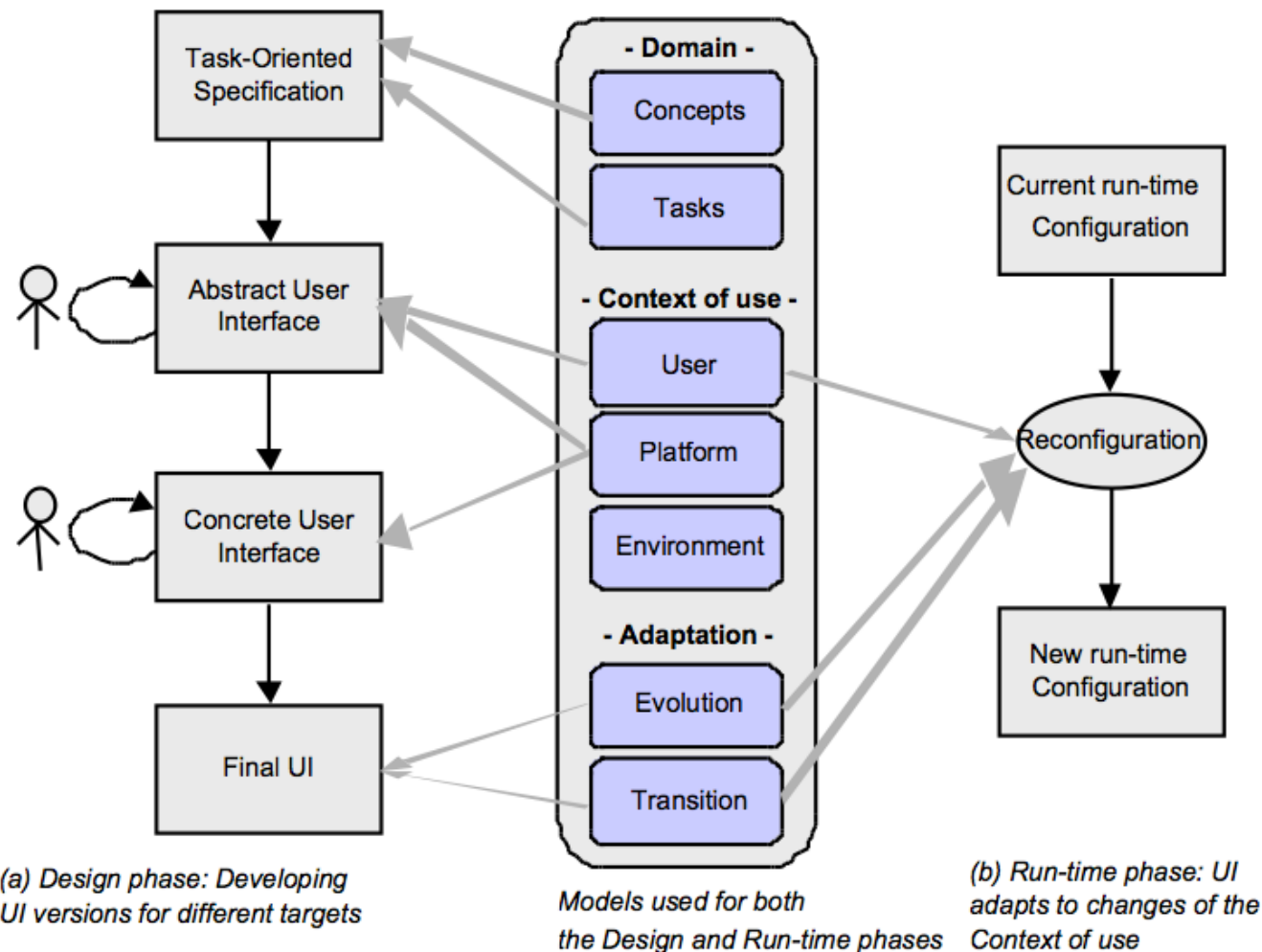
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Shell xmlns="http://www.eclipse.org/xwt/presentation">
  <Label text="Enter username:"/>
  <Text text="&lt;enter username&gt;">
  <Label text="Enter password:"/>
  <Text text="*****"/>
  <Label text="Select role:"/>
  <Combo text="Admin"/>
  <Button text="Login"/>
  <Shell.Layout>
    <GridLayout numColumns="1"/>
  </Shell.Layout>
</Shell>
```

# Pet shop [Palanque] – Modeling safety critical UIs with ICO PetriNets





# Cameleon framework – targeting multiple devices

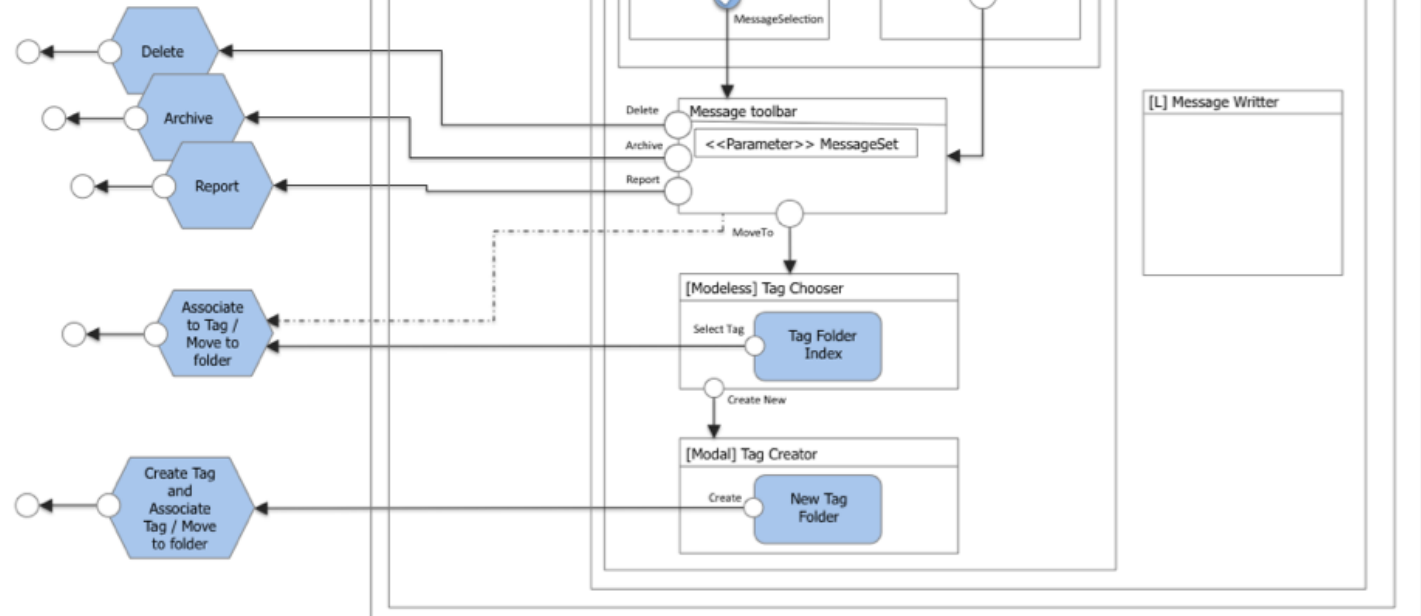


# Lots of pragmatic approaches (read: non-academic and useful)

- XML-based formats for describing user interface layout and style
  - XHTML (W3C) , XAML (Microsoft), JavaFX (Oracle), XUL (Mozilla)
  - template languages for web pages
- DSLs
  - Ecore-based: Eclipse 4's workbench model, Wazaabi
  - Xtext-based: APPlause, MOBL, Agentry
- Application modeling
  - Esito's Genova – business applications for the desktop and web
  - WebRatio - business applications for the web
- Standardization
  - WebML
  - IFML (in progress)
  - Model-Based User Interfaces (MBUI) Working Group

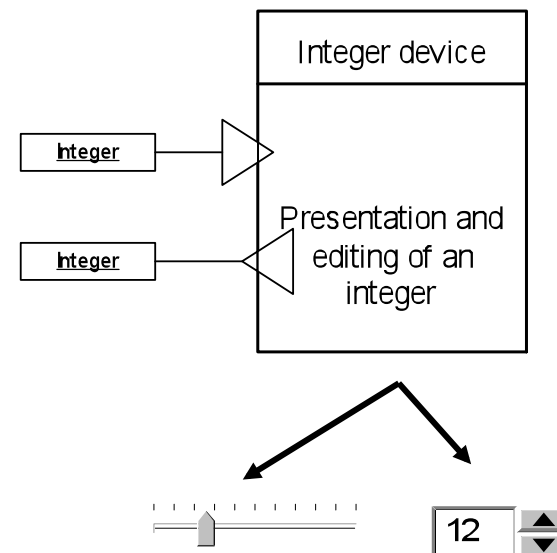
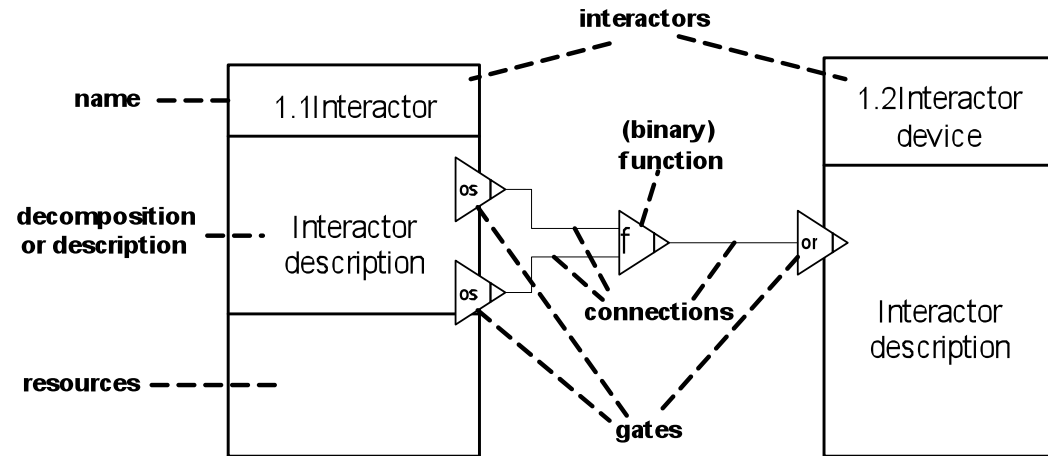
# IFML – Interaction Flow Modeling Language

- OMG RFP
- Proposal by WebRatio++
- Abstract UI model
- Functional units and view containers
- Dataflow and control/activation signals



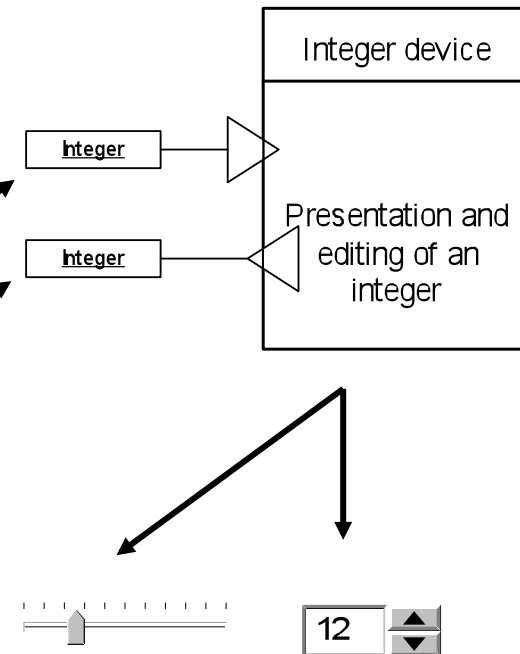
# Dialog modelling with DiaMODL

- Based on Pisa interactors and Harel's Statecharts
  - interactors, gates and connections
  - hierarchical states
  - transitions, events/actions, conditions
- Abstraction of IO function
- Composition in terms of
  - interactor structure
  - state hierarchy (and, or)



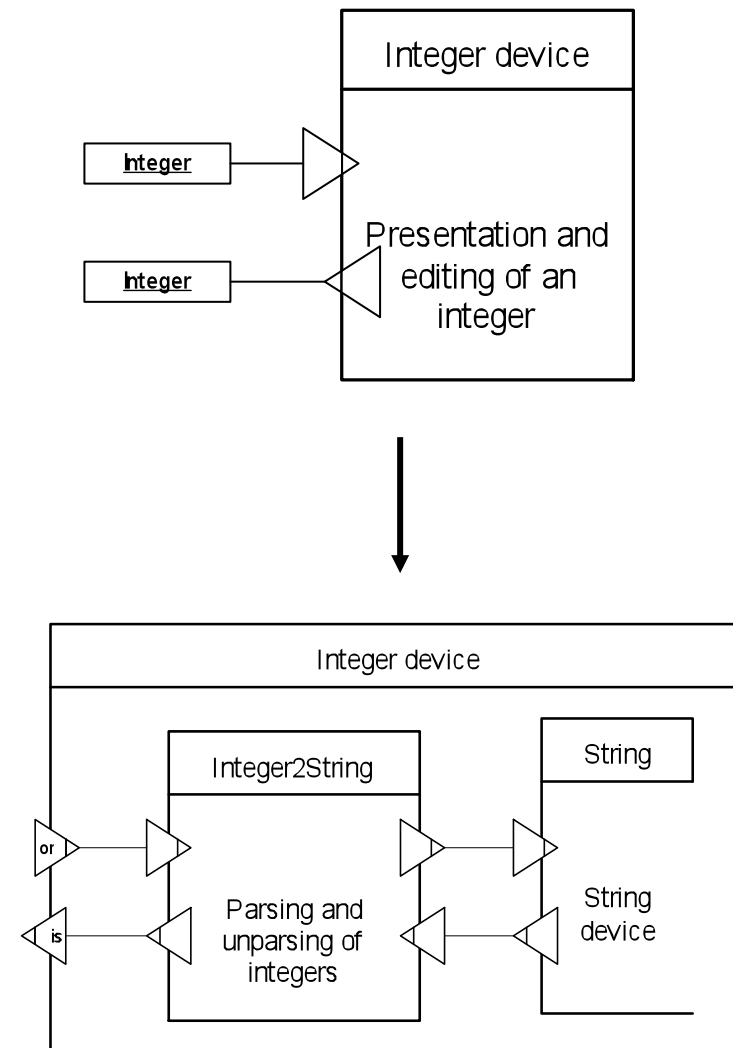
# Generic interactor abstraction

- Notation for generic input and output components
- Dataflow-oriented
- Interactor mediates information in two directions:
  - output: system to user
  - input: user to system



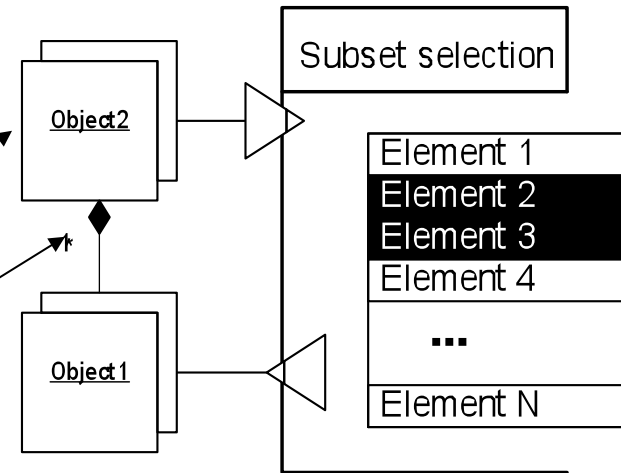
# Scalable notation

- Specification of concrete interaction object's functionality
  - output and input interface
- Description of construction of concrete interaction objects
  - composition of sub-interactors
  - string input combined with parsing and unparsing
- Same abstract description, many alternatives

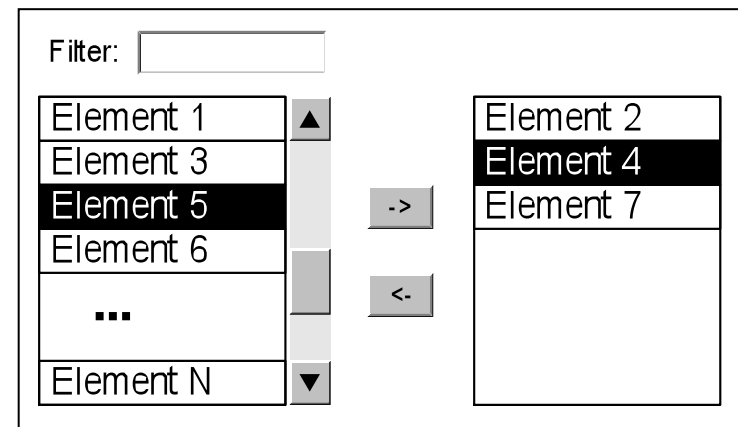


# More complex interaction objects

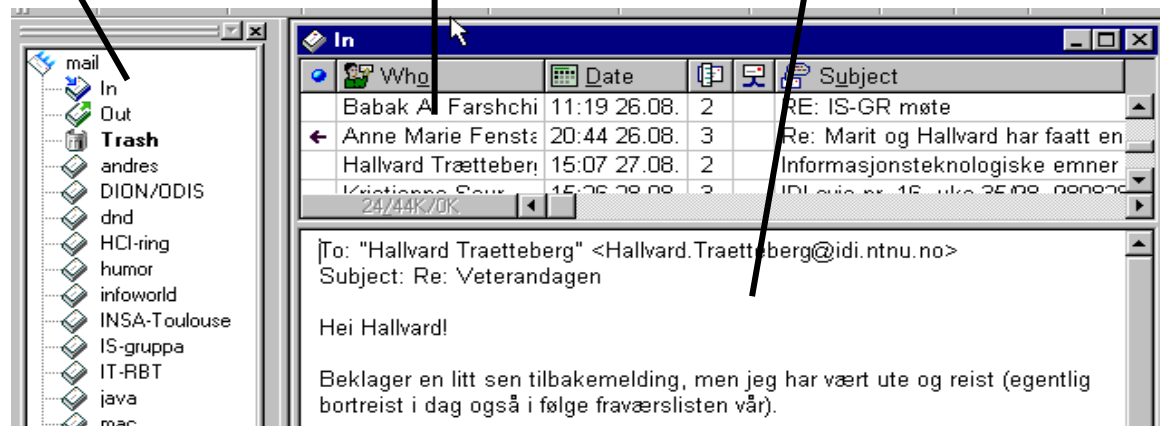
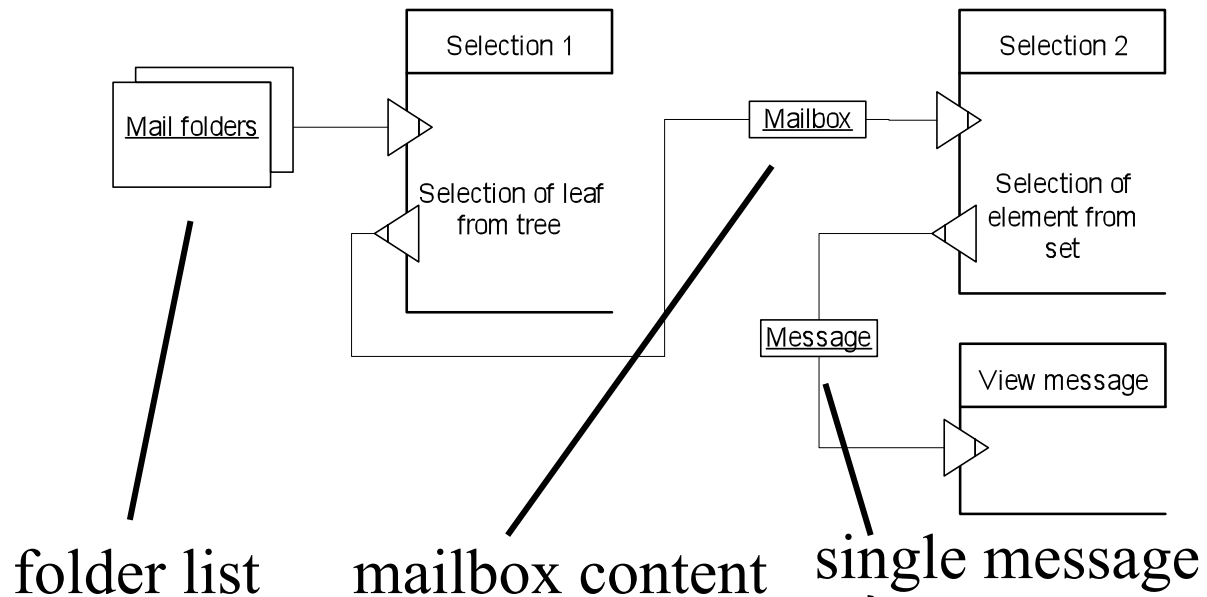
- Functionality defined in terms of configuration of domain objects
- Utilise power of domain modelling language
  - Output: set
  - Input: subset



Alternative implementation

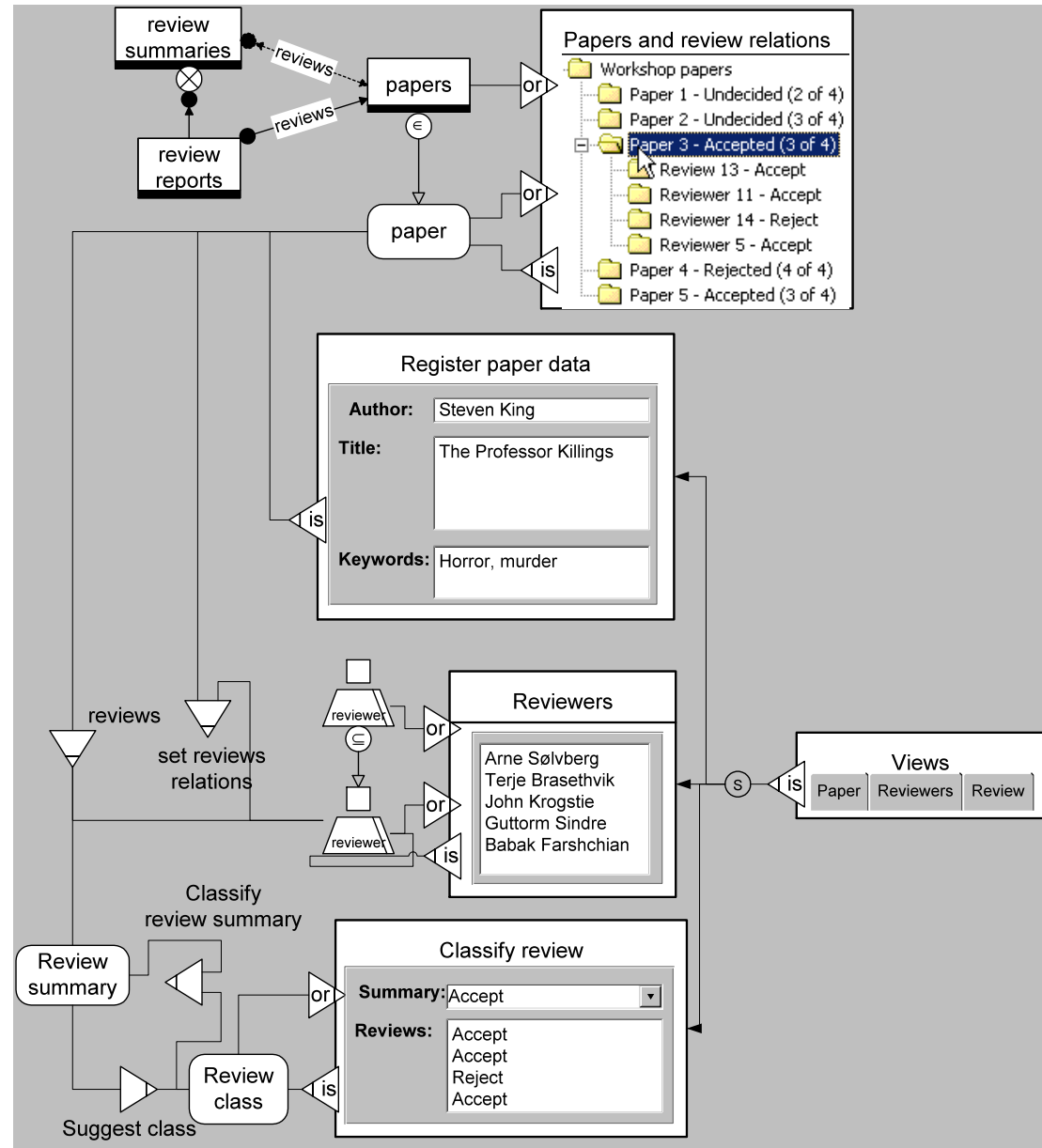


# Configuration of larger elements



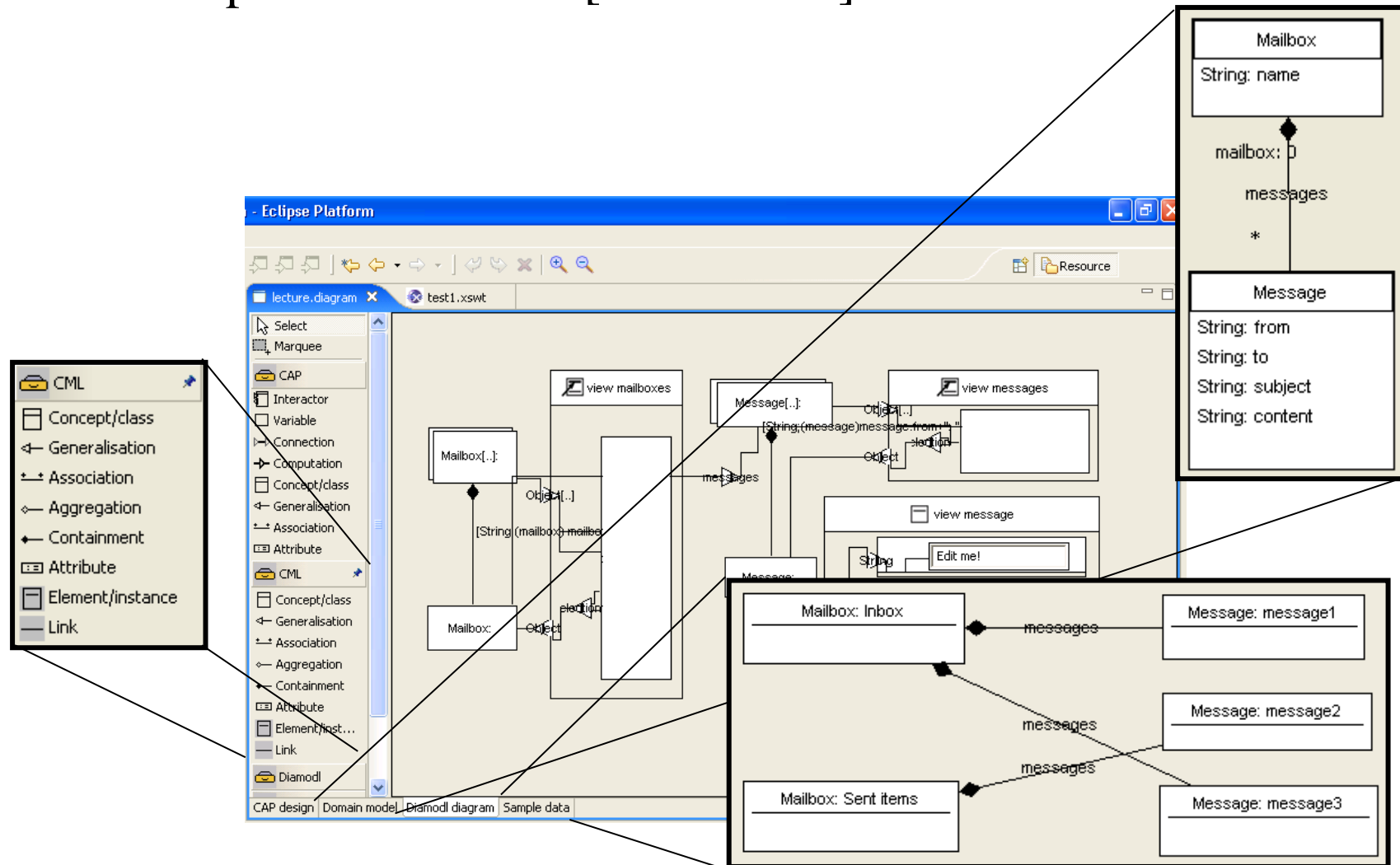


# Interactor-based GUI-builder



# Integrating domain and dialog modeling

- Eclipse-based editor [CADUI'06]

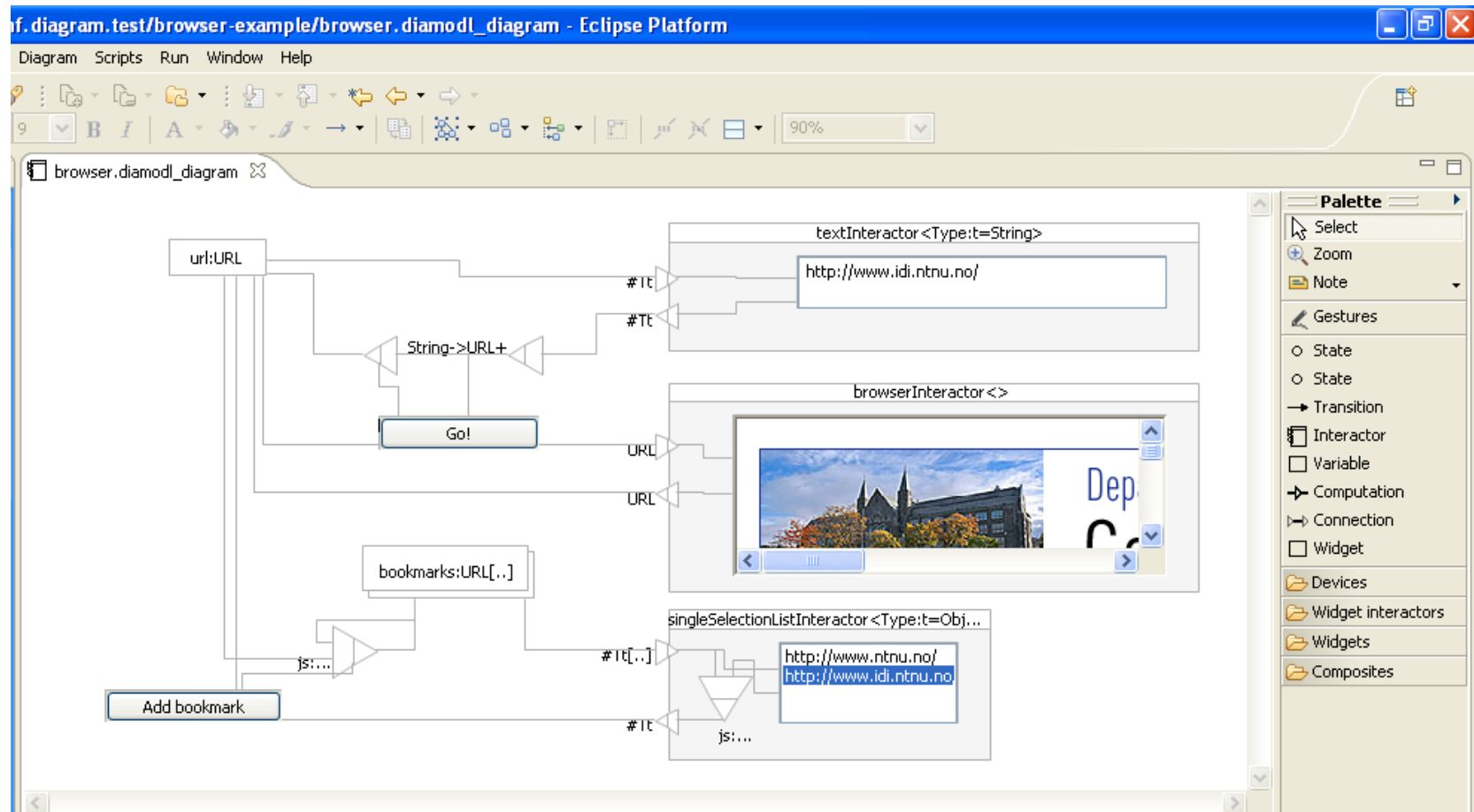


# Prototyping with Diamodl

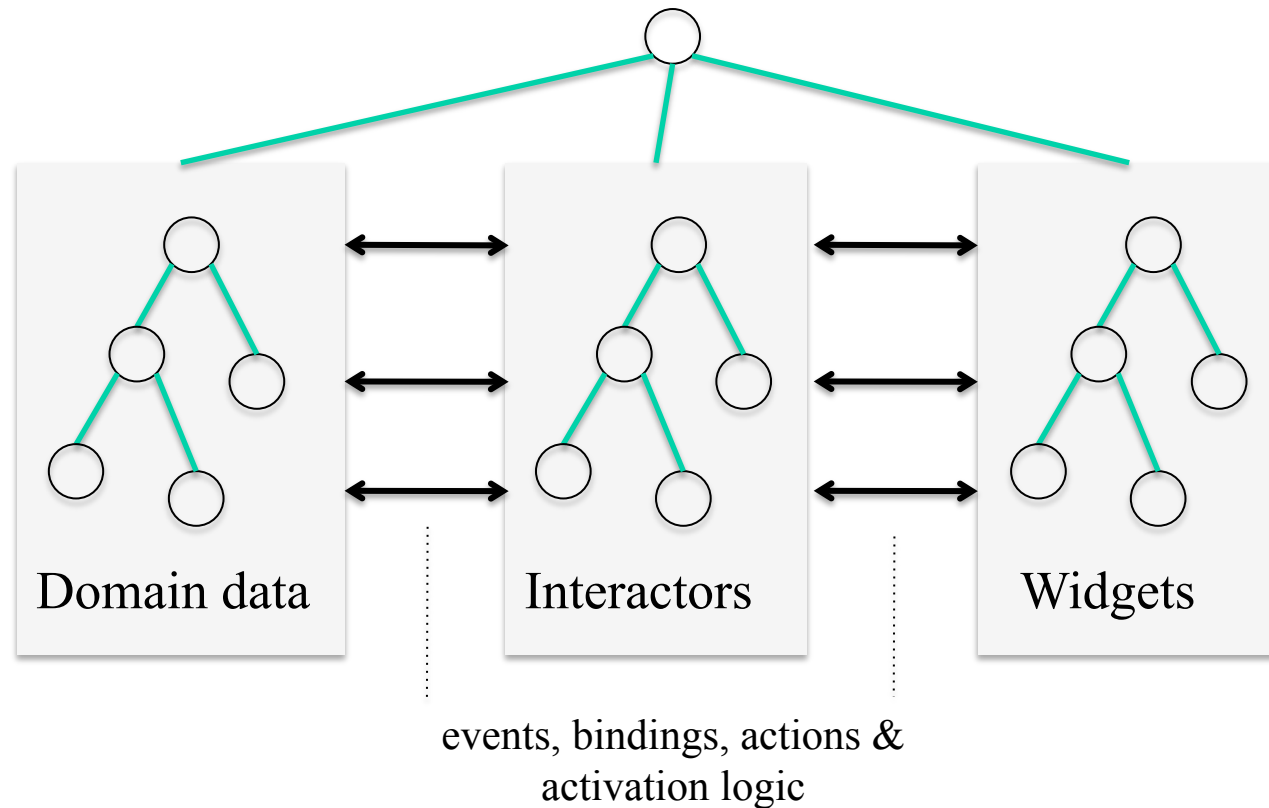
The screenshot displays the Eclipse IDE interface with the following components:

- Window Title Bar:** Java - no.hal.diamodl.gmf.diagram.test/browser-example/browser.diamodl\_diagram - Eclipse Platform
- Menu Bar:** File, Edit, Diagram, Navigate, Search, Project, Scripts, Run, Window, Help
- Toolbar:** Standard Eclipse IDE icons for file operations and editing.
- Editor Tabs:** browser.diamodl\_diagram, library.diamodl\_diagram, calc1.xswt
- Left Panel (Outline):** A tree view showing 56 events, including interactions like <singleSelectionListInteract, >singleSelectionListInteract, and widget-related events like >uri#value and <uri#value.
- Diagram Area:** A UML-like diagram showing:
  - uri:URI** connected to **String** and **String->URI+**.
  - String** connected to **textInteractor<>** (containing **textWidget:text**) and **browserInteractor<>** (containing **browserWidget:bro...**).
  - String->URI+** connected to **goUriButton...**.
  - uri:URI** connected to **bookmarks:URI[...]**.
  - bookmarks:URI[...]** connected to **addBookmarksButton...**.
  - addBookmarksButton...** connected to **singleSelectionListInteractor <Type=Object>** (containing **listWidget:list**).
  - Various other connections involving **js:...** and **#Tt[...]** labels.
- Right Panel (Palette):** A palette with categories like Select, Zoom, Note, Gestures, State, Transition, Interactor, Variable, Computation, Connection, Widget, Devices, Widget interactors, Widgets, and Composites.
- Bottom Panel:** Includes tabs for Problems, Javadoc, Declaration, Error Log, Diamodl Runtime Xswt View, Properties, and Search. Below these is a browser window showing the URL <http://www.idi.ntnu.no/> and a page header for the Department of Information Technology.

# Prototyping with Diamodl



# Application architecture



- The whole runtime state is captured as coordinated graphs of data
- The widget hierarchy is continuously rendered on a device

# Rendering widgets

- Ecore model of toolkit, with instances rendered in Eclipse view

The screenshot displays the Eclipse IDE interface. The top part shows the Ecore model tree for 'TabFolder.xmi'. The tree structure is as follows:

- platform:/resource/org.eclipse.gmt.emfacade.rwt.builder.test/examples/TabFolder.xmi
  - Tab Folder
    - Tab Item Text
    - Tab Item Buttons
      - Composite
        - Button
        - Button
        - Button (highlighted)
        - Button
        - Button
        - Fill Layout HORIZONTAL
    - Tab Item Lists
      - Composite
    - Tab Item Browser
      - Composite

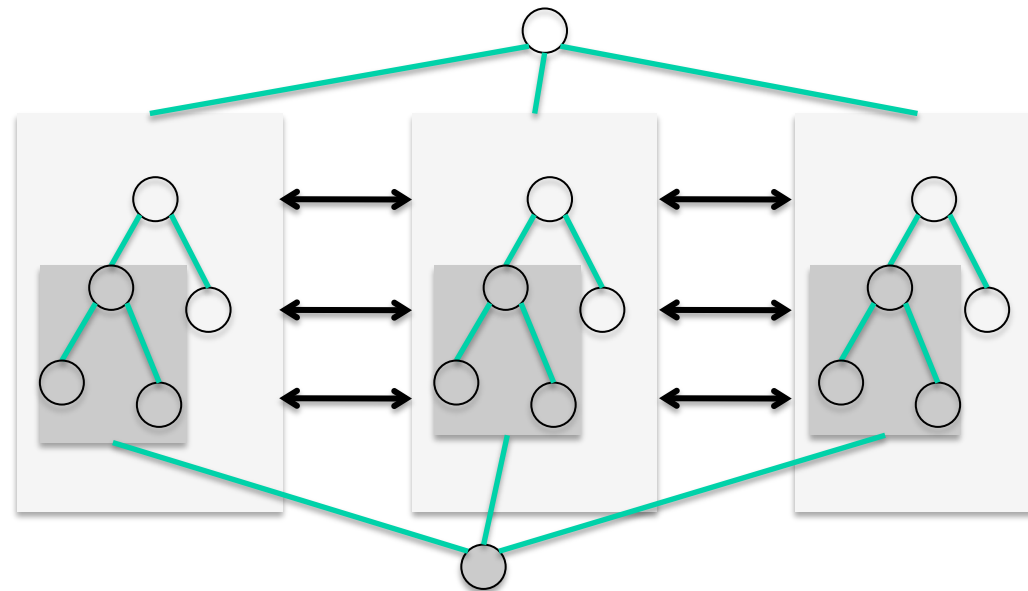
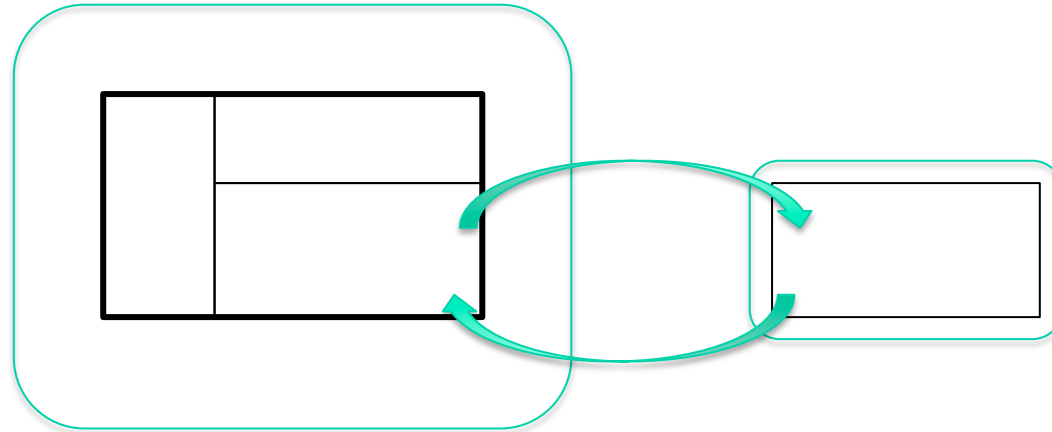
The bottom part of the screenshot shows the rendered widgets in the 'Rwt Emfacade View'. The 'Buttons' tab is selected, showing four widget types: Toggle Button, Push Button, Check Box (checked), and Radio Button. A large black play button is visible on the right side of the rendered area.

Property	Value
Arrow Style	NONE
Border Style	NONE
Button Style	CHECK
Enabled	true
Image	
Parent	
Selection	true
Size	
Style	33554464
Text	Check Box
Text Orientation...	LEFT_TO_RIGHT
Tool Tip Text	
Touch Enabled	false
Visible	true

# Rendering widgets across platforms

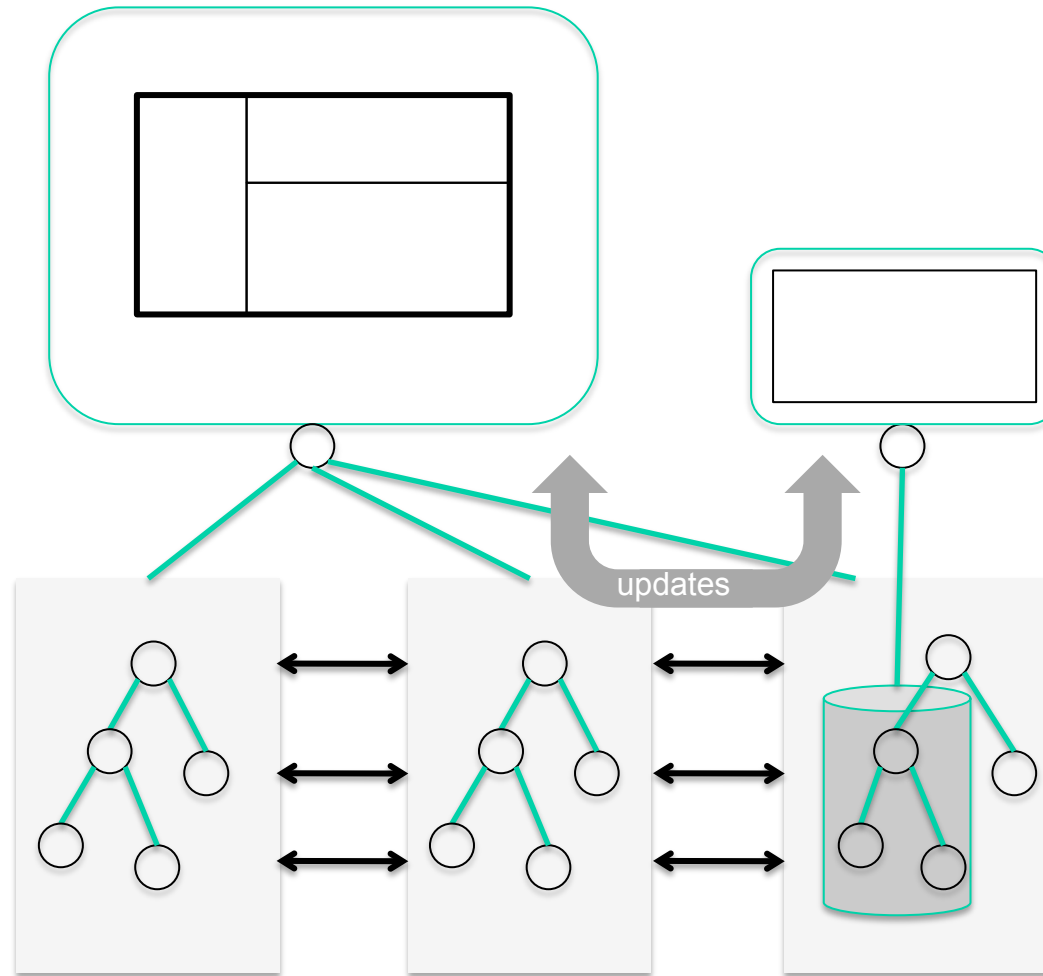
The screenshot displays an IDE interface with two views showing the rendering of widgets. The top view is a browser window at `http://localhost:8082/rwt/builderapp` with tabs for Text, Buttons, Lists, and Browser. The bottom view is the RWT Emfacade View for the application 'builderapp' on port 10010, with tabs for Text, Buttons, Lists, and Browser. Both views show a 'Toggle Button', a 'Push Button', a checked 'Check Box', and an unchecked 'Radio Button'. The IDE interface includes a toolbar with 'Problems', 'Javadoc', 'Declaration', 'Swf Emfacade View', 'Console', and 'Rwt Emfacade View'. Below the toolbar, the application name 'builderapp' and port '10010' are shown, along with 'Init RWT app' and 'Build with RWT' buttons.

# Moveable application

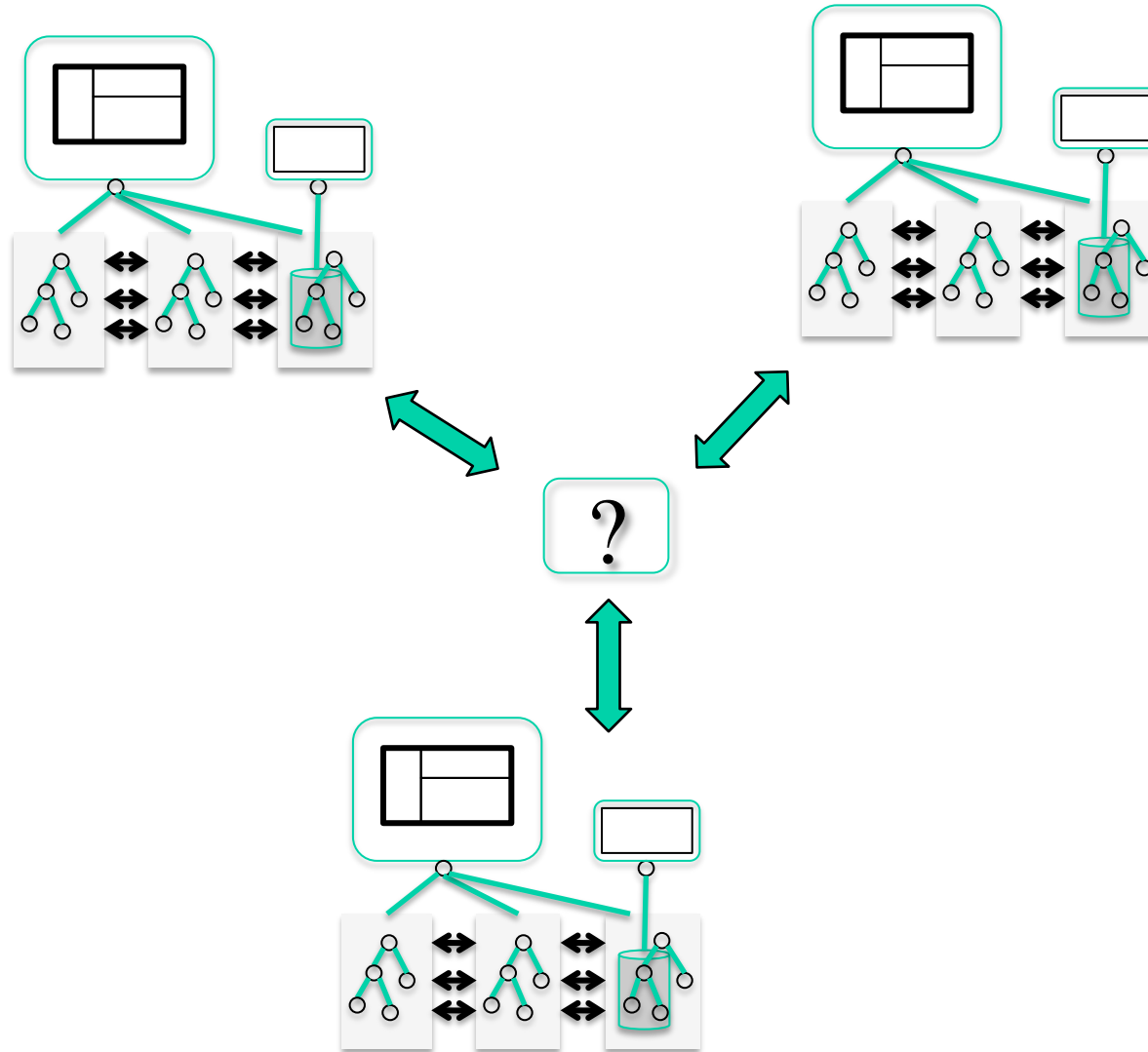




# Shareable application



# Distributed application



# ptui – ptolemy-based tool for UI development

- Diamodl
  - concepts are very close to Ptolemy's
  - interactors, computations and variables can all be modeled as actors
  - its weakness, the (lack of) semantics, is Ptolemy's strength
- Ptolemy can provide
  - a (set of possible) semantics
  - a solid runtime platform
- Ptolemy
  - describes the behavior of a cyber-physical system, but
  - has poor support for modeling user interaction
- Diamodl can provide
  - an approach to integrating UI elements
  - runtime support for rendering widgets locally or in a browser