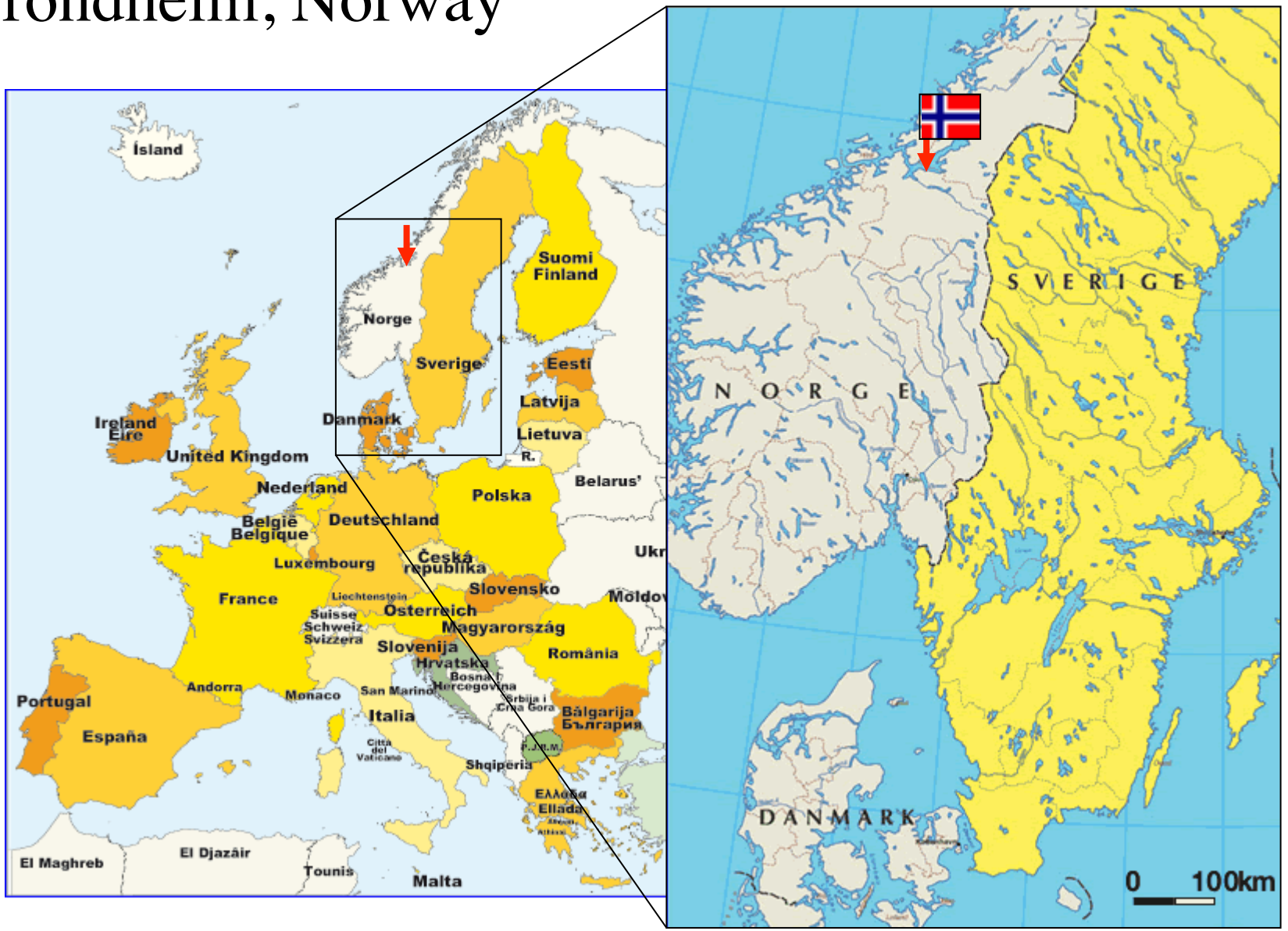
NTNU

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

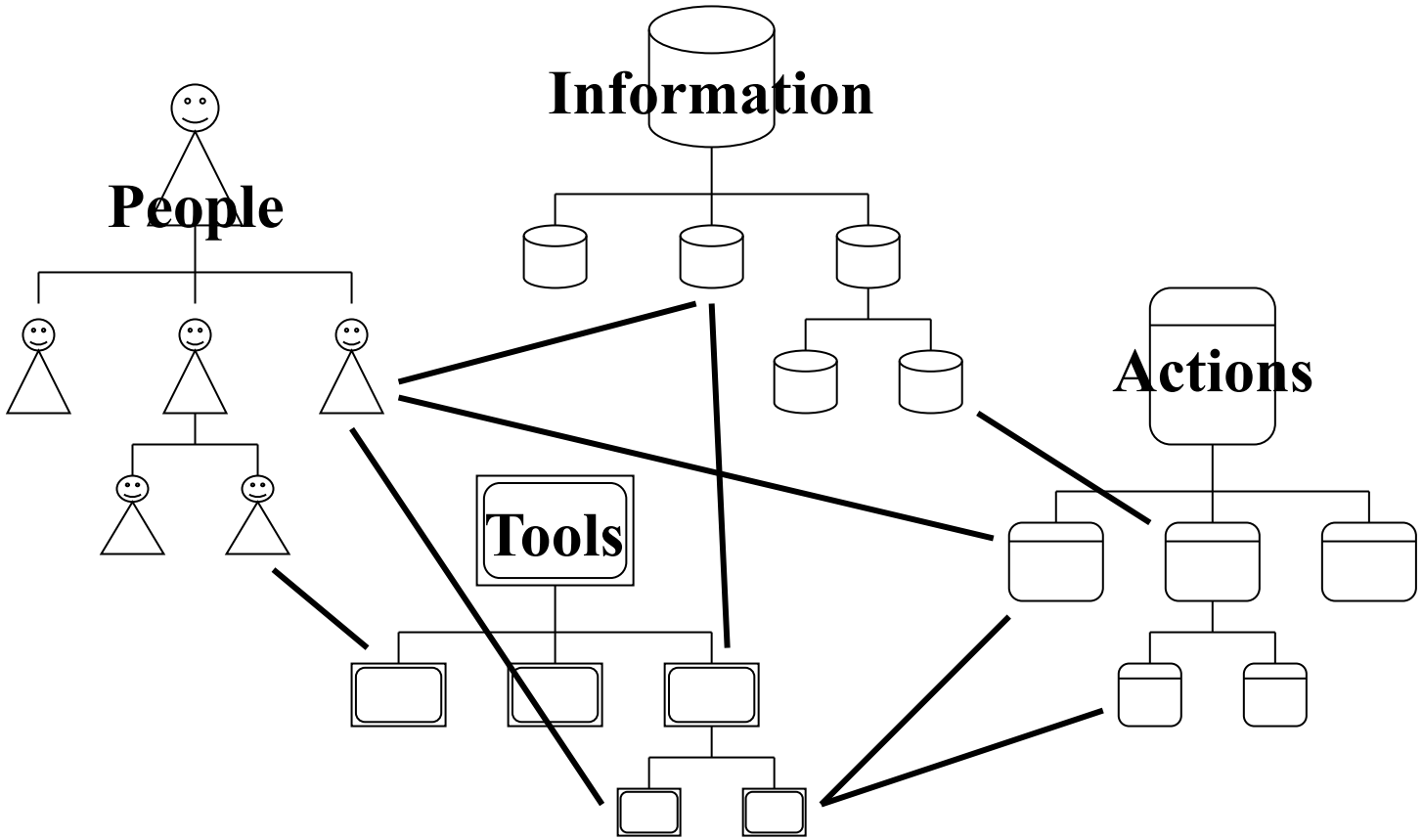


NTNU

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

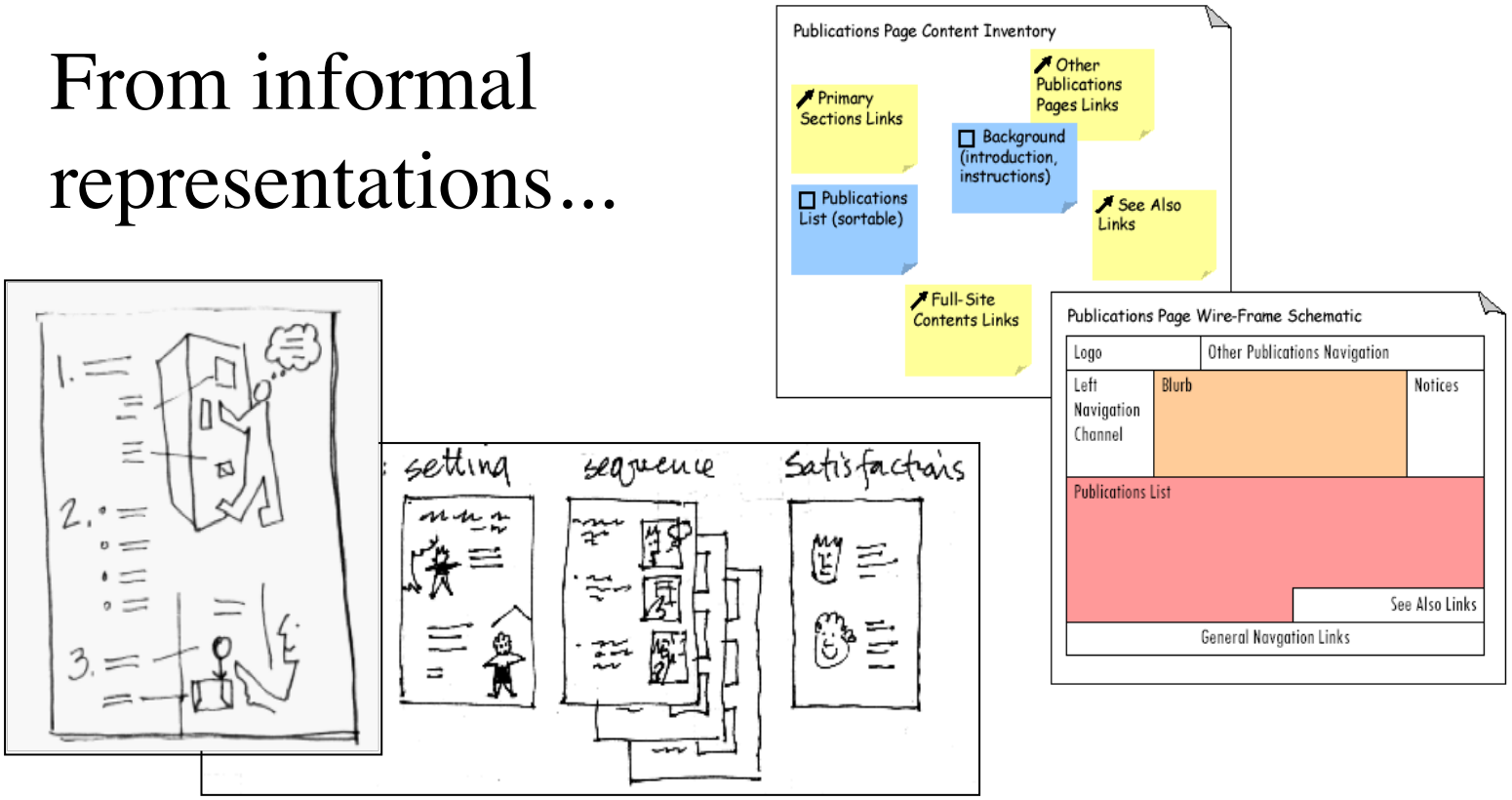


Roles representations play

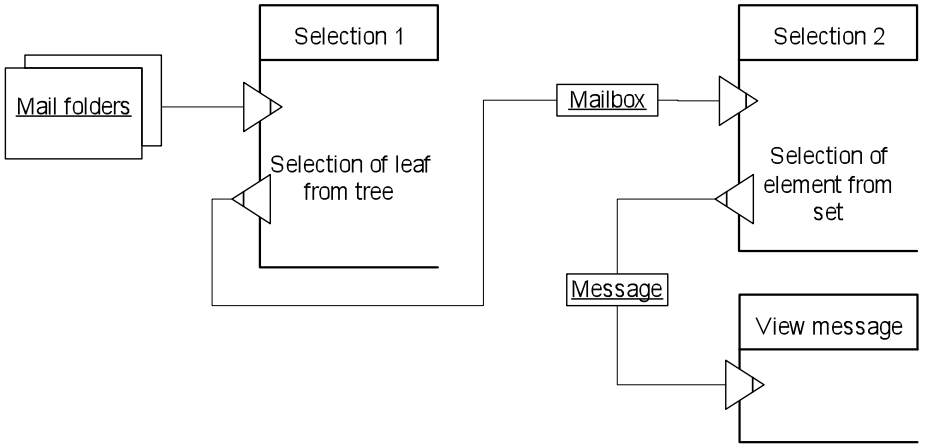
- Semantic
  - accurately and completely capture knowledge
- Communicative
  - support communication among designer and end-users
- Constructive
  - stimulate, guide and constrain further design
- Analytic
  - support interpretation and evaluation

- Engineers and designers focus on different roles

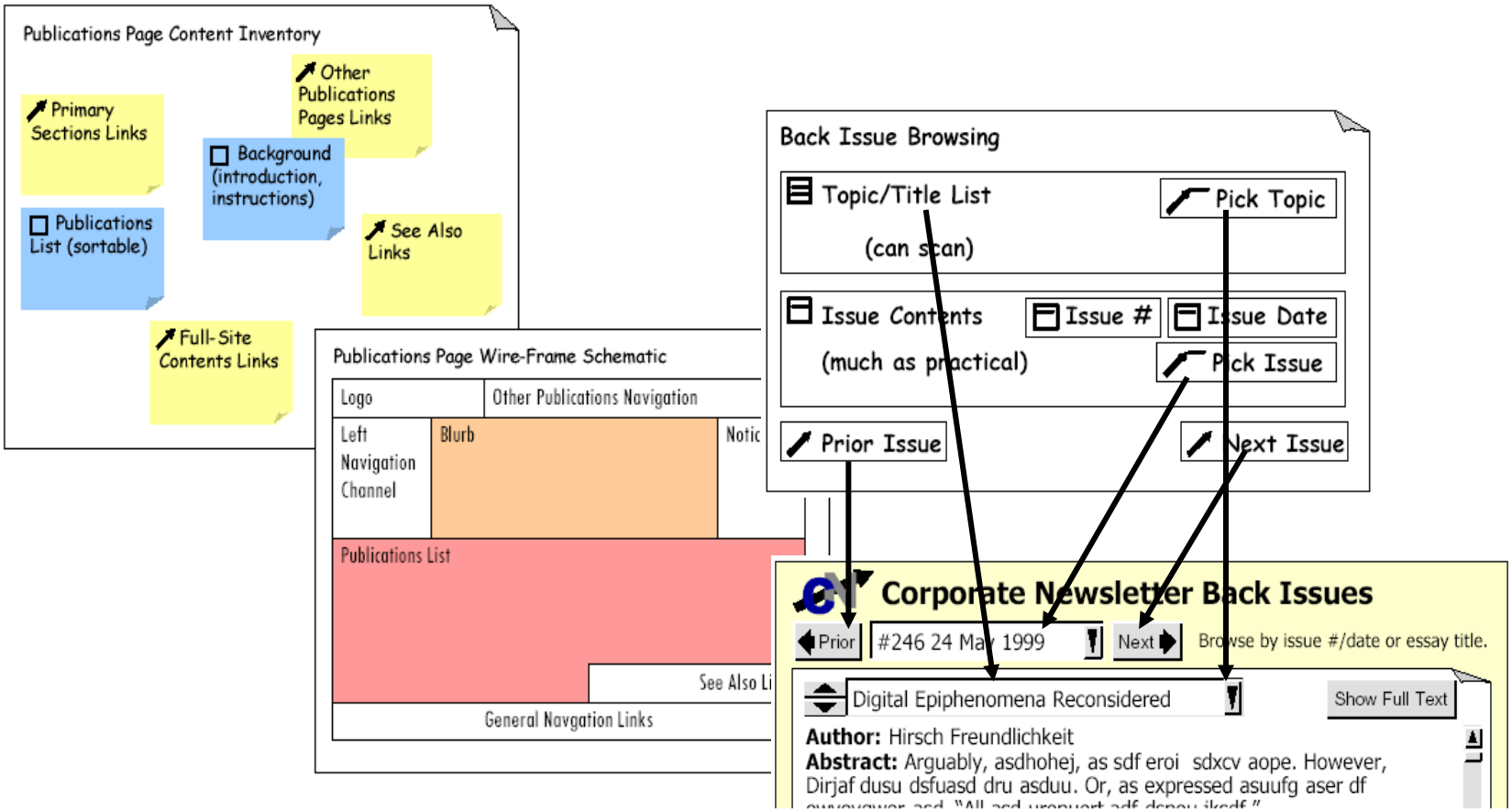
From informal representations...



... to models



Canonical Abstract Prototypes [Constantine] – semi-formal sketching



What aspects of a UI do we want to capture?

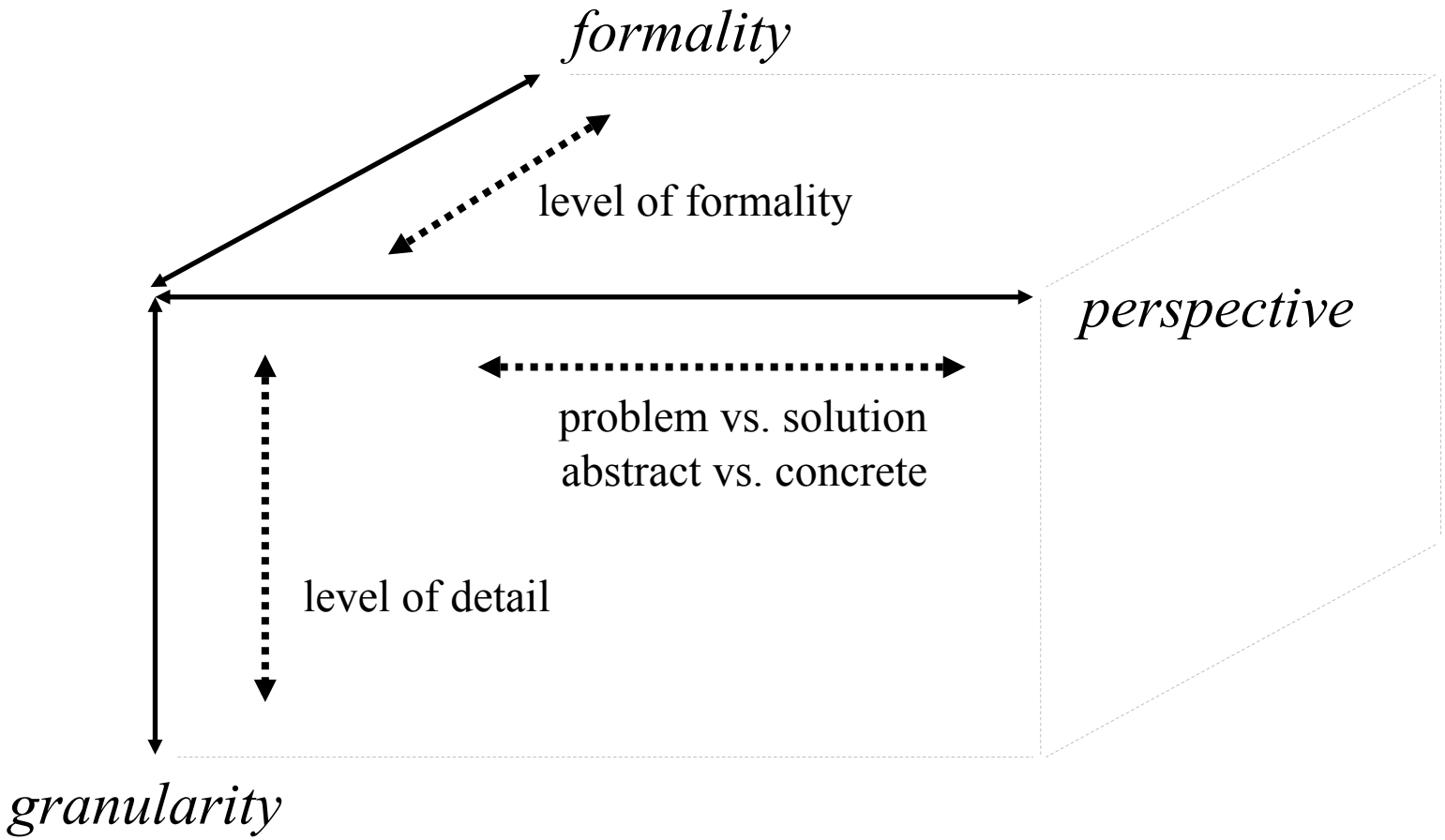
- Structure
  - hierarchical structure of interaction elements
- Information
  - what information is accessible in which parts of the UI
  - what is the relationsship between information in various parts of the UI
- Behavior
  - when are the various interaction elements active
  - how are changes in 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

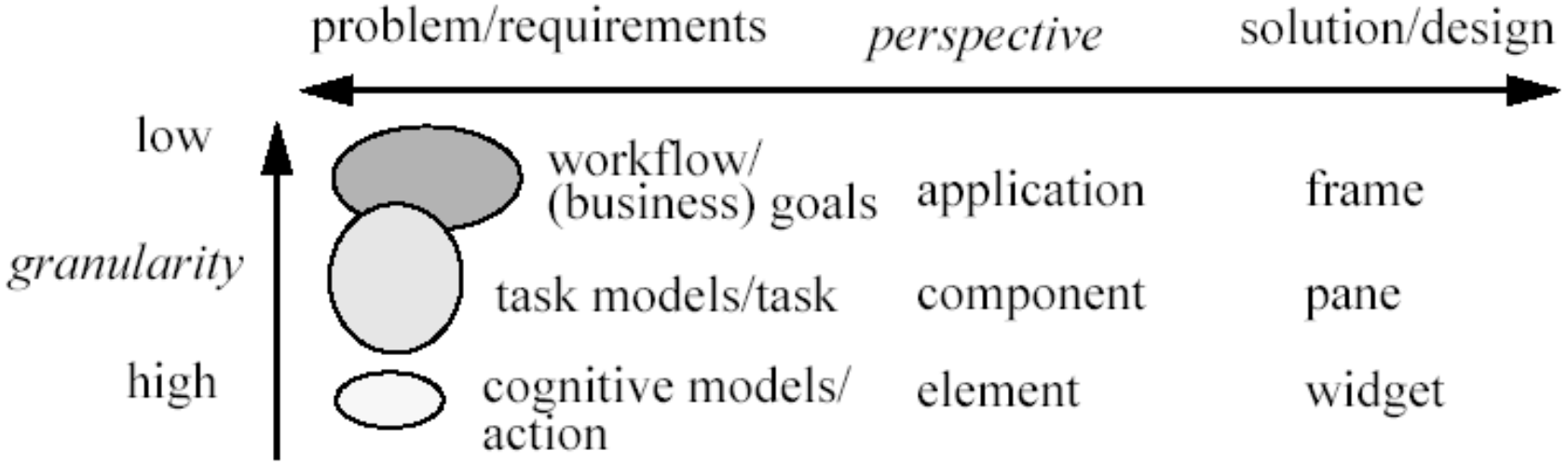
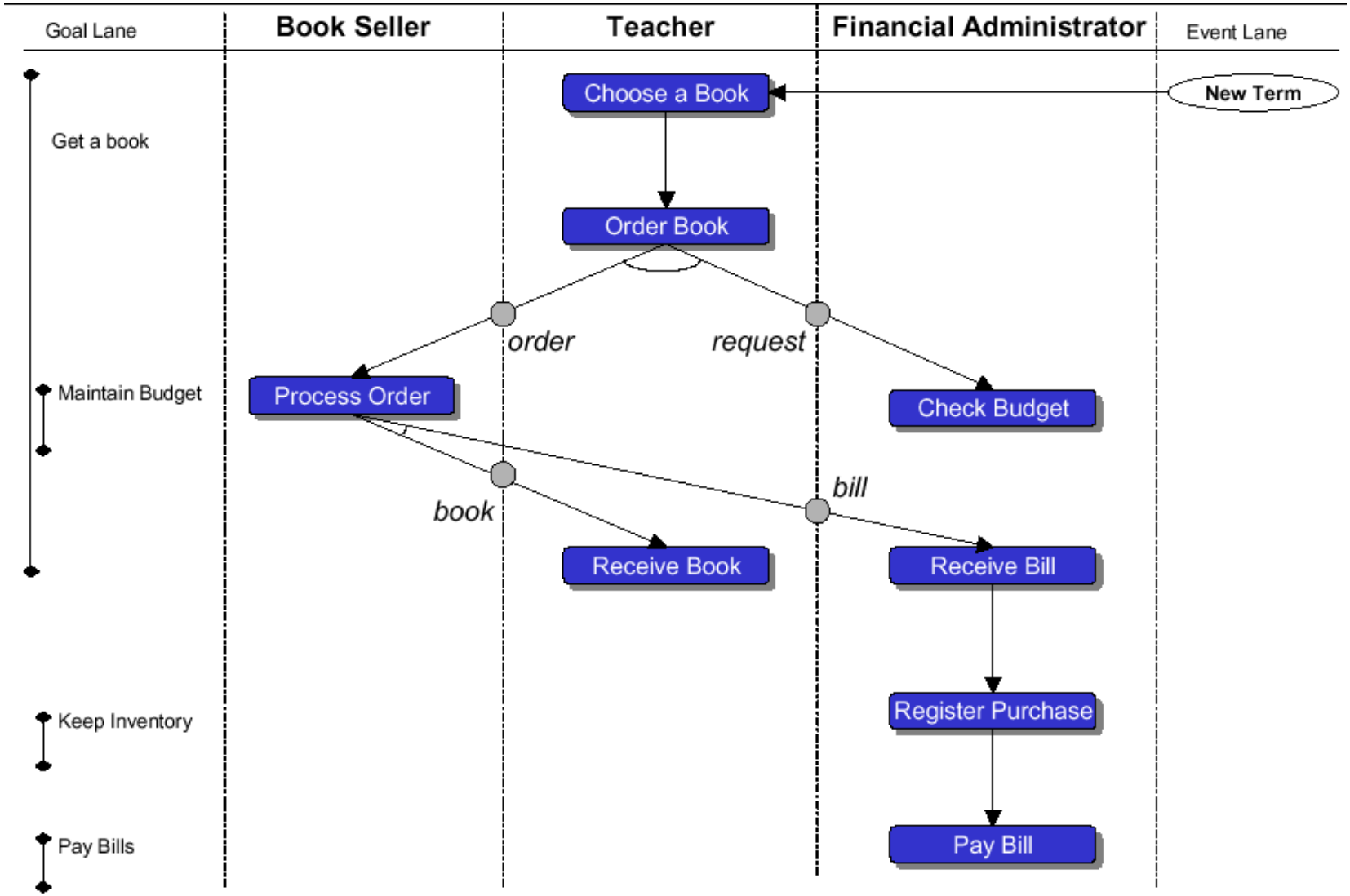
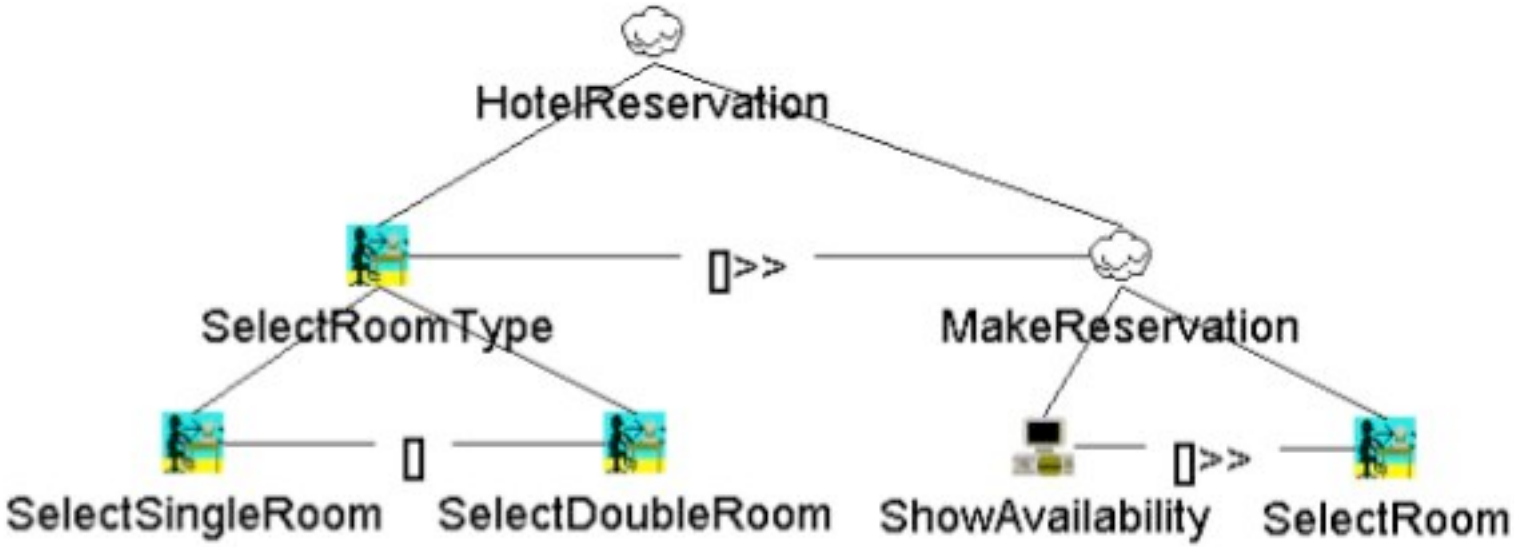


Figure 12. The level/granularity dimension interpreted across perspectives

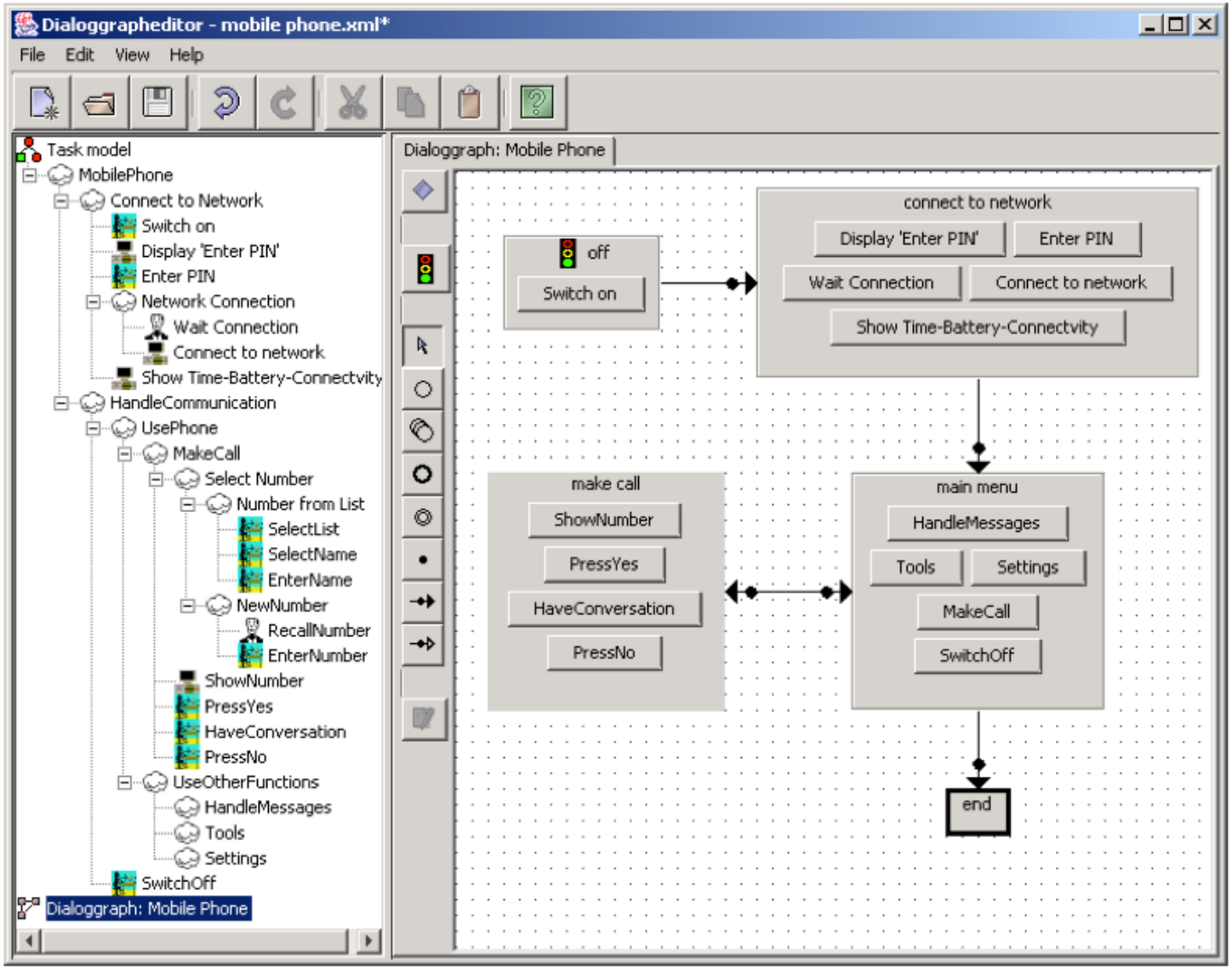
Dutch [van der Veer] - task models as activity charts



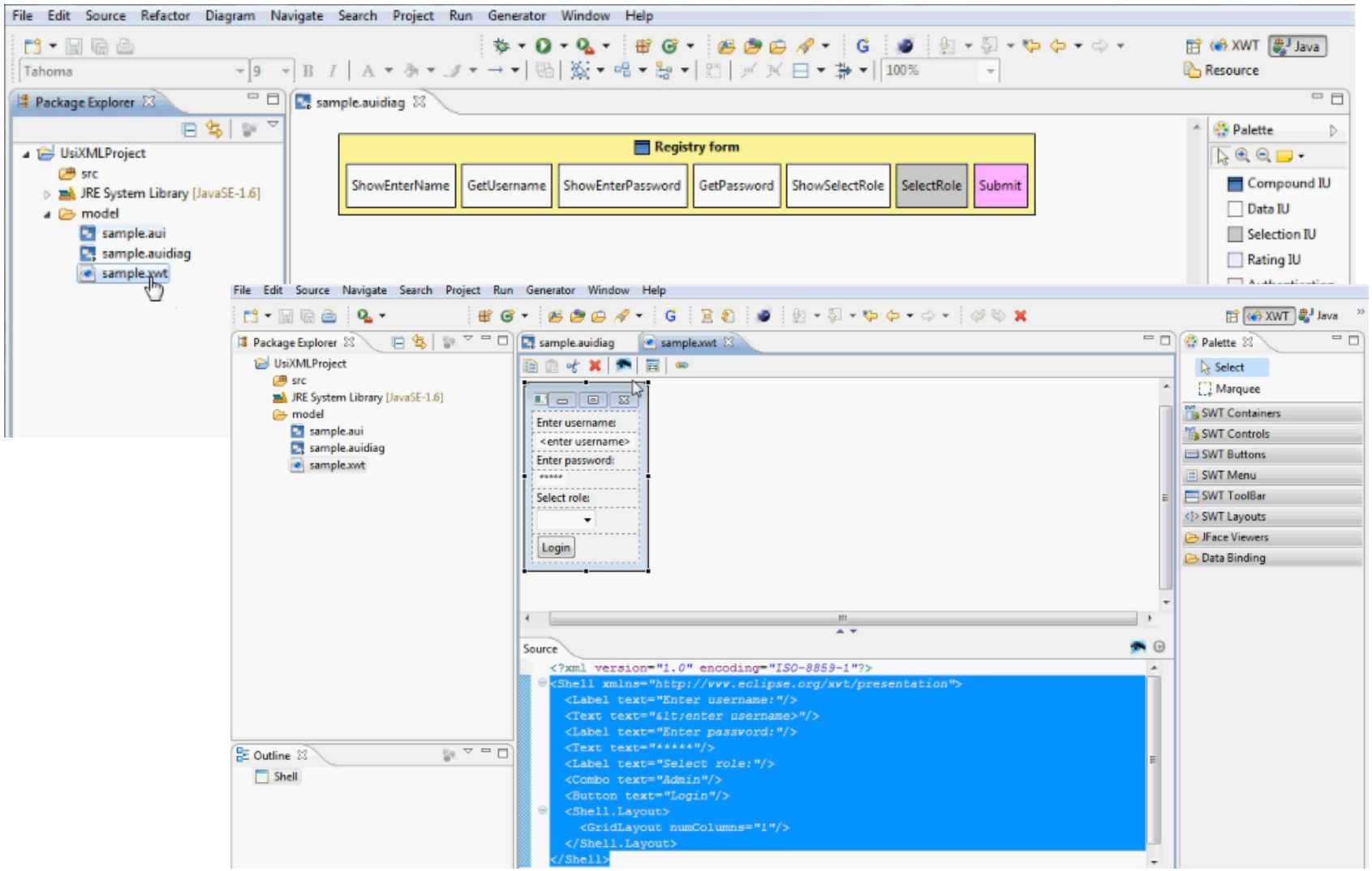
ConcurTaskTrees [Paternò] – task hierarchies with temporal operators



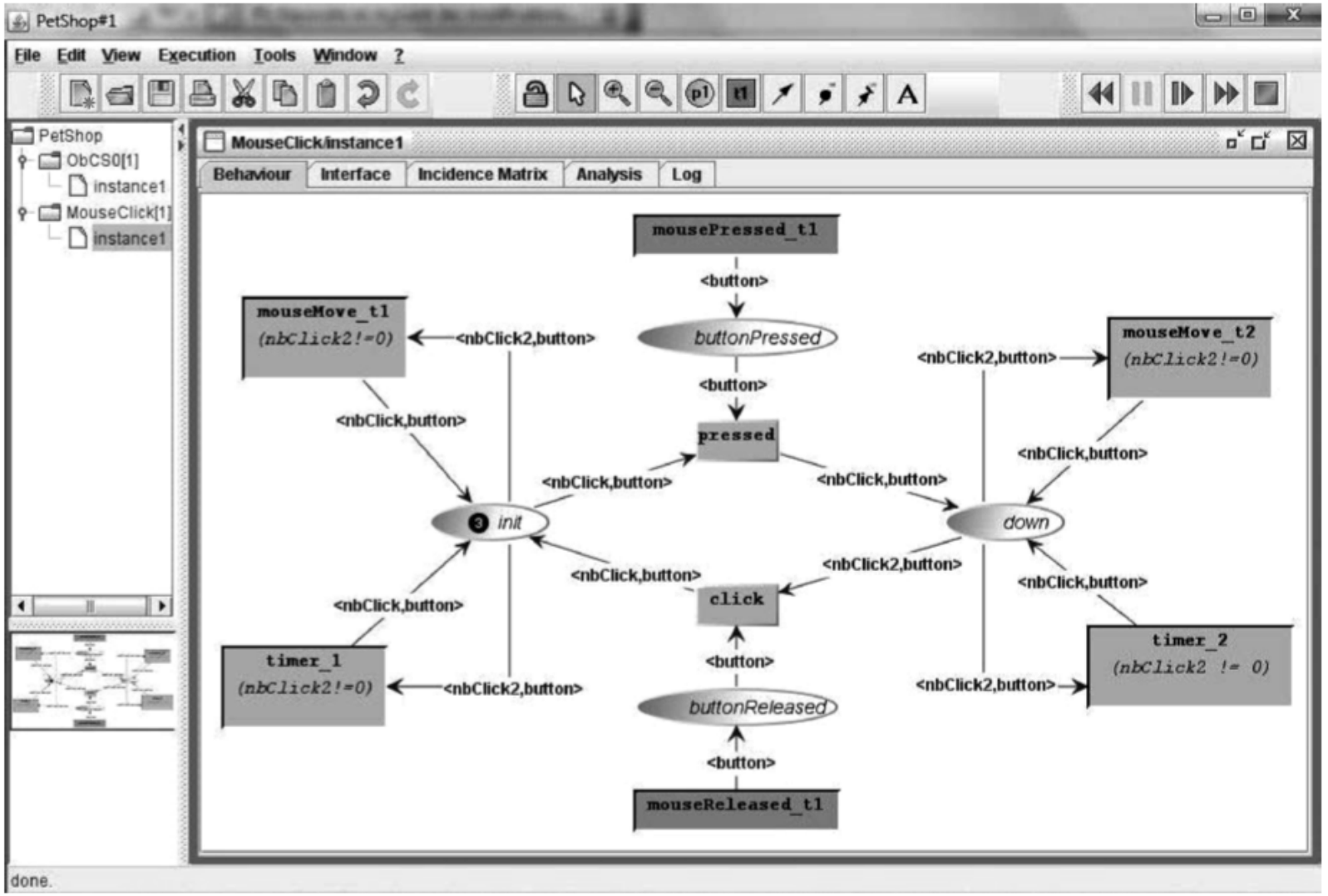
Dialog graphs [Forbrig] – Relating tasks to dialog



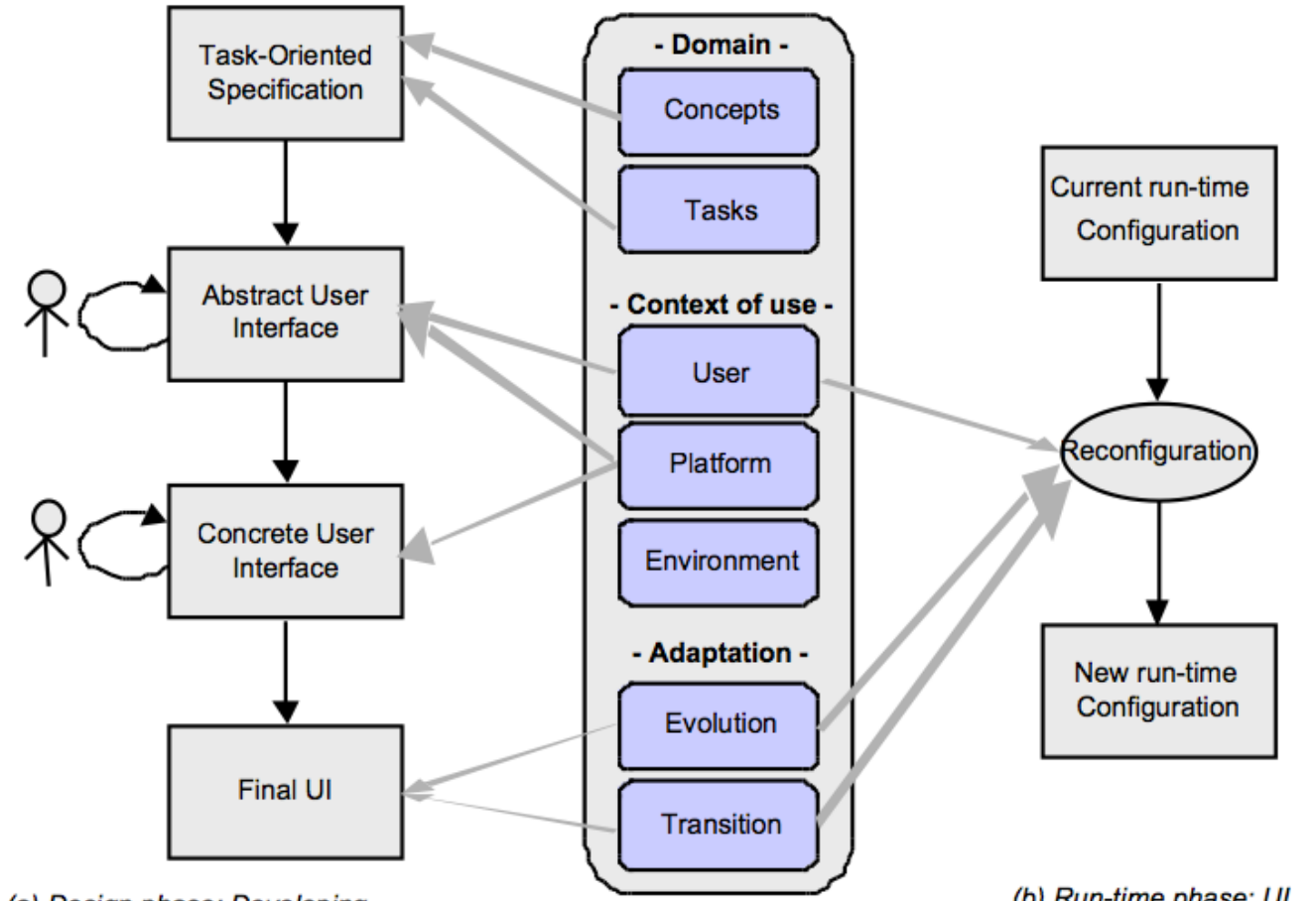
UsiXML [Vanderdonckt] – A family of XML-based notations for UI elements



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



Cameleon framework – targeting multiple devices



(a) Design phase: Developing UI versions for different targets

Models used for both the Design and Run-time phases

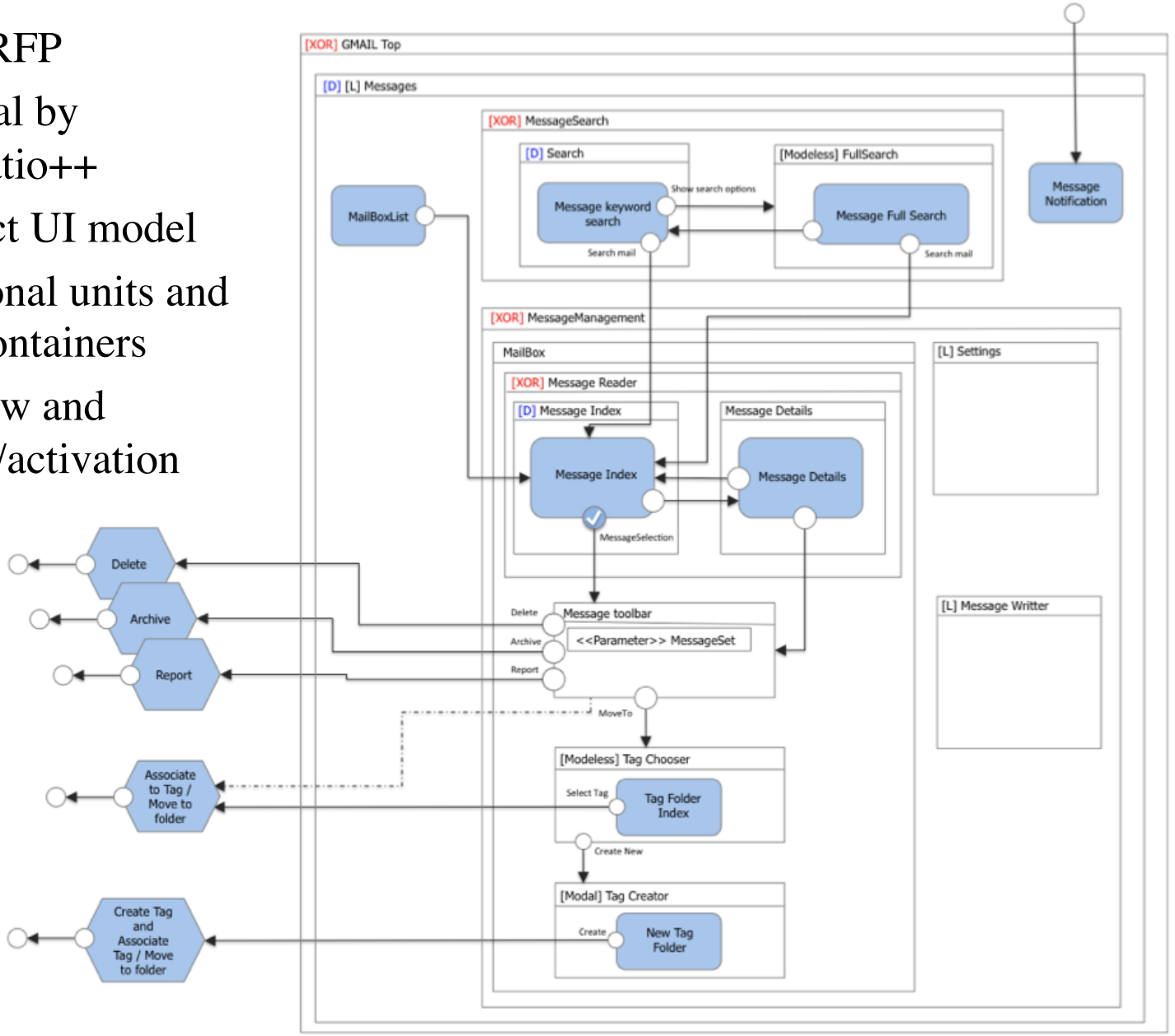
(b) Run-time phase: UI adapts to changes of the Context of use

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

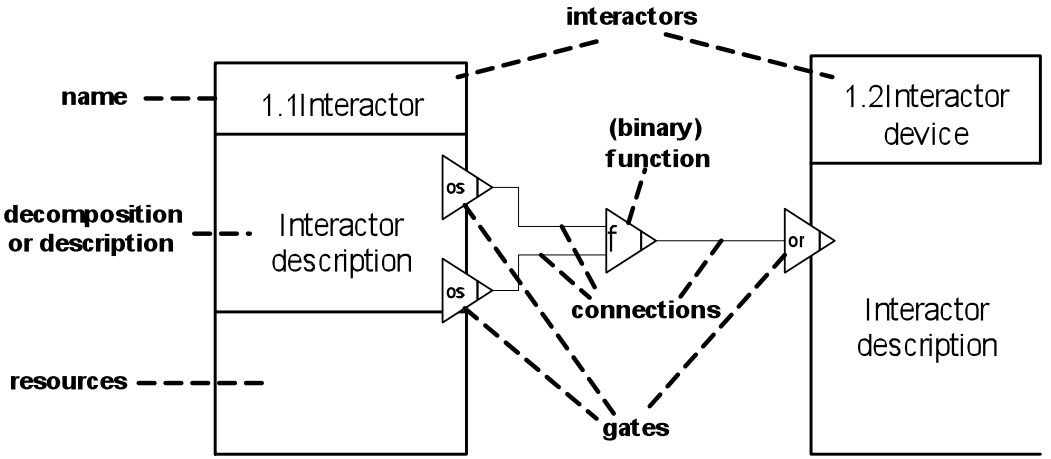
IFML – Interaction Flow Modeling Language

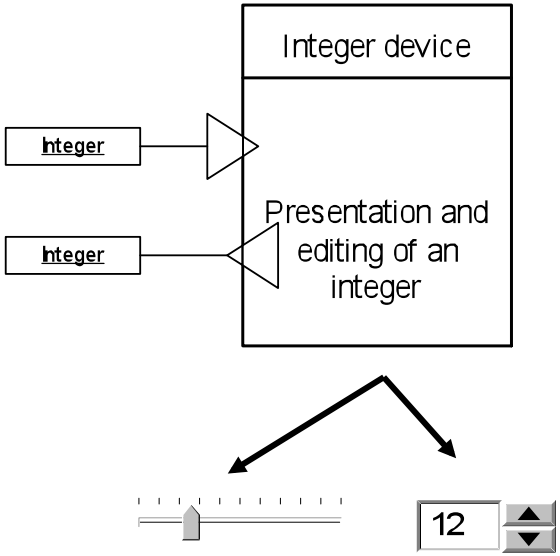
- OMG RFP
- Proposal by WebRatio++
- Abstract UI model
- Functional units and view containers
- Dataflow and control/activation signals



Dialog modelling with DiaMODL

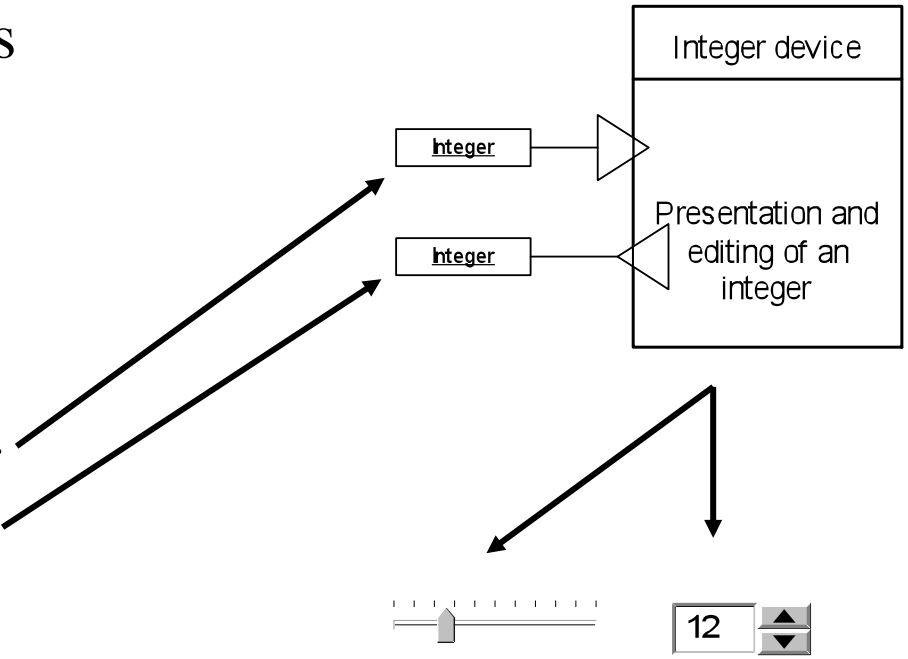
- Based on Pisa interactors and Harel's Statecharts
  - interactors, gates and connections
  - hierarchical states
  - transitions, events/actions, conditions
- Abstraction of IO function
- Composition in terms of
  - interactor structure
  - state hierarchy (and, or)





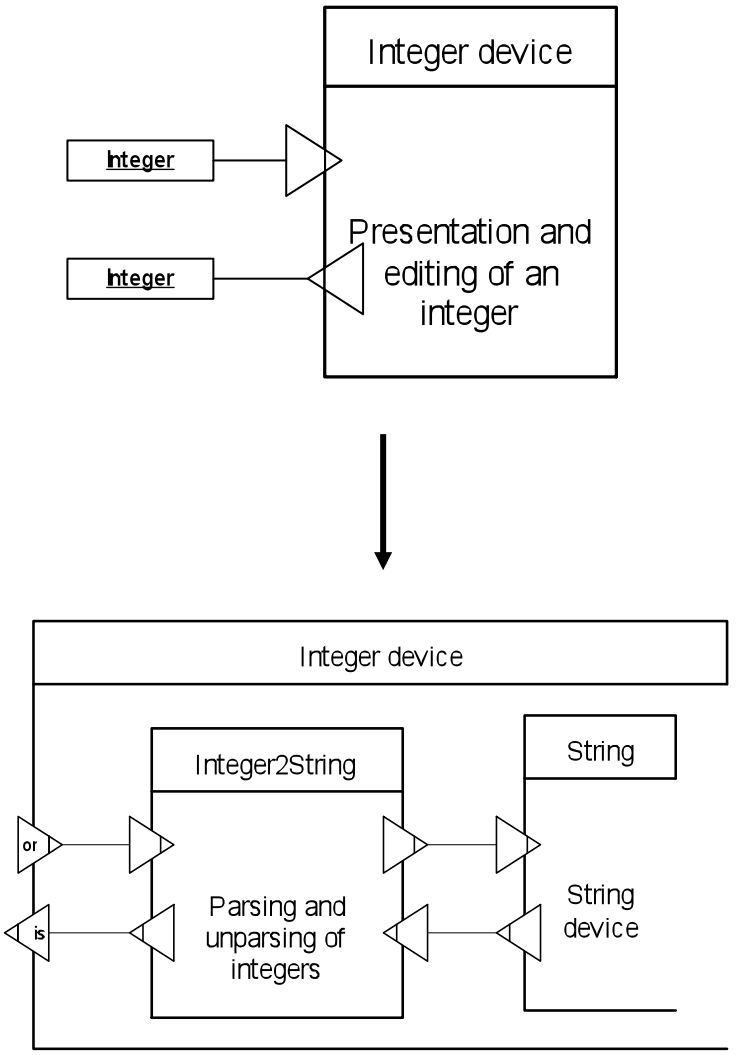
Generic interactor abstraction

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



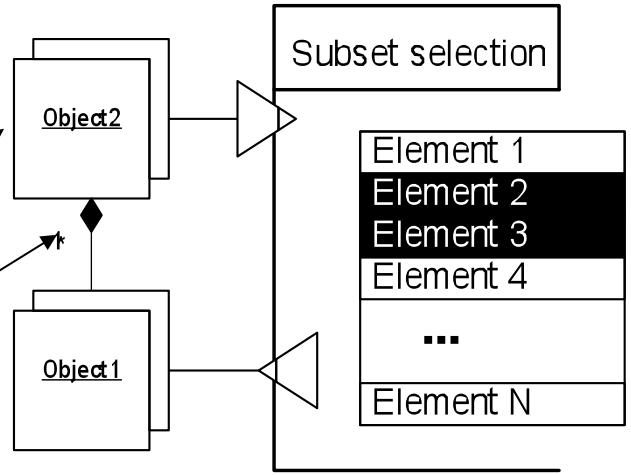
Scalable notation

- 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

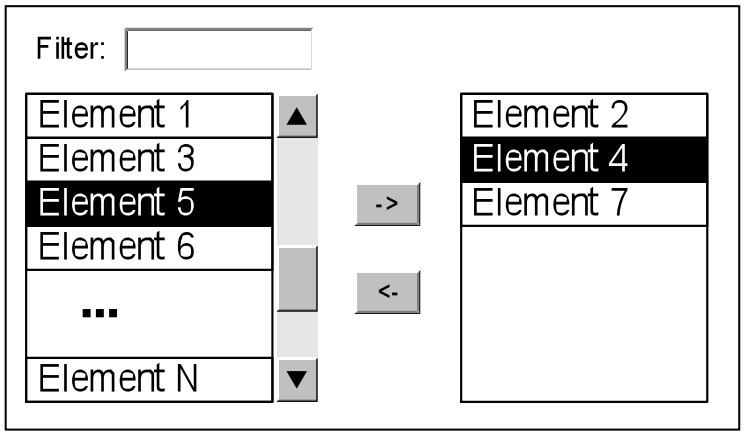


More complex interaction objects

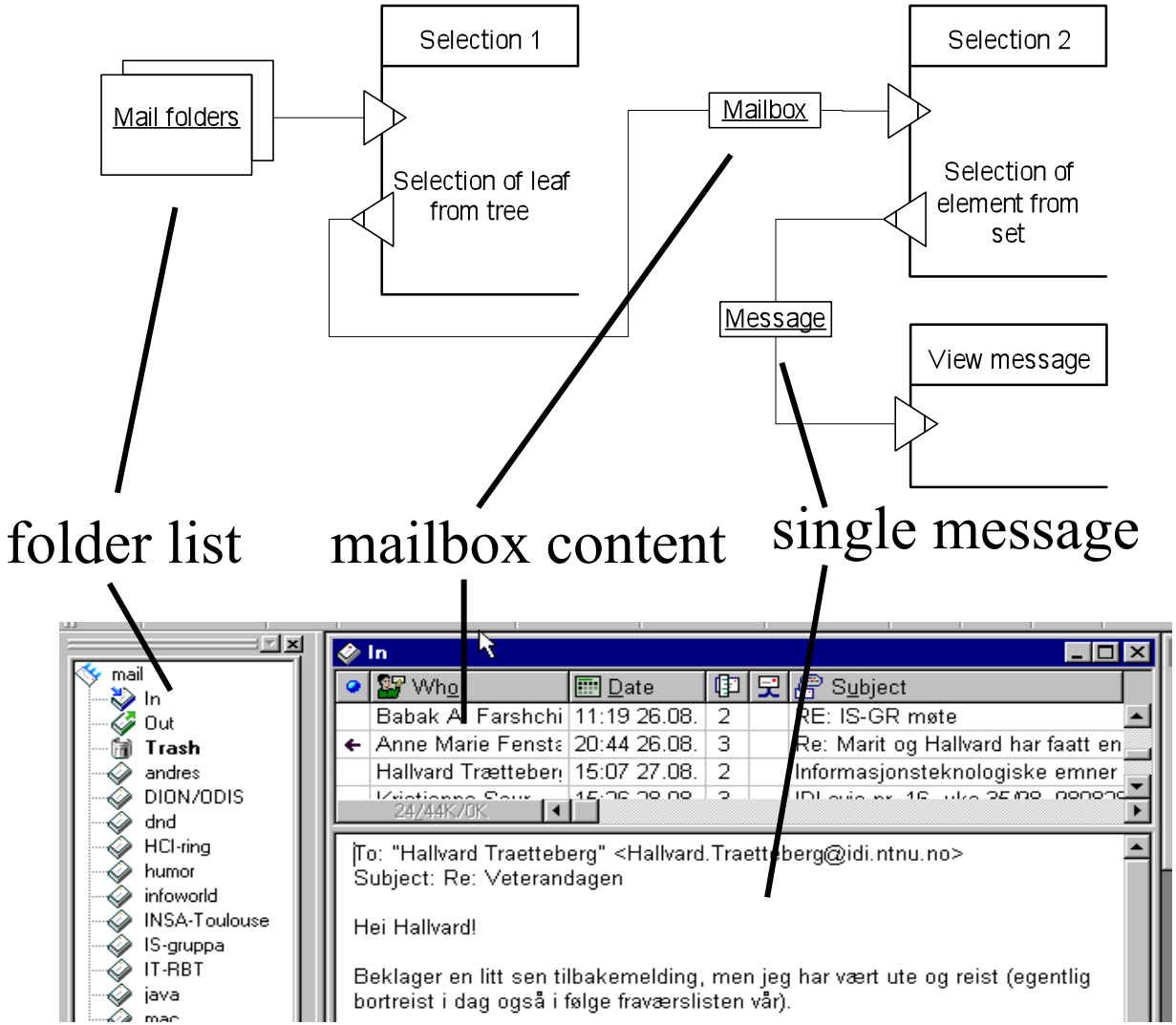
- Functionality defined in terms of configuration of domain objects
- Utilise power of domain modelling language
  - Output: set
  - Input: subset



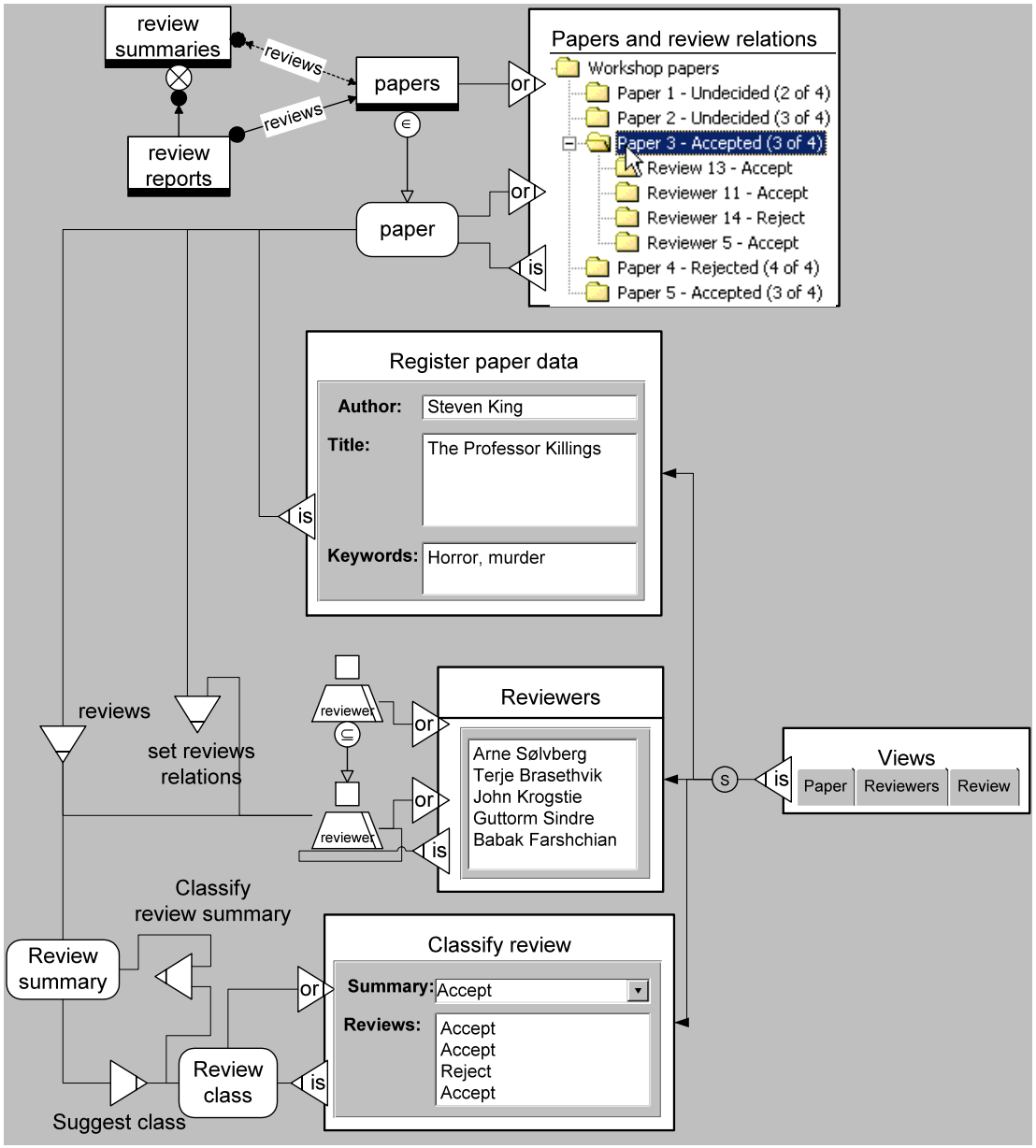
Alternative implementation



Configuration of larger elements

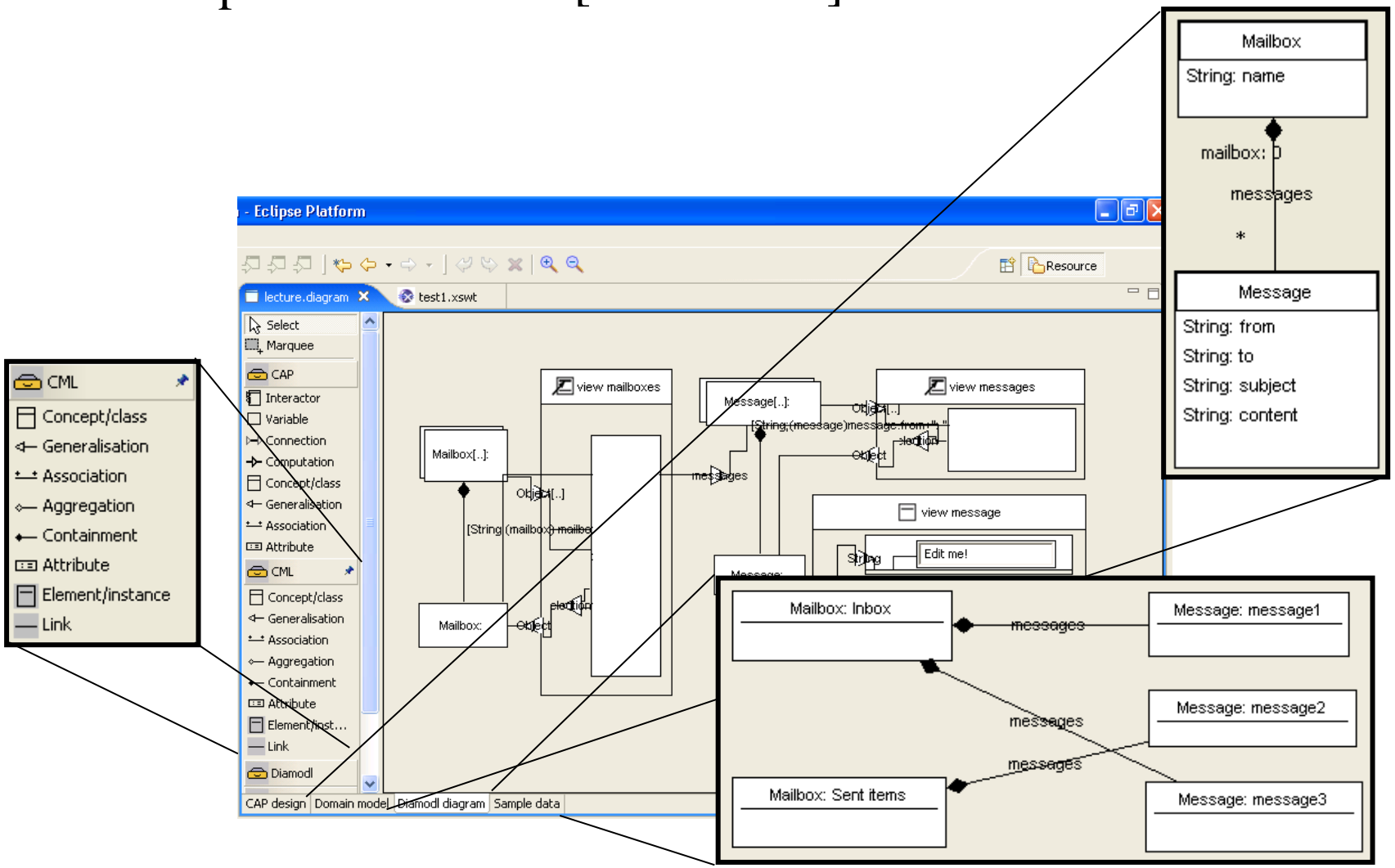


Interactor-based GUI-builder

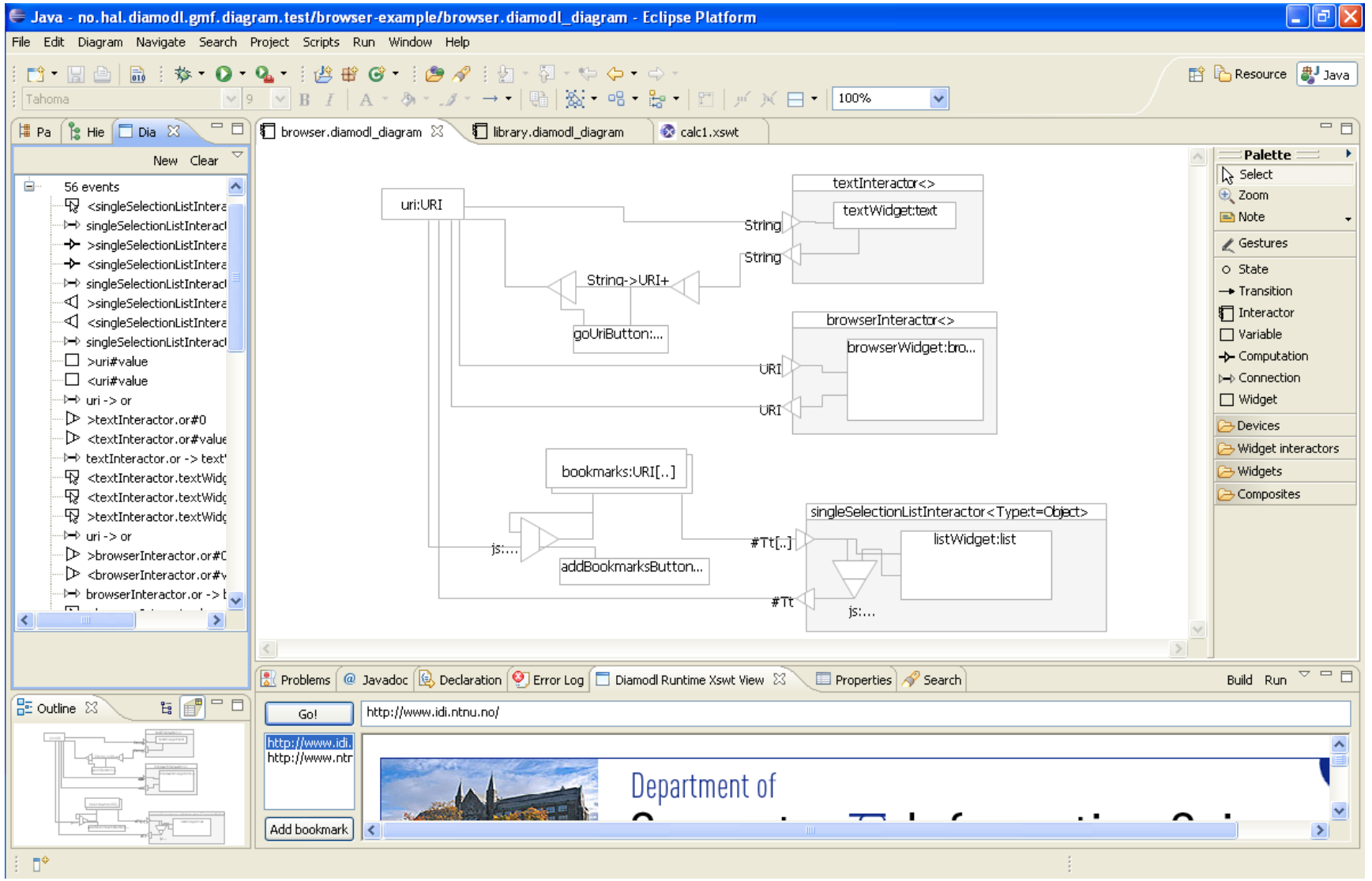


Integrating domain and dialog modeling

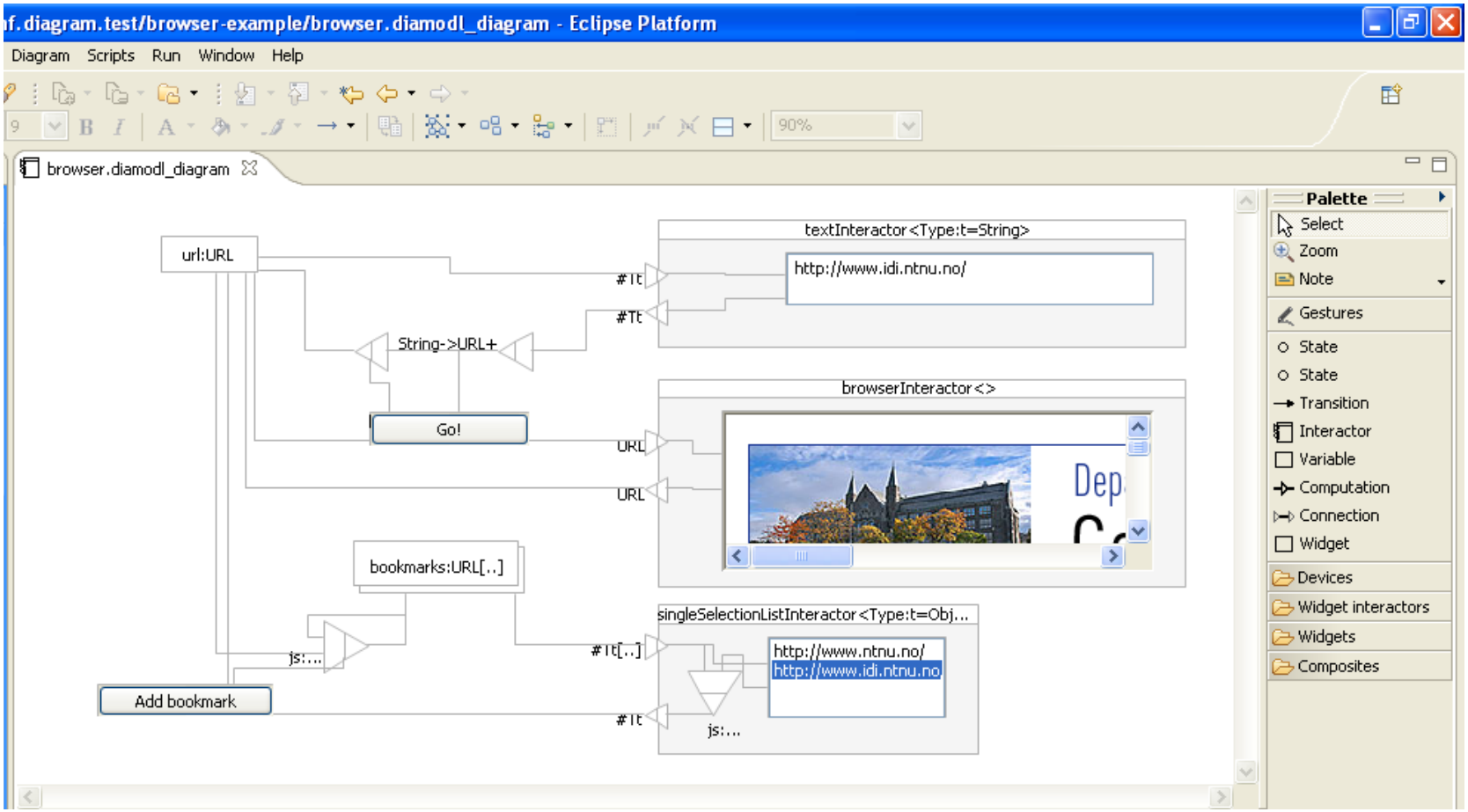
- Eclipse-based editor [CADUI'06]



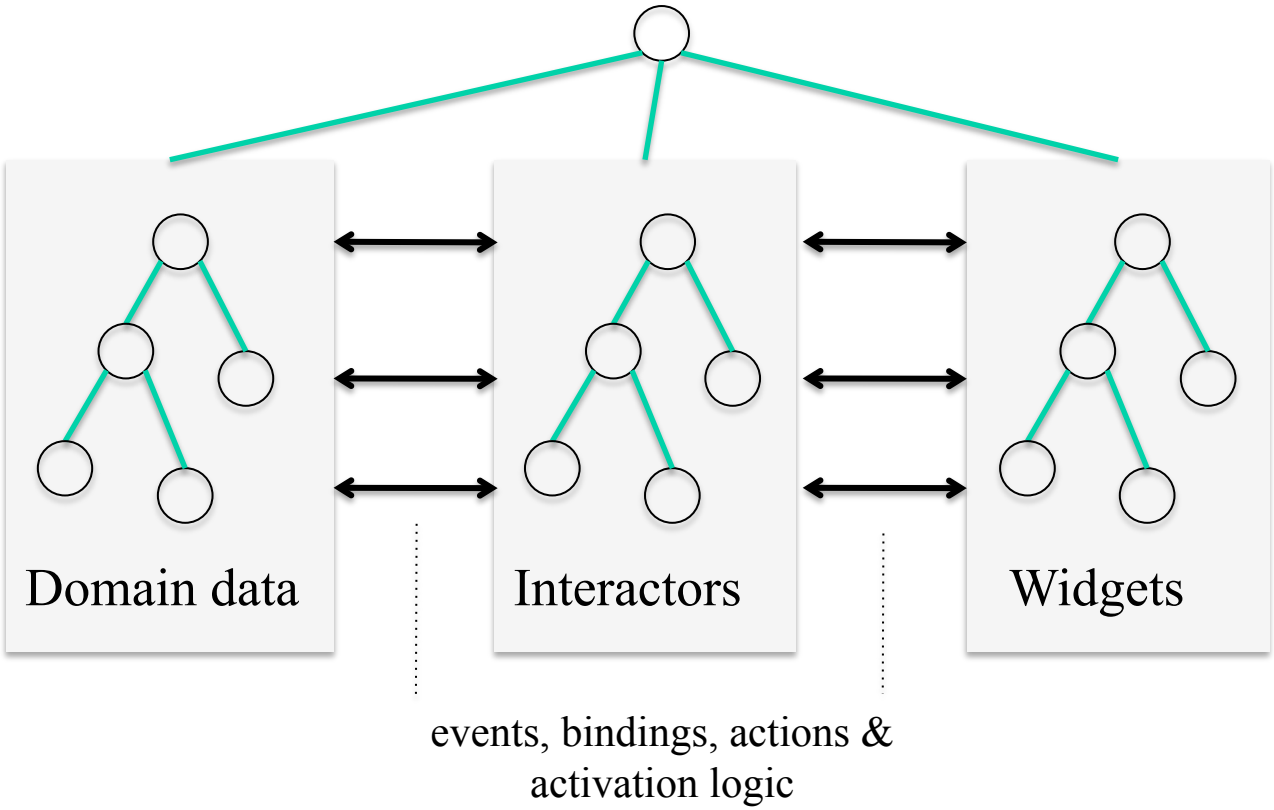
Prototyping with Diamodl



Prototyping with Diamodl



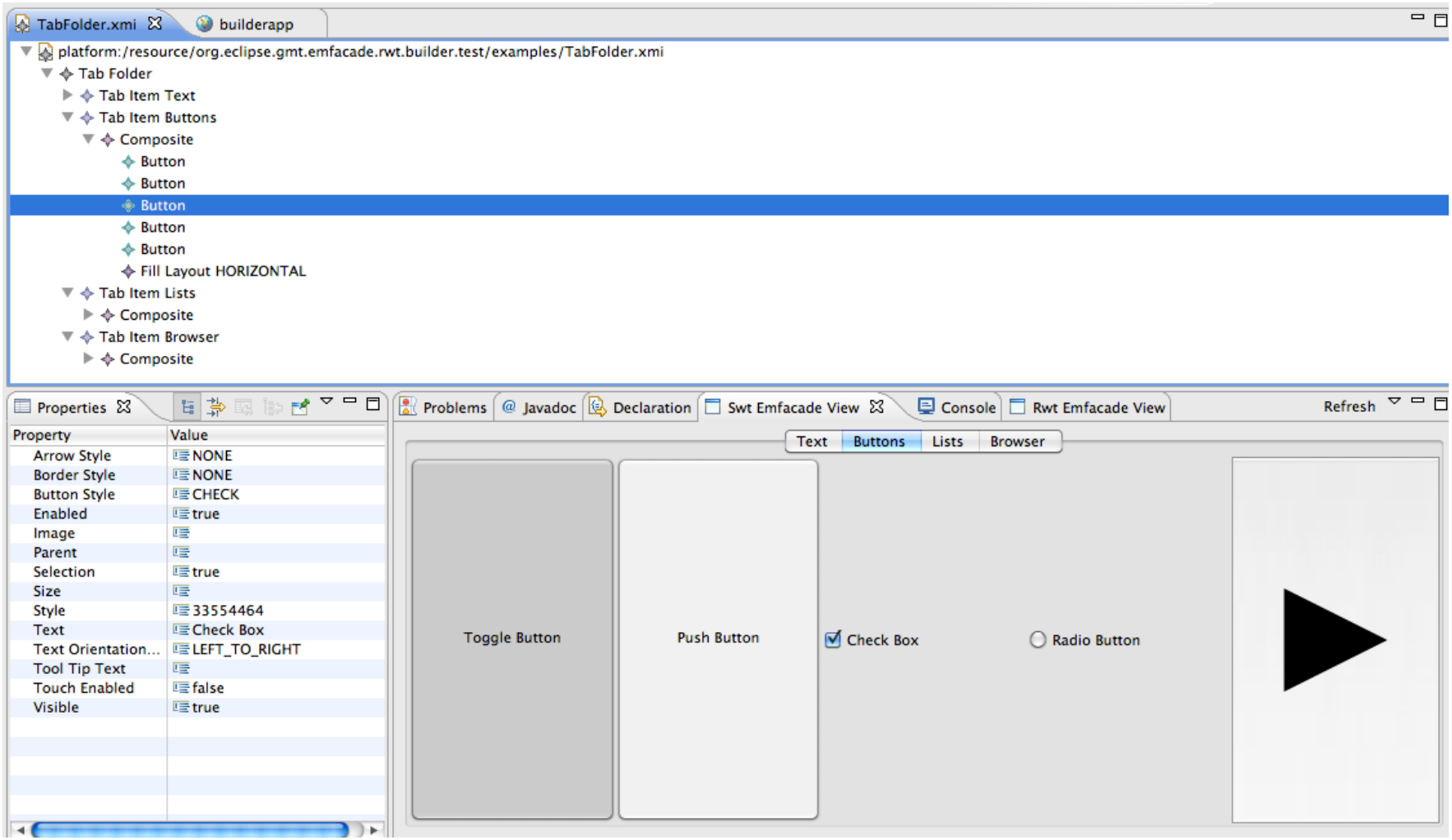
Application architecture



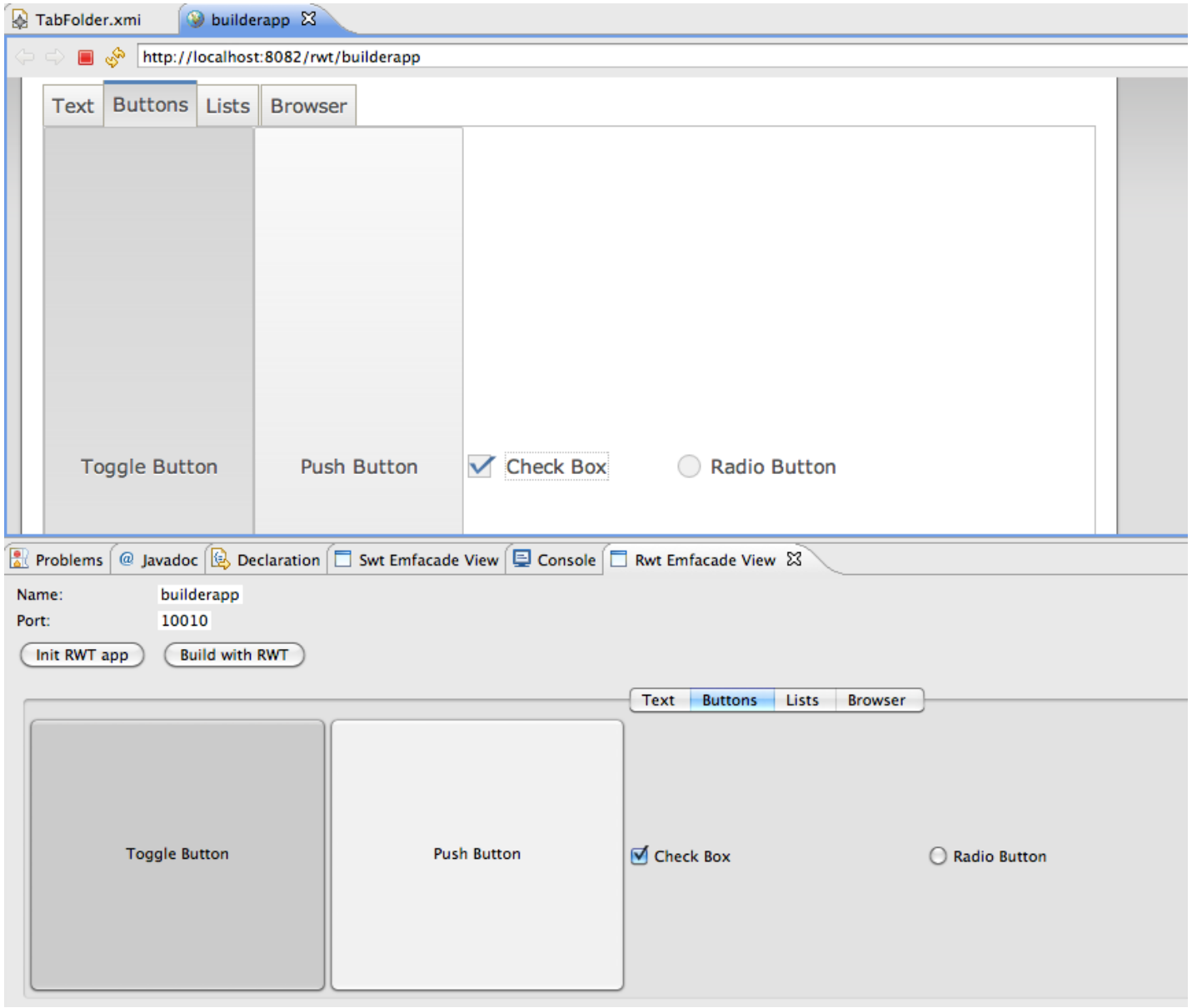
- The whole runtime state is captured as coordinated graphs of data
- The widget hierarchy is continuously rendered on a device

Rendering widgets

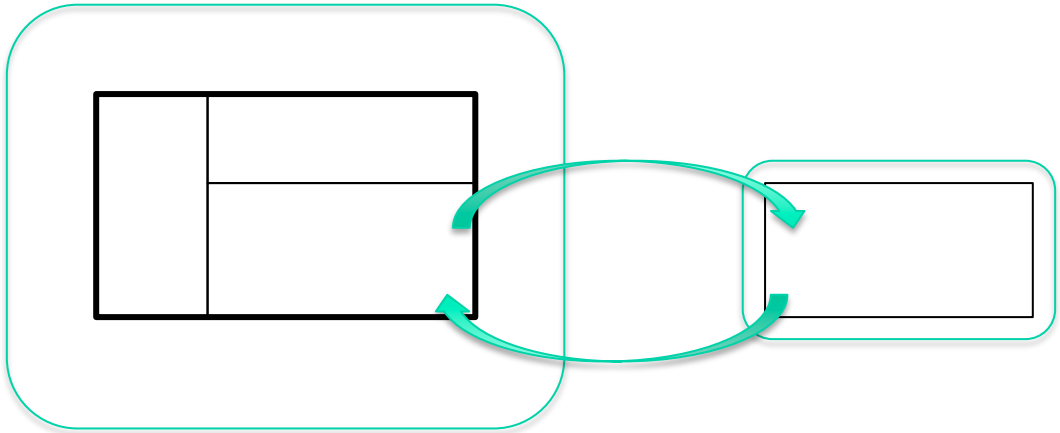
- Ecore model of toolkit, with instances rendered in Eclipse view

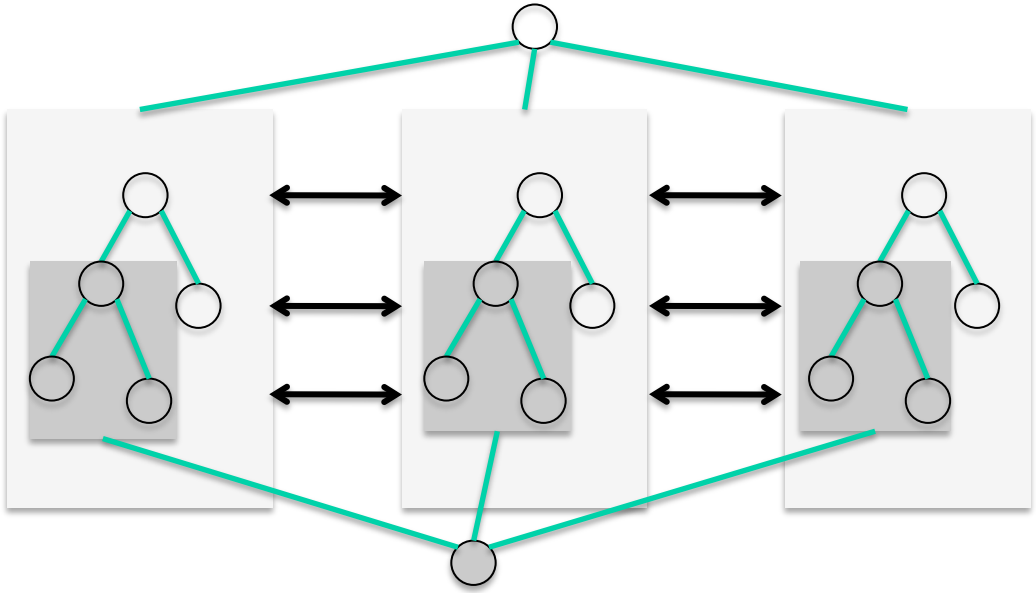


Rendering widgets across platforms

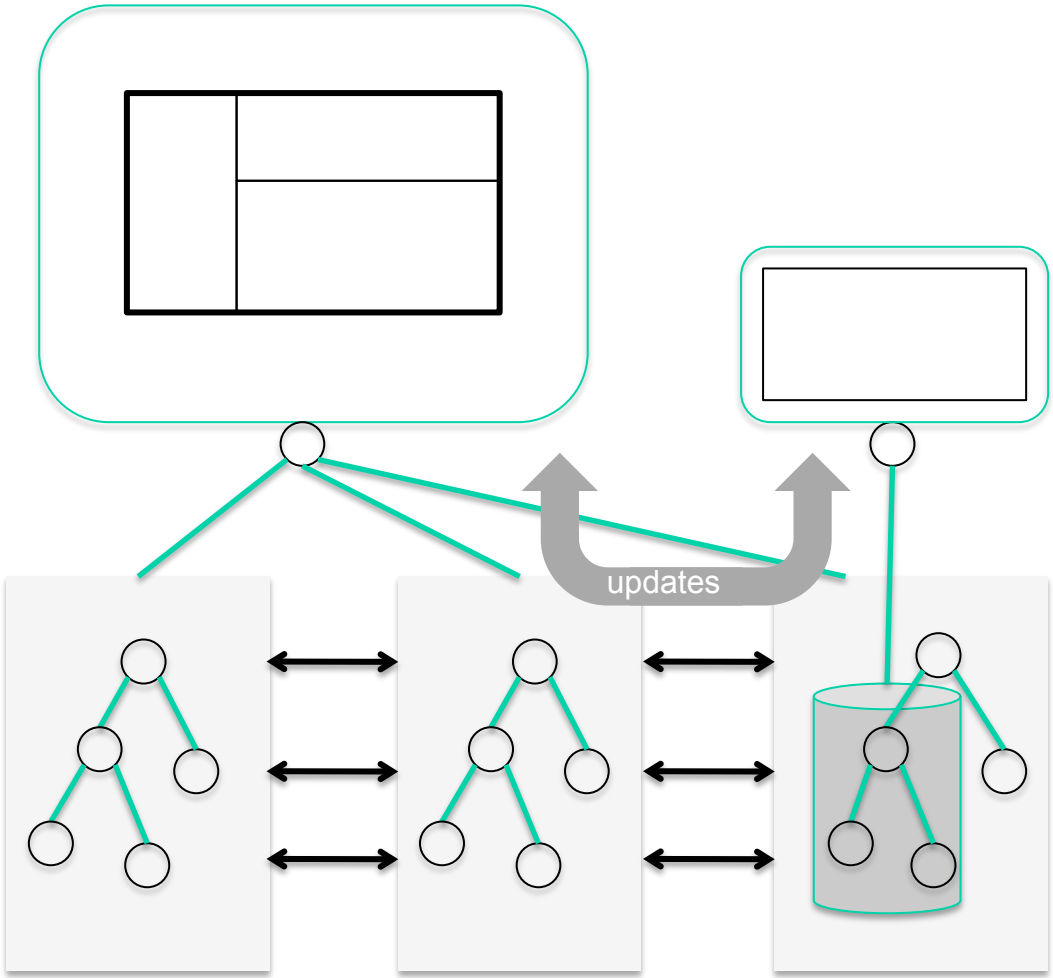


Moveable application

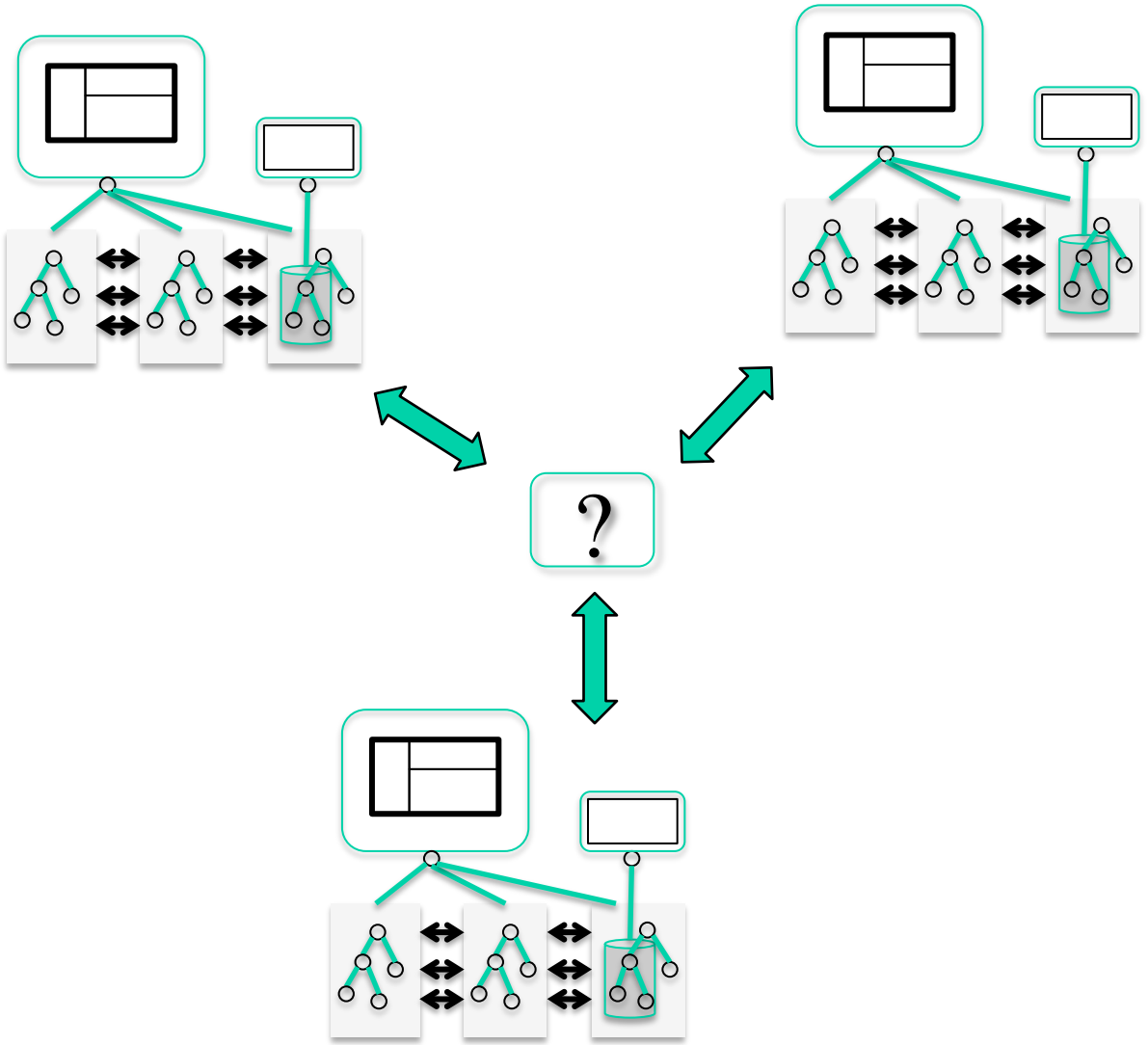




Shareable application



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