UsiXML, a User Interface Model and Language Engineering approach

*Jean Vanderdonckt, **Juan Manuel Gonzalez Calleros***

Université catholique de Louvain (UCL)
Louvain School of Management (LSM)
Information Systems Unit (ISYS)
Belgian Laboratory of Computer-Human Interaction (BCHI)
http://www.isys.ucl.ac.be/bchi

The problem

- Complexity and the diversity of existing development environments.
- These difficulties are exacerbated when the same UI should be developed for multiple contexts of use.
- Although designers and programmers are involved in these types of project, the available tools are mainly target at the developer.

What is UsiXML

- UsiXML (which stands for USer Interface eXtensible Markup Language) is a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces, and Multimodal User Interfaces.
  - UsiXML consists of a User Interface Description Language (UIDL), that is a declarative language capturing the essence of what a UI is or should be independently of physical characteristics.
  - UsiXML describes at a high level of abstraction the constituting elements of the UI of an application: widgets, controls, containers, modalities, interaction techniques, ...
  - UsiXML supports device independance
  - UsiXML supports platform independance
  - UsiXML supports modality independance.
  - UsiXML allows reuse of elements previously described in anterior UIs to compose a UI in new applications.

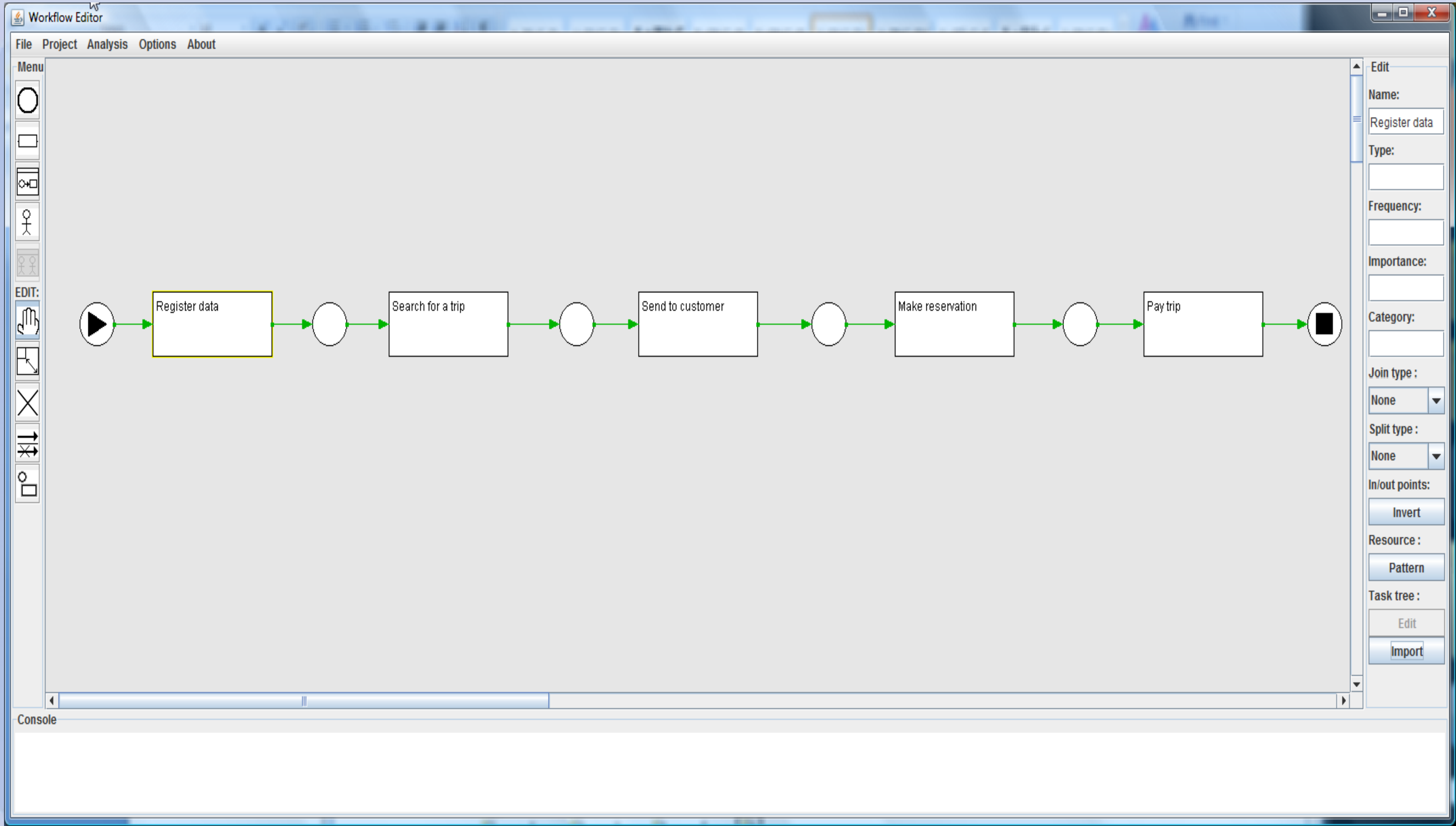
What is not UsiXML

- The coverage of UsiXML in terms of target UIs is large. However, it is not supposed to cover all features of all types of UI. Therefore,
  - UsiXML does not want to introduce yet another language for UI implementation. Instead, it proposes the integration of some of these formats: cHTML, WML, HTML, XHTML, VoiceXML, VRML, Java, C++,…. It is up to the underlying implementation to support the transformation of UsiXML into such a format.
  - UsiXML does not describe the low-level details of elements involved in the various modalities, such as operating system attributes, events, and primitives.
  - UsiXML cannot be rendered nor executed by its own: it relies on an implementation in any third-party rendering engine.
  - UsiXML does not want to support all attributes, events, and primitives of all widgets existing in nearly all toolkits. Instead, it is intended to support a common subset of them.

Language Engineering approach

- UsiXML is different from a pure UI authoring language as it could be used as a specification language.
- The ultimate goal is not only to generate code, but also to have the capability of reasoning about the UI specifications:
  - model checking
  - UI evaluation
  - model-driven engineering
  - maintenance of repository of UI cases or patterns
  - static and dynamic analysis
  - model testing

An example

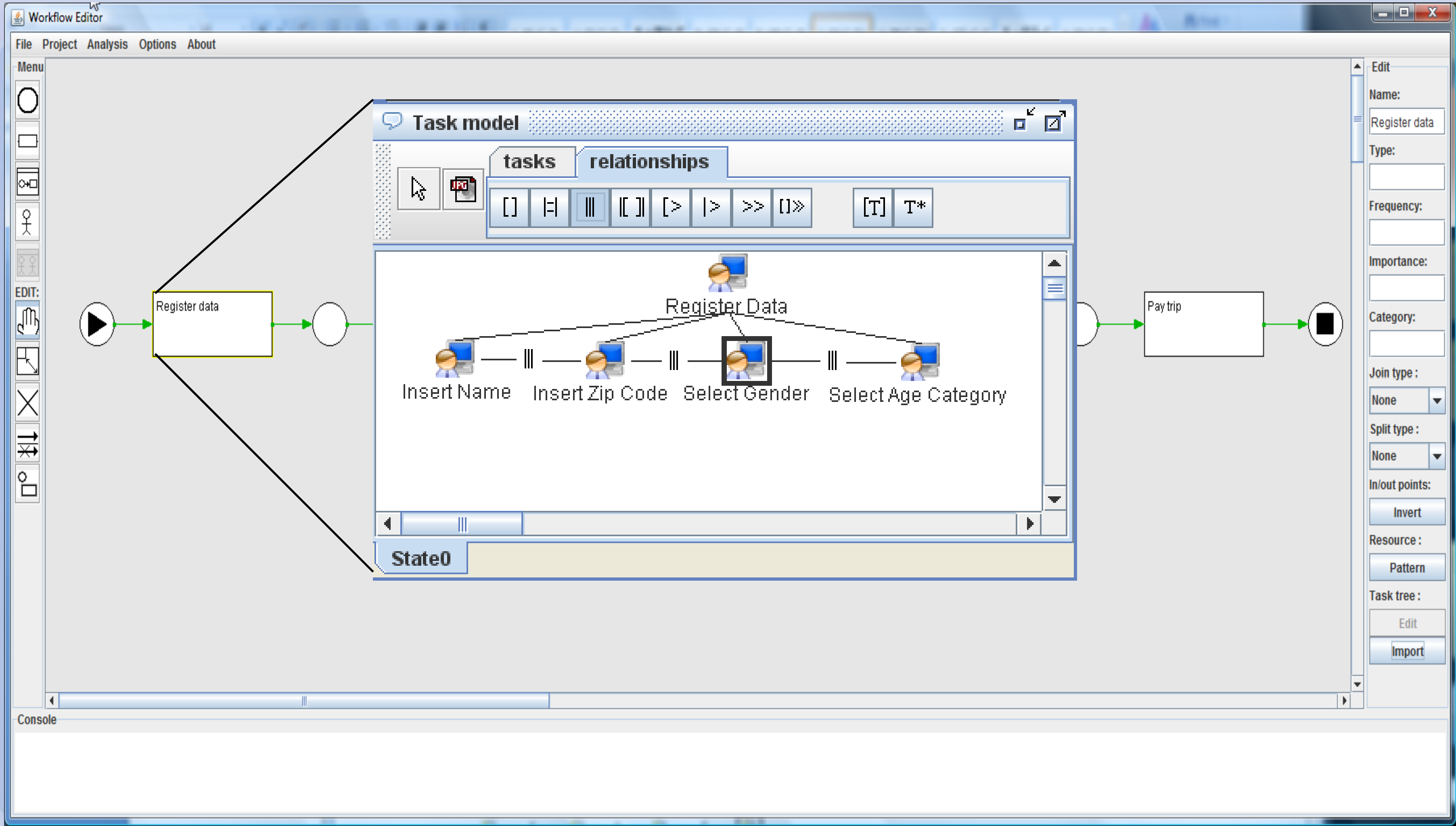


Workflow Editor
File Project Analysis Options About
Menu
EDIT:
Register data
Search for a trip
Send to customer
Make reservation
Pay trip
Edit
Name:
Register data
Type:
Frequency:
Importance:
Category:
Join type :
None
Split type :
None
In/out points:
Invert
Resource :
Pattern
Task tree :
Edit
Import
Console

XML of the Process Model

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ProcessModel>
  - <Process>
    - <joinPattern type="sequential">
        <source taskId="4" taskName="Make reservation" />
        <destination taskId="6" taskName="Pay trip" />
      </joinPattern>
    - <joinPattern type="sequential">
        <source taskId="3" taskName="Send to customer" />
        <destination taskId="4" taskName="Make reservation" />
      </joinPattern>
    - <splitPattern type="sequential">
        <source taskId="4" taskName="Make reservation" />
        <destination taskId="6" taskName="Pay trip" />
      </splitPattern>
    - <joinPattern type="sequential">
        <source taskId="2" taskName="Search for a trip" />
        <destination taskId="3" taskName="Send to customer" />
      </joinPattern>
    - <splitPattern type="sequential">
        <source taskId="3" taskName="Send to customer" />
        <destination taskId="4" taskName="Make reservation" />
      </splitPattern>
    - <joinPattern type="sequential">
        <source taskId="1" taskName="Register data" />
        <destination taskId="2" taskName="Search for a trip" />
      </joinPattern>
    - <splitPattern type="sequential">
        <source taskId="2" taskName="Search for a trip" />
        <destination taskId="3" taskName="Send to customer" />
      </splitPattern>
    - <splitPattern type="sequential">
        <source taskId="1" taskName="Register data" />
        <destination taskId="2" taskName="Search for a trip" />
      </splitPattern>
    </Process>
  </ProcessModel>
```

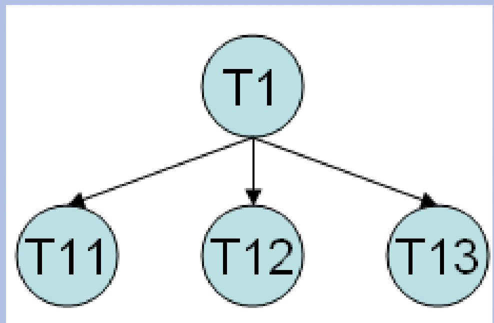
An example – Task Model

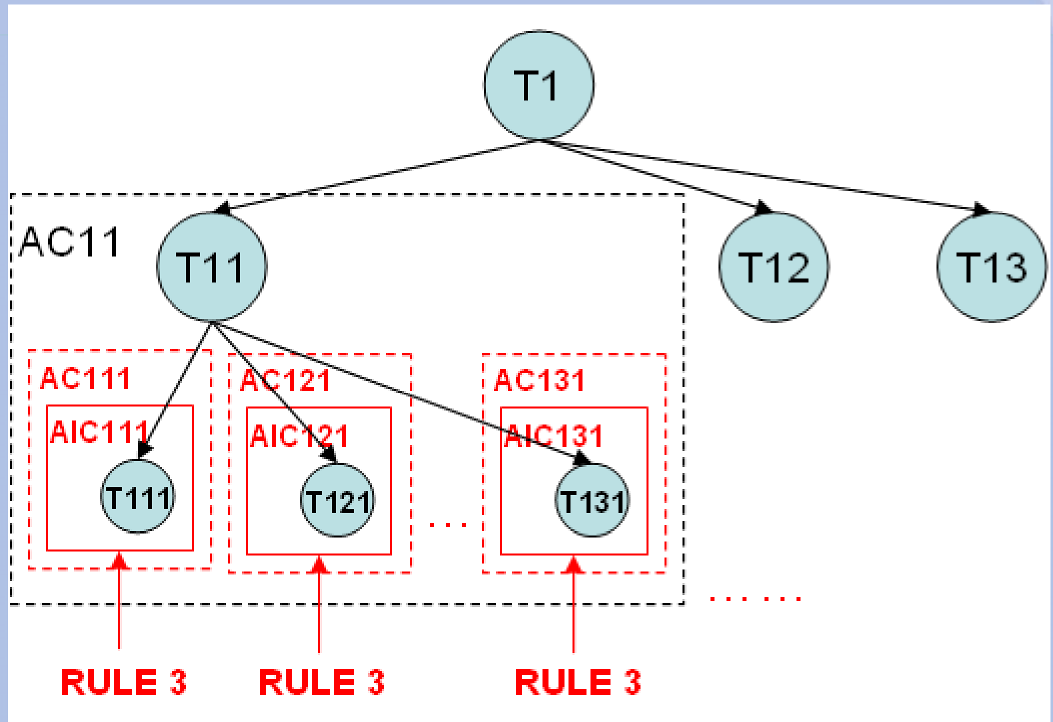


XML of the task Model

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<taskmodel>
  <!-- Tareas  -->
- <task id="st0task0" name="Register Data" type="interaction">
    <task id="st0task1" name="Insert Name" type="interaction" />
    <task id="st0task2" name="Insert Zip Code" type="interaction" />
    <task id="st0task3" name="Select Gender" type="interaction" />
    <task id="st0task4" name="Select Age Category" type="interaction" />
  </task>
  <!-- Relaciones entre tareas  -->
- <orderIndependence>
    <source sourceId="st0task1" />
    <target targetId="st0task2" />
  </orderIndependence>
- <orderIndependence>
    <source sourceId="st0task2" />
    <target targetId="st0task3" />
  </orderIndependence>
- <orderIndependence>
    <source sourceId="st0task3" />
    <target targetId="st0task4" />
  </orderIndependence>
</taskmodel>
```

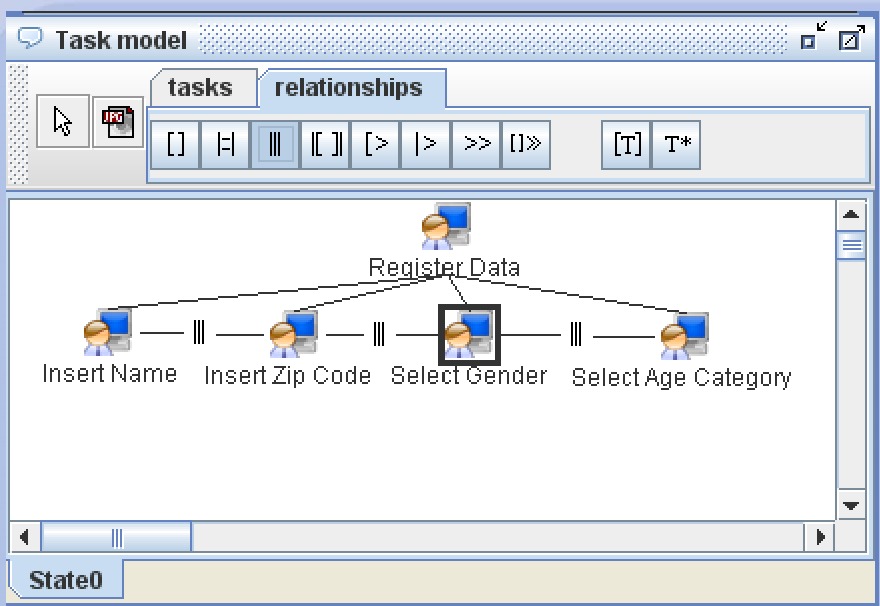
From task to AUI

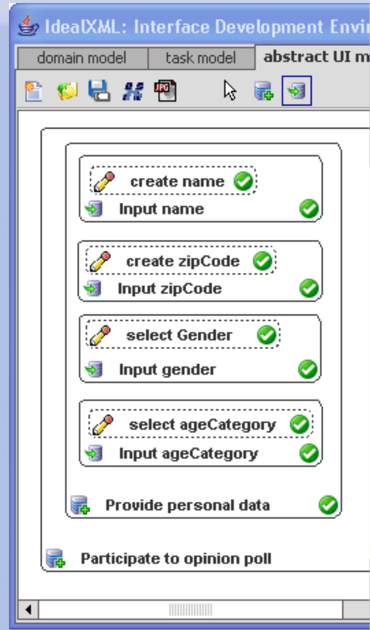




- AC= Abstract Container
- AIC=Abstract Individual Component

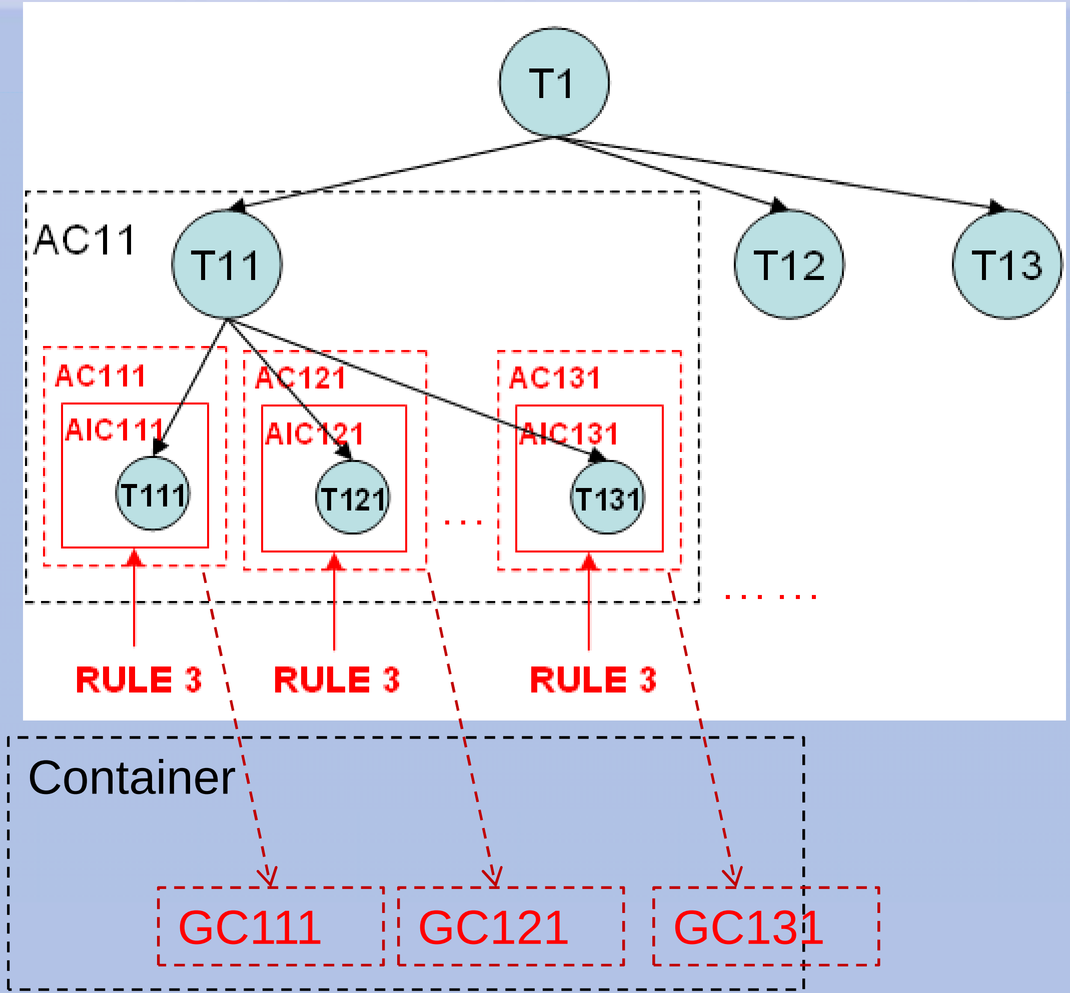
From task to AUI



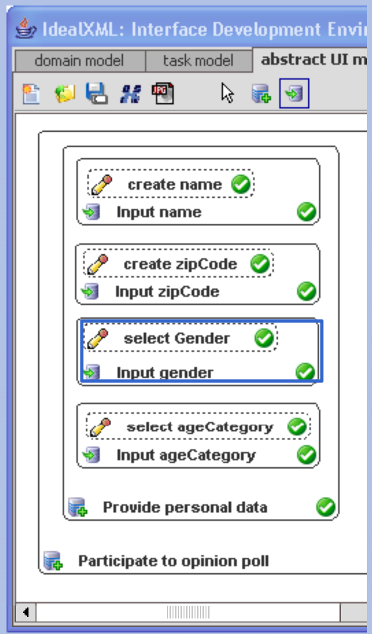


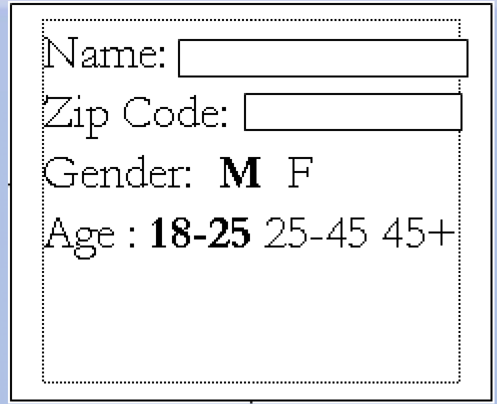
- Abs.container
- Abs. component
- Input
- Output
- Navigation
- Control
- Select

From AUI to CUI



From AUI To CUI

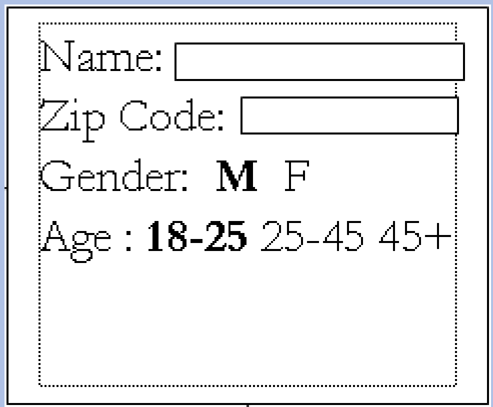




When an **input facet** of type **select element** is encountered the rule generates a GC of type **box** that embed:

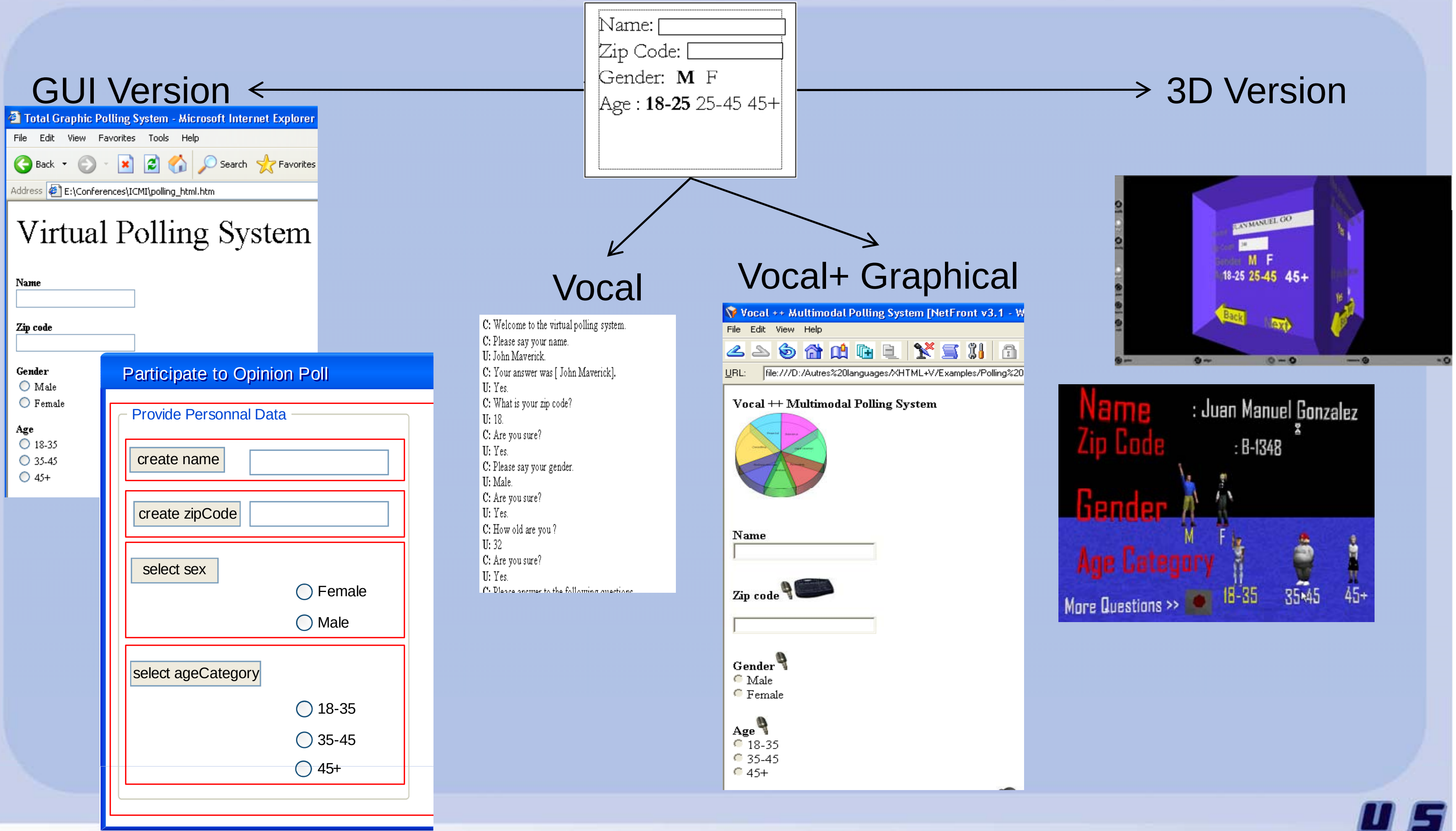
- Three GICs: a **radio button group with radio buttons with the options** and an **outputText** (the label associated to the radioButton group)

CUI

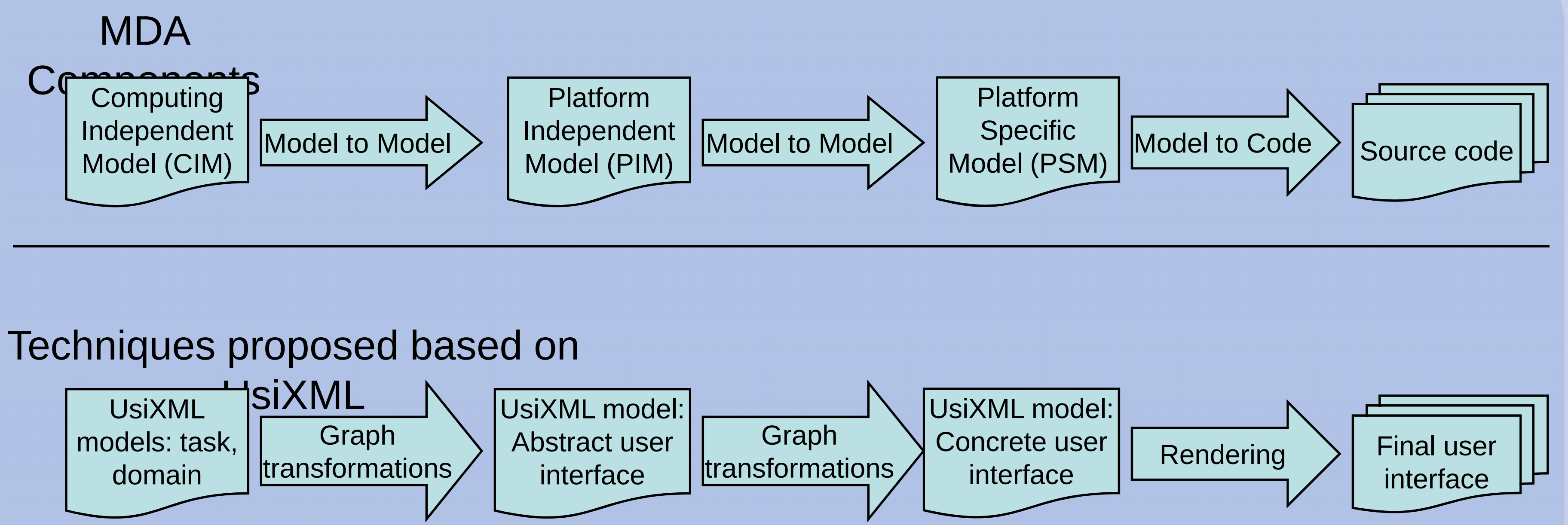



```xml
<abstractContainer id="idao2" name=“Register Data">
    <abstractIndividualComponent id="idao9" name="Input Zip Code">
        <input id="idao15" name="input zip code" actionType="interaction" dataType="String"
               attributeDomainCharacterization="zipCode" />
    </abstractIndividualComponent>
    <abstractIndividualComponent id="idao10" name="Input Name">
        <input id="idao14" name="input Name" actionType="interaction" dataType="String"
               attributeDomainCharacterization="name" />
    </abstractIndividualComponent>
    <abstractIndividualComponent id="idao11" name="input gender">
        <input id="idao16" name="Select gender" actionType="interaction" dataType="String"
               attributeDomainCharacterization="gender" />
    </abstractIndividualComponent>
    <abstractIndividualComponent id="idao12" name="input age cathegory">
        <input id="idao17" name="input ageCategory" actionType="interaction" dataType="String"
               attributeDomainCharacterization="ageCategory" />
    </abstractIndividualComponent>
```

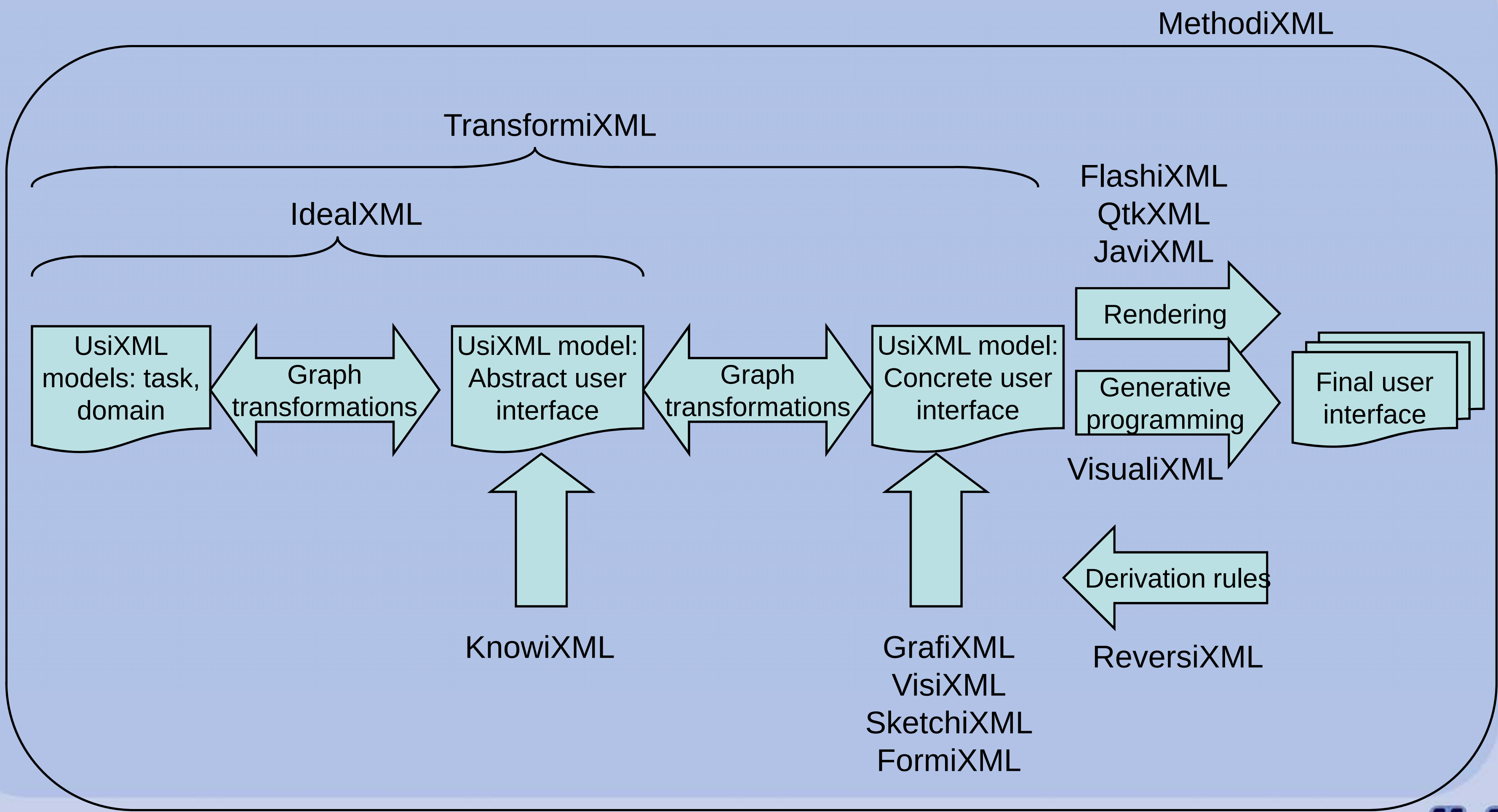
From CUI to FUI



MDE based on UsiXML



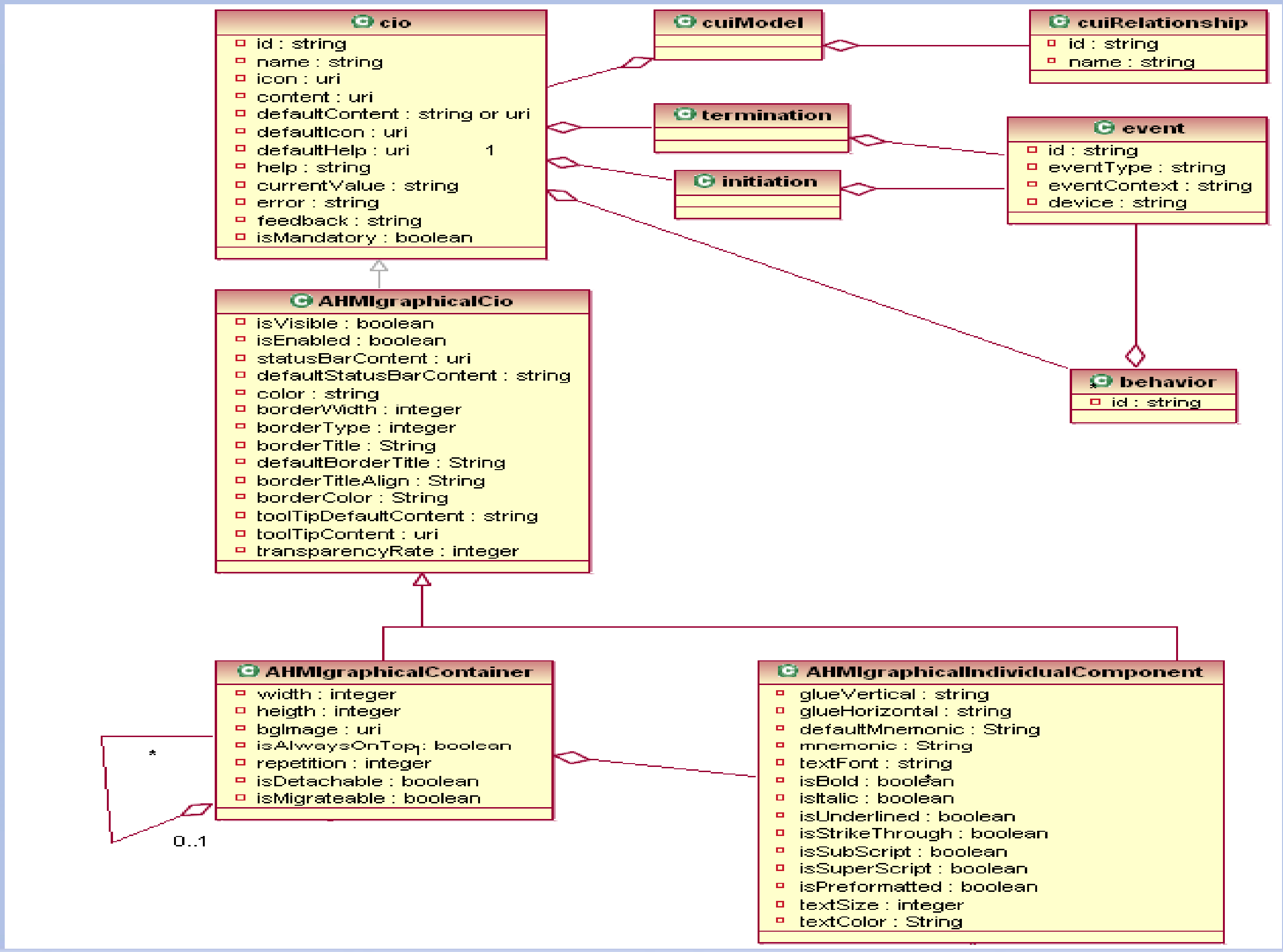
The big picture of MDA supported in UsiXML



Concepts description

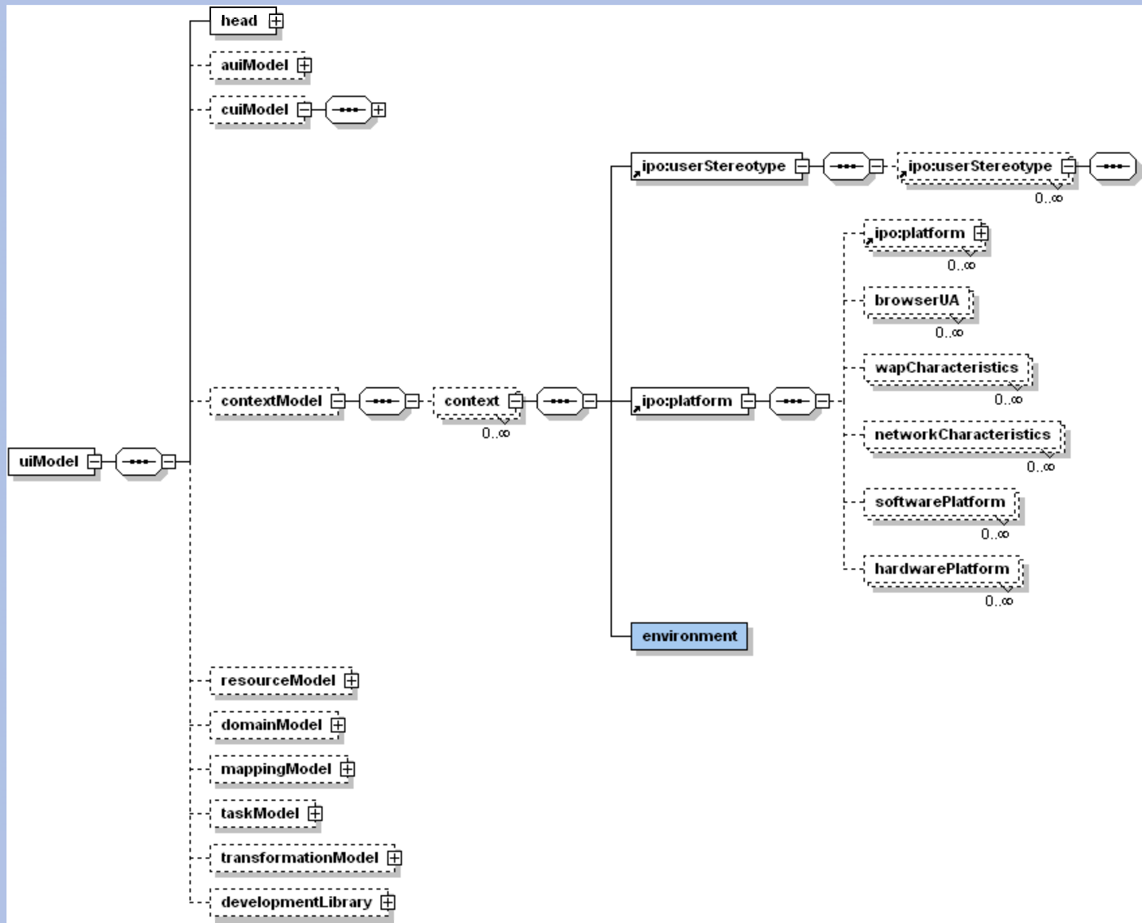
- Three level of description
  - Semantics
  - Syntax
  - Stylistics

Semantics



cio
id : string
name : string
icon : uri
content : uri
defaultContent : string or uri
defaultIcon : uri
defaultHelp : uri
help : string
currentValue : string
error : string
feedback : string
isMandatory : boolean
1
cuiModel
cuiRelationship
id : string
name : string
termination
initiation
event
id : string
eventType : string
eventContext : string
device : string
behavior
id : string
AHMIgraphicalCio
isVisible : boolean
isEnabled : boolean
statusBarContent : uri
defaultStatusBarContent : string
color : string
borderWidth : integer
borderType : integer
borderTitle : String
defaultBorderTitle : String
borderTitleAlign : String
borderColor : String
toolTipDefaultContent : string
toolTipContent : uri
transparencyRate : integer
AHMIgraphicalContainer
width : integer
heigth : integer
bgImage : uri
isAlwaysOnTop : boolean
repetition : integer
isDetachable : boolean
isMigrateable : boolean
*
0..1
AHMIgraphicalIndividualComponent
glueVertical : string
glueHorizontal : string
defaultMnemonic : String
mnemonic : String
textFont : string
isBold : boolean
isItalic : boolean
isUnderlined : boolean
isStrikeThrough : boolean
isSubScript : boolean
isSuperScript : boolean
isPreformatted : boolean
textSize : integer
textColor : String

Abstract Syntax



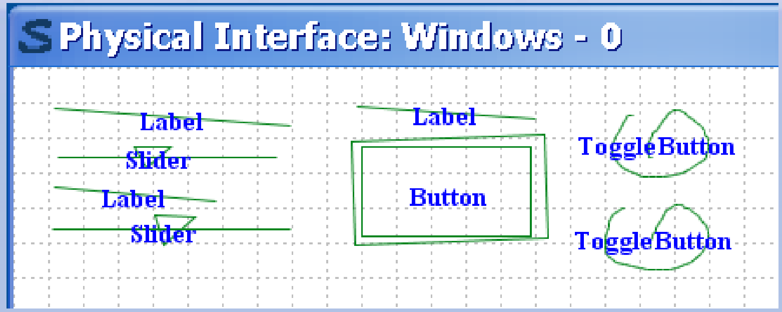
Concrete Syntax

Excerpt for a UsiXML CUI specification

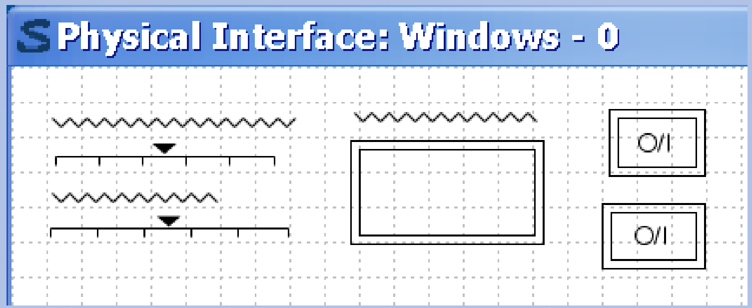
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cuiModel  name="MyModel">
  <version modifDate="2004-03-24T17:09:17.402+01:00" xmlns="">7</version>
  <authorName xmlns="">Youri</authorName>
  <window height="500" width="600" name="Formulaire (2/5)" id="window_1">
    <box relativeHeight="100" name="box1_0" id="box1_0">
       <box type="vert" name="boxTodo" id="boxTodo">
          ...
       <box type="horiz" name="box_2_2_2_1" id="box_2_2_2_1">
         <textComponent defaultContent="Sexe" isBold="true" id="label_2"/>
         <radioButton groupName=“grupo01" defaultContent="Femme"
                    defaultState="false" id="radiobutton_0"/>
         <radioButton groupName="grupo01" defaultContent="Homme"
                    defaultState="true" id="radiobutton_1"/>
       </box>
         ...
       </box>
    </box>
  </window>
</cuiModel>
```

Stylistics (CUI Model)

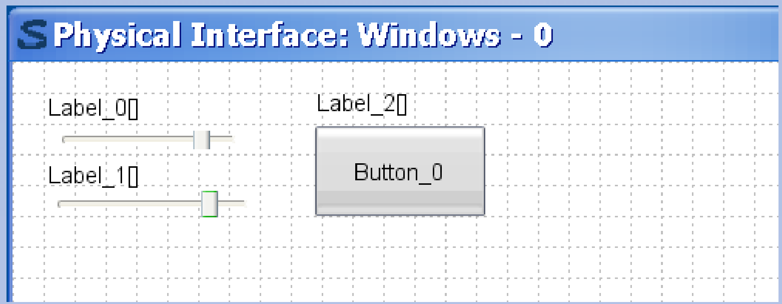
- Low Fidelity



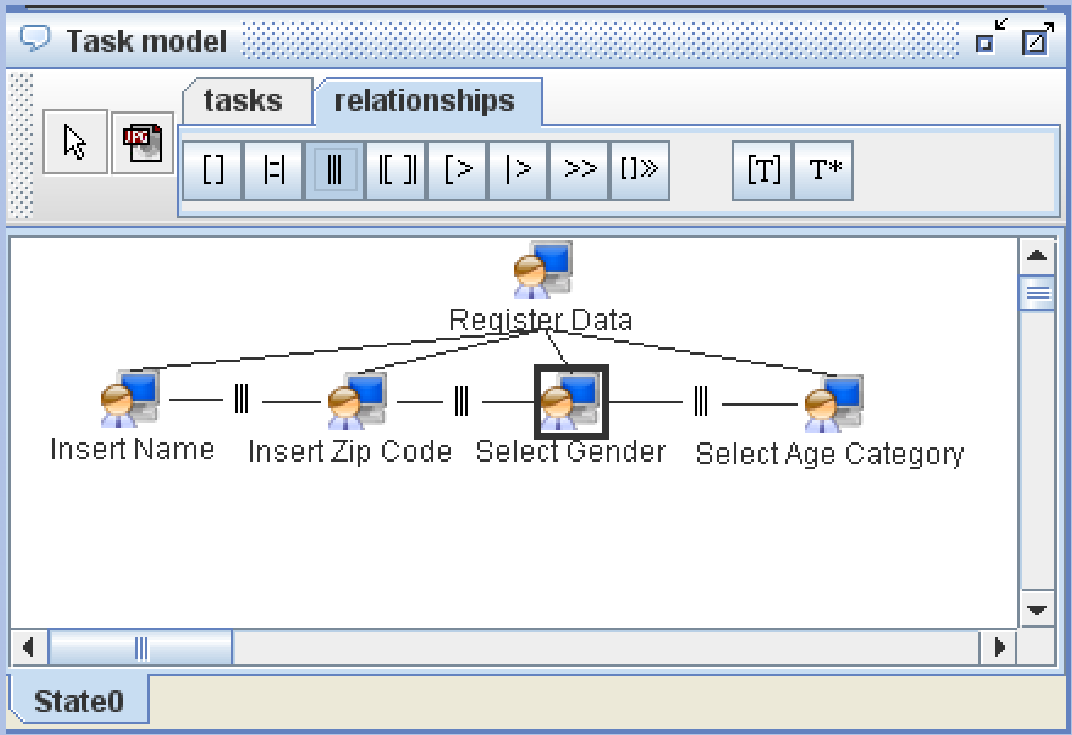
- Medium Fidelity



- High Fidelity



CIM Step 1: Task model



New Abstraction: the user's task

- Task = set of actions carried out by a user in a given context to reach a goal
- Logical decomposition of task into sub-tasks
- Temporal ordering: Extended version of LOTOS operators

– T1 >> T2	Enabling
– T1[ ]>>T2	Enabling + information passing
– T1 |> T2	Suspend/resume
– T1 [ ] T2	Non-deterministic choice
– T1 π T2	Deterministic choice
– T1 [> T2	Disabling (e.g. Form submit)
– T1 |=| T2	Independence (any order, but finished)
– T1*	Iteration
– T1{n}	Finite iteration
– T1 ||| T2	Concurrency
– T1 |[x]| T2	Concurrency + information passing
– [T]	Optional
– T	Recursion

[Markopoulos, 1992]

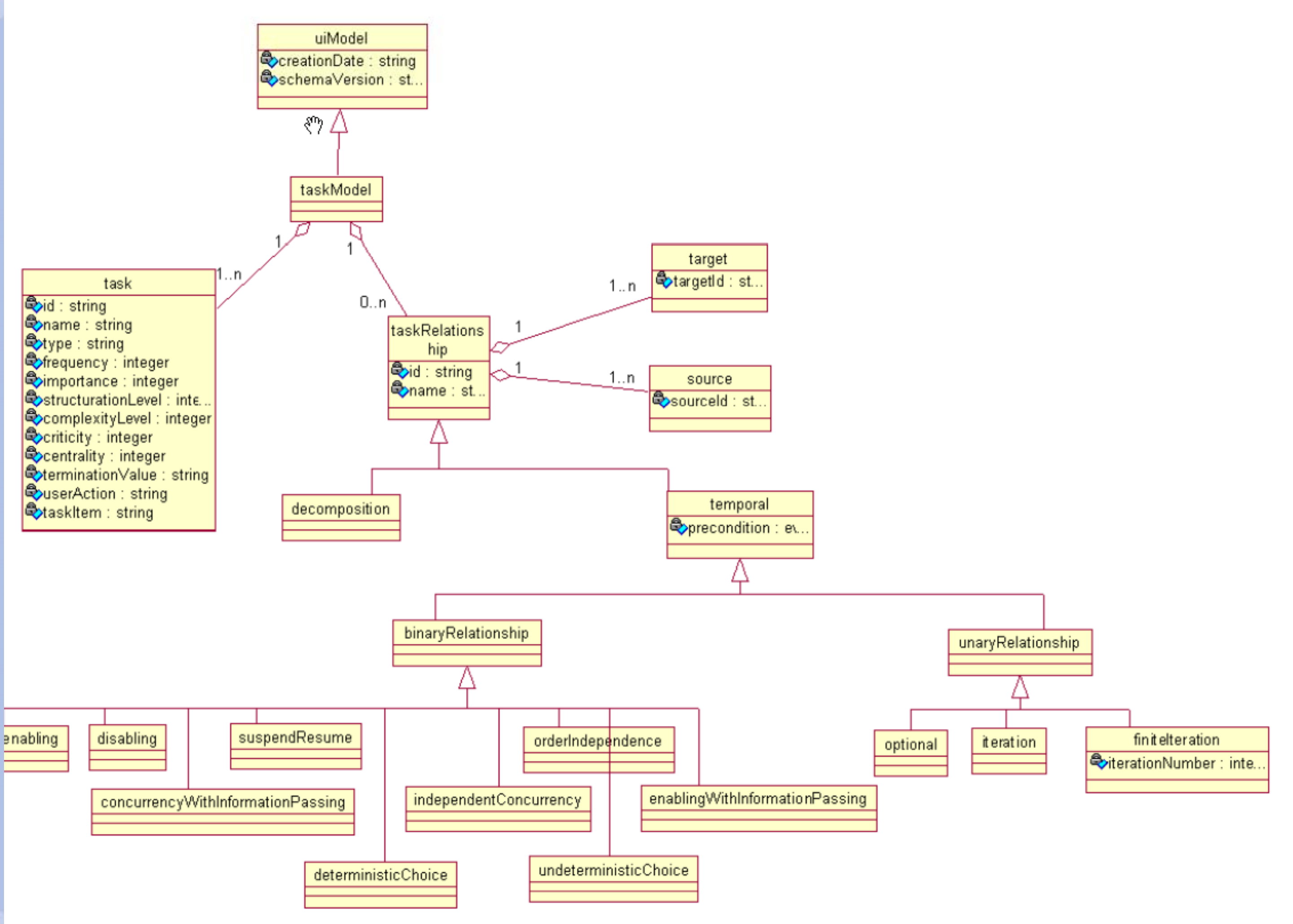
Operators selection

	AMBOSS	ANSI/CEA	Diane +	GOMS	GTA	HTA	TKS	TOOD	UsiXML
Decomposition	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy
Sequence	√ SEQ	+- Ordered = true, information passing (Postcondition)	√ Ordered sequence	√ Sequence	√ Seq	√ Fixed sequence	√ Sequence	√ Sequence	√ Enabling, enabling with information passing
Iteration	X	+- MinOccurs+ MaxOccurs	√ Loop	+- Loop (If, then, else)	X	+- Stop rules	X	X	√ Iteration, finite iteration
Choice	√ ALT	+- Precodition	√ Required choice, free choice	+- Or (If, then, else)	√ Or	√ Selective rule	√ Or	√ Choice	√ Deterministic choice, undeterministic choice, inclusive choice
Optionality	+- Barrier	+- MinOccurs+ MaxOccurs	√ Optional	+- Optional (If, then, else)	+- Start condition	X	X	X	√ Optional

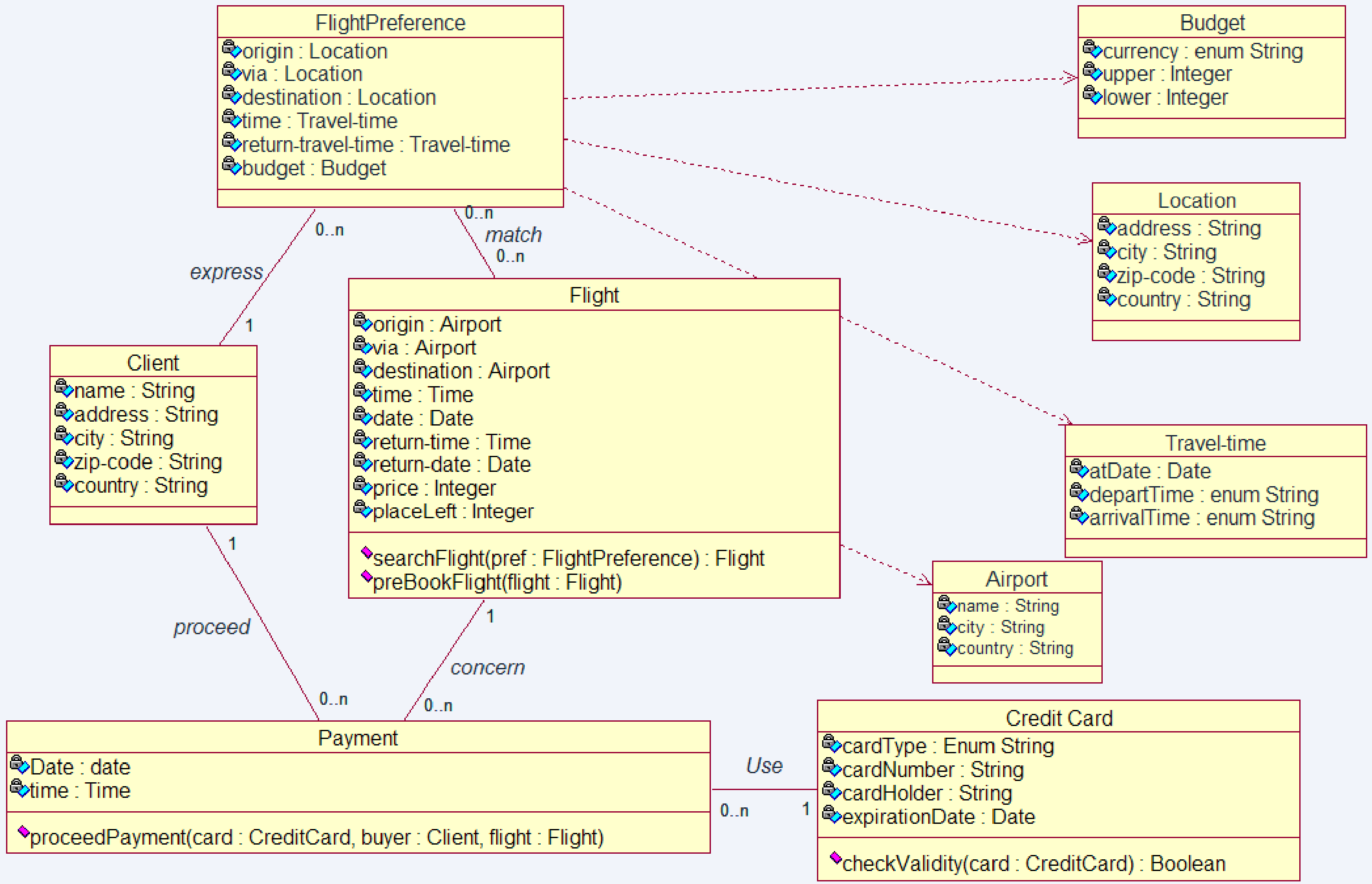
Operators selection (Cont...)

	AMBOSS	ANSI/CEA	Diane +	GOMS	GTA	HTA	TKS	TOOD	UsiXML
Interruption	X	X	X	+- Optional (If, then, else)	+- Stop condition	√ Stop rules	X	√ Interruption	√ Suspend-resume, disabling, disabling with information passing
Concurrency	√ SER	+- Ordered = false	√ unordered sequence	+- Concurrency (If, then, else)	X	+- Selective rule	X	√ Concurrency	√ Independent concurrency, concurrency with information passing, order independence
Cooperation	+- Precondition	X	X	X	√ Cooperation	+- Teamwork	√ Collaboration(FKS extension)	√ Collaboration	√ Cooperation
Parallel	√ PAR, SIM	X	√ Parallel	X	√ And	√ Dual task (time sharing)	√ And	√ Simultaneity	√ parallelSplit (process model)

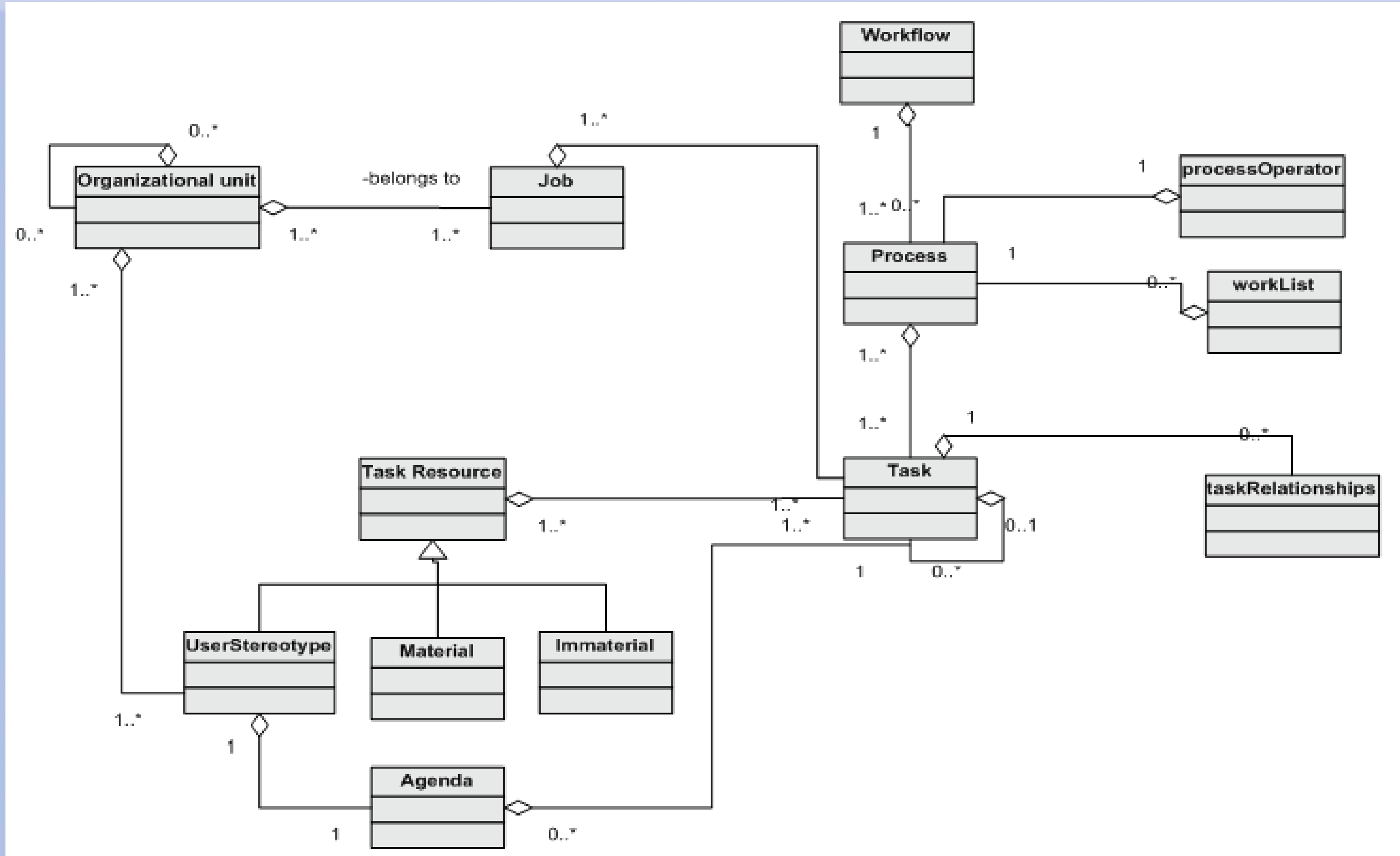
New Abstraction: the task meta-model



CIM Step 2: domain model



Organizational Modeling



Task Identification criteria

	Time	Space (location)	Resource	Type
Workflow	Series of time periods	Different locations; same organization	Same or different groups of resources	-
Process	Series of time periods	Different locations	Within groups, group as a whole, or among groups	Primary (production), secondary (support), or tertiary (managerial)
Task	Same time period	Same location	One or two types of resources	User, interactive, system, abstract, or machine task

Resource task allocation patterns

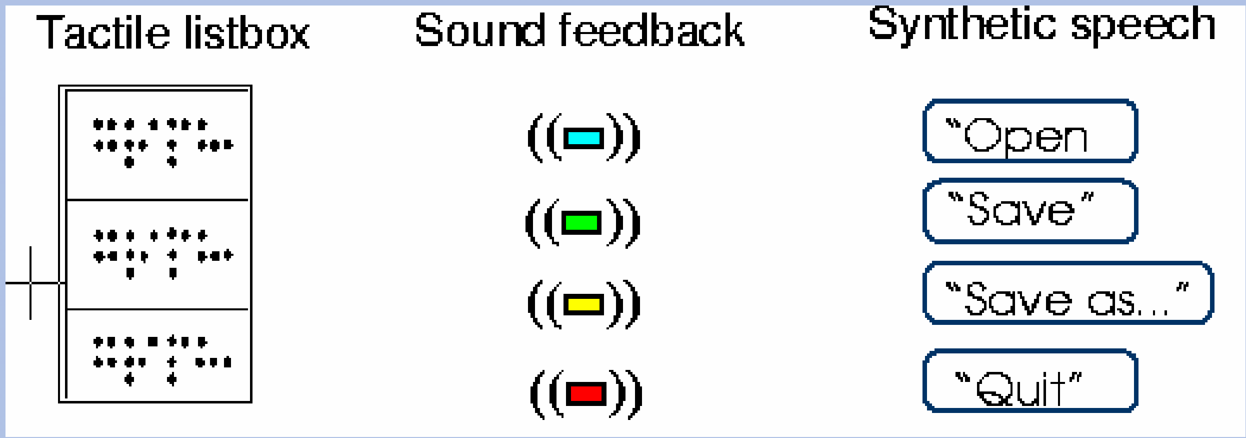

```
- <mappingModel>
  - <resourcePatterns taskId="6" taskName="Pay trip">
      <creationType name="Direct" />
      <distributionType name="Offer to single resource" />
      <distributionTime name="Early" />
      <executionType name="Resource-initiated execution" />
      <detours Delegation="unallowed" Escalation="unallowed" Deallocation="unallowed" Statefull_realocation="unallowed"
        Stateless_realocation="unallowed" Suspension-resumption="unallowed" Skip="unallowed" Redo="unallowed" Pre-do="unallowed" />
      <options Configurable_unallocated_work_items_visibility="unallowed" Configurable_allocated_work_items_visibility="unallowed"
        Simultaneous_execution="unallowed" Additional_resources="unallowed" />
    </resourcePatterns>
  - <resourcePatterns taskId="4" taskName="Make reservation">
      <creationType name="Direct" />
      <distributionType name="Offer to single resource" />
      <distributionTime name="Early" />
      <executionType name="Resource-initiated execution" />
      <detours Delegation="unallowed" Escalation="unallowed" Deallocation="unallowed" Statefull_realocation="unallowed"
        Stateless_realocation="unallowed" Suspension-resumption="unallowed" Skip="unallowed" Redo="unallowed" Pre-do="unallowed" />
      <options Configurable_unallocated_work_items_visibility="unallowed" Configurable_allocated_work_items_visibility="unallowed"
        Simultaneous_execution="unallowed" Additional_resources="unallowed" />
    </resourcePatterns>
```

New Abstraction: the user's task

- Task definition = action + object
  - Action types
    - CRUD pattern: create, read, update, delete
    - Select, control,...
    - Acquire, render, modify, publish, compute, derive,...
  - Object types:
    - Element, list, table, collection, compound,...

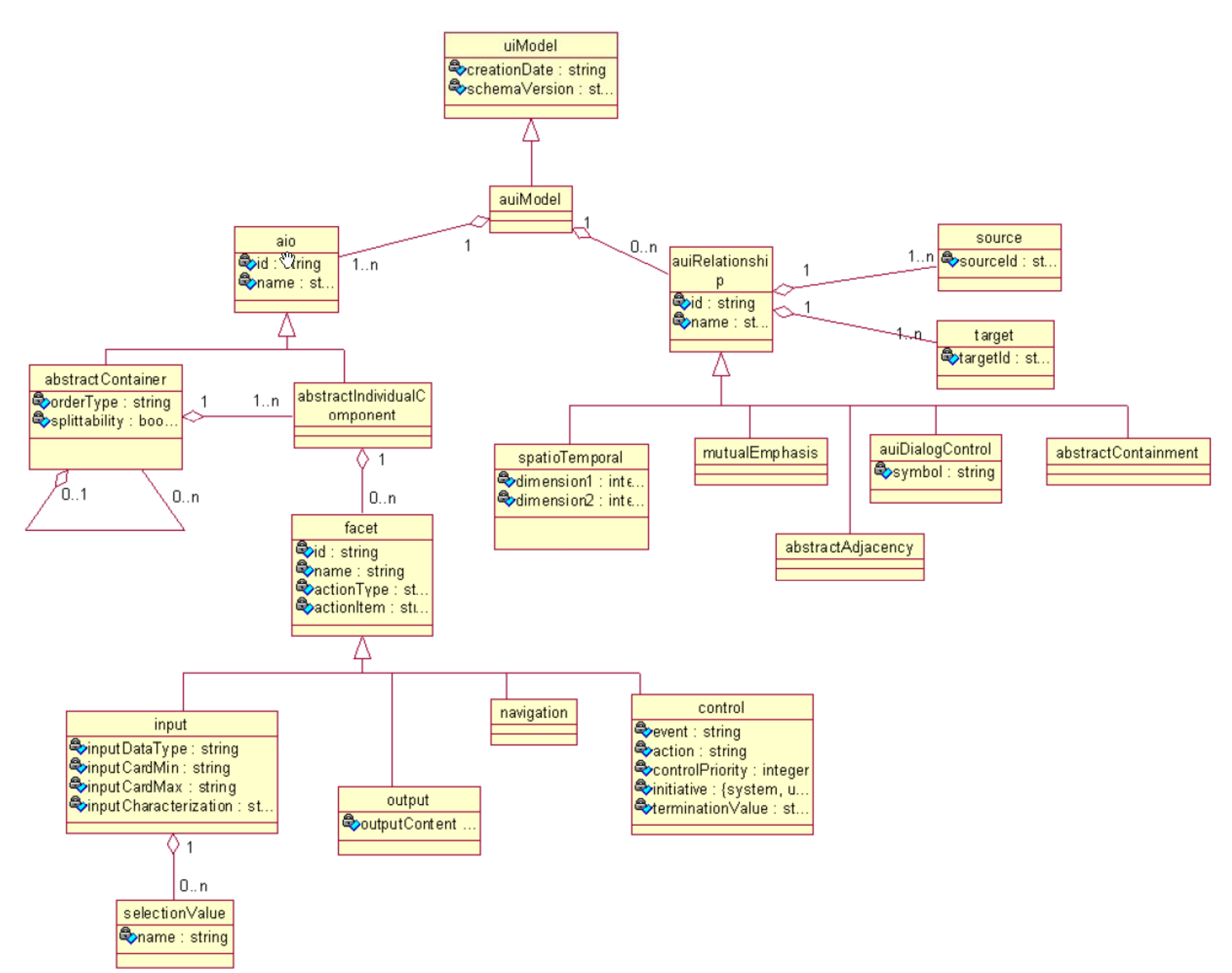
New Abstraction: the abstract UI

- Different CIOs can be used for the same purpose, but with different interaction modalities
- Definition
  - Abstract Container = set of Abstract Individual Component
  - AIC = abstraction of CIOs of the same type, but independently of any interaction modality
  - Abstract User Interface (AUI) = decomposition into AC+AIC



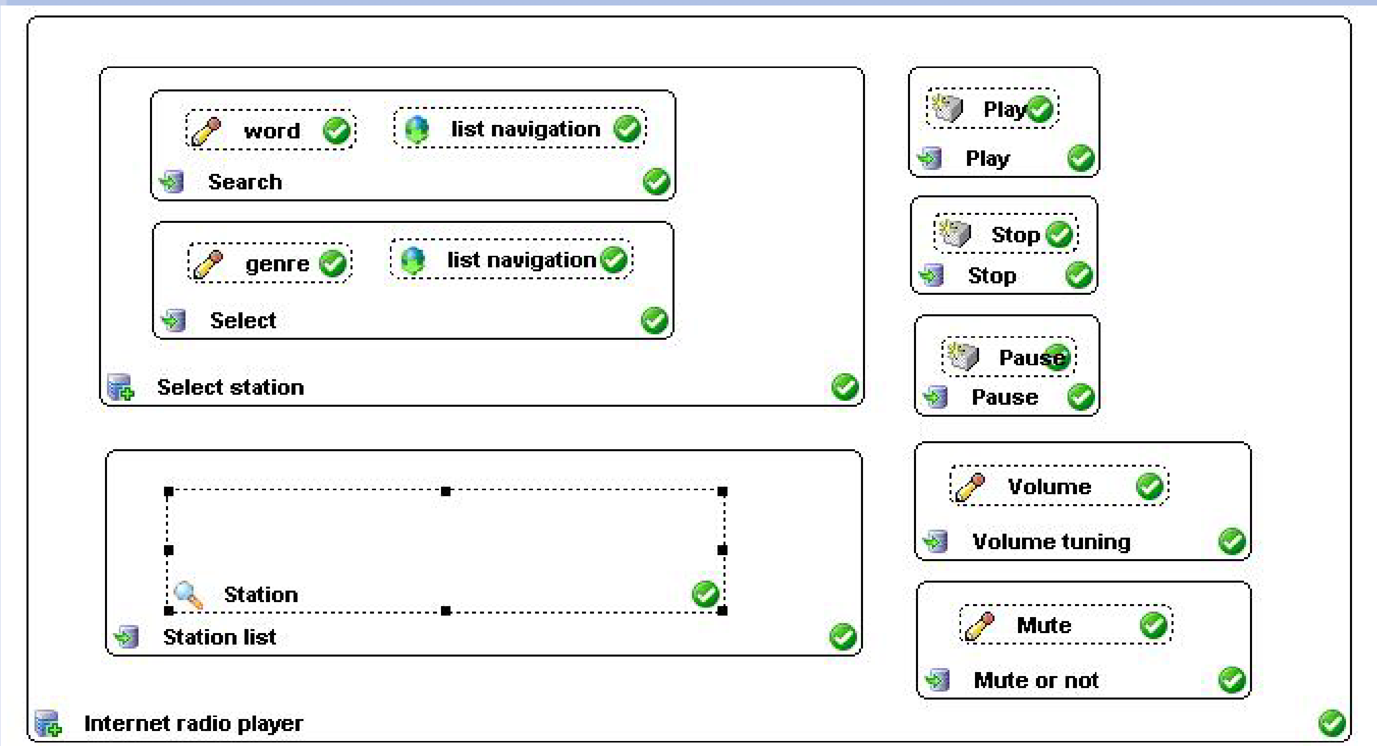
[Vanderdonckt & Bodart,1993]

Abstraction: the abstract UI



Abstraction: the abstract UI

- Notation: based on L. Constantine's notation for canonical abstract prototypes [Constantine,2003]

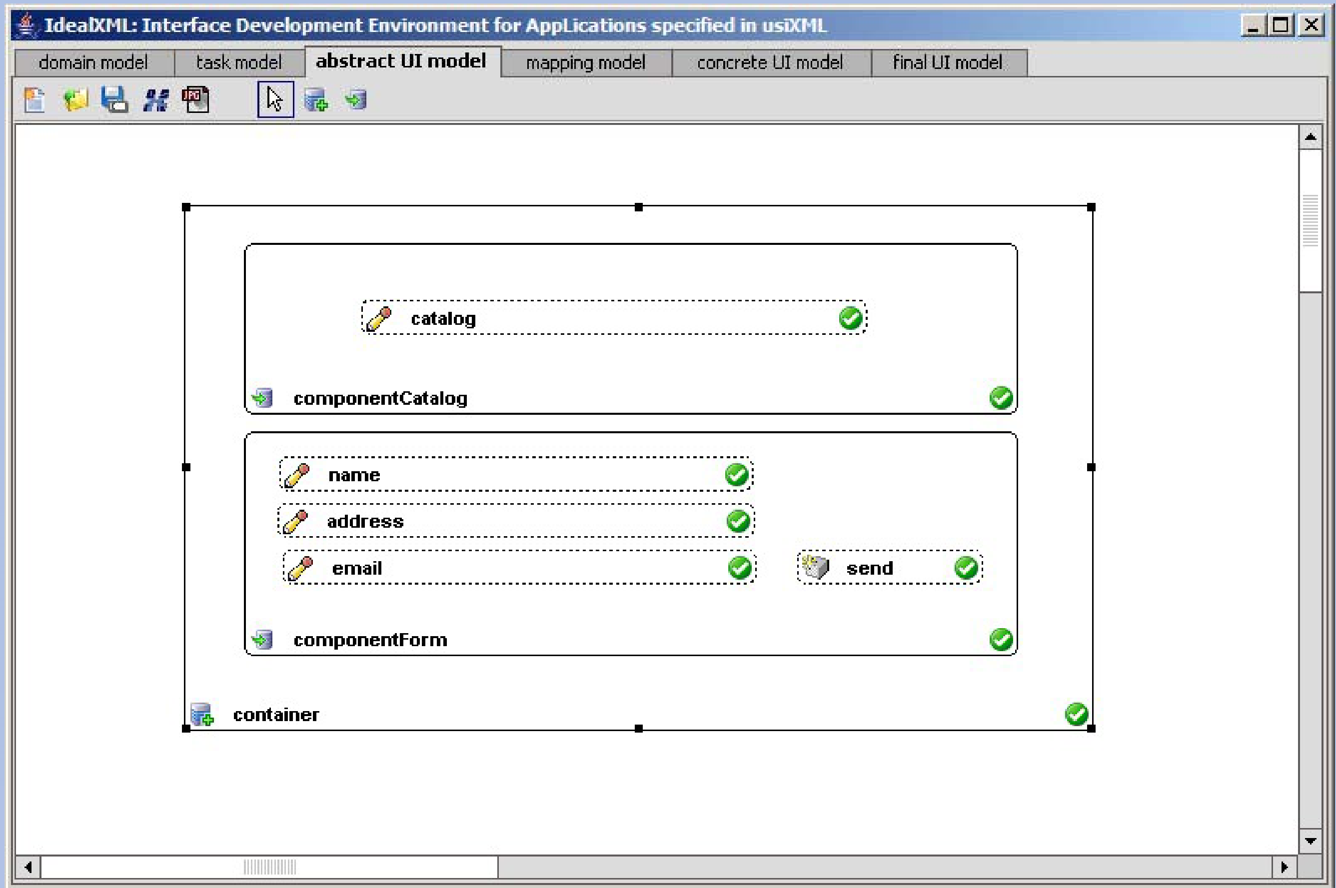


- Abs.container
- Abs. compone
- Input
- Output
- Navigation
- Control
- Select

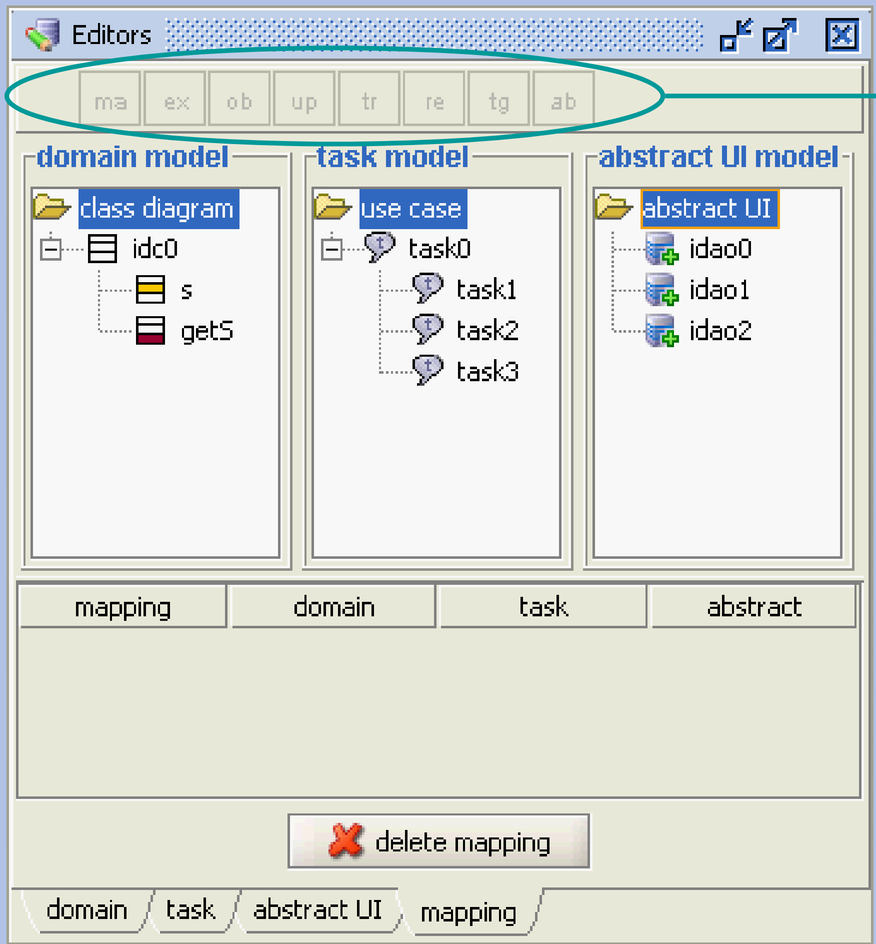
IdealXML

[Montero et al.,2005]

Example of AUI produced



Mapping the models

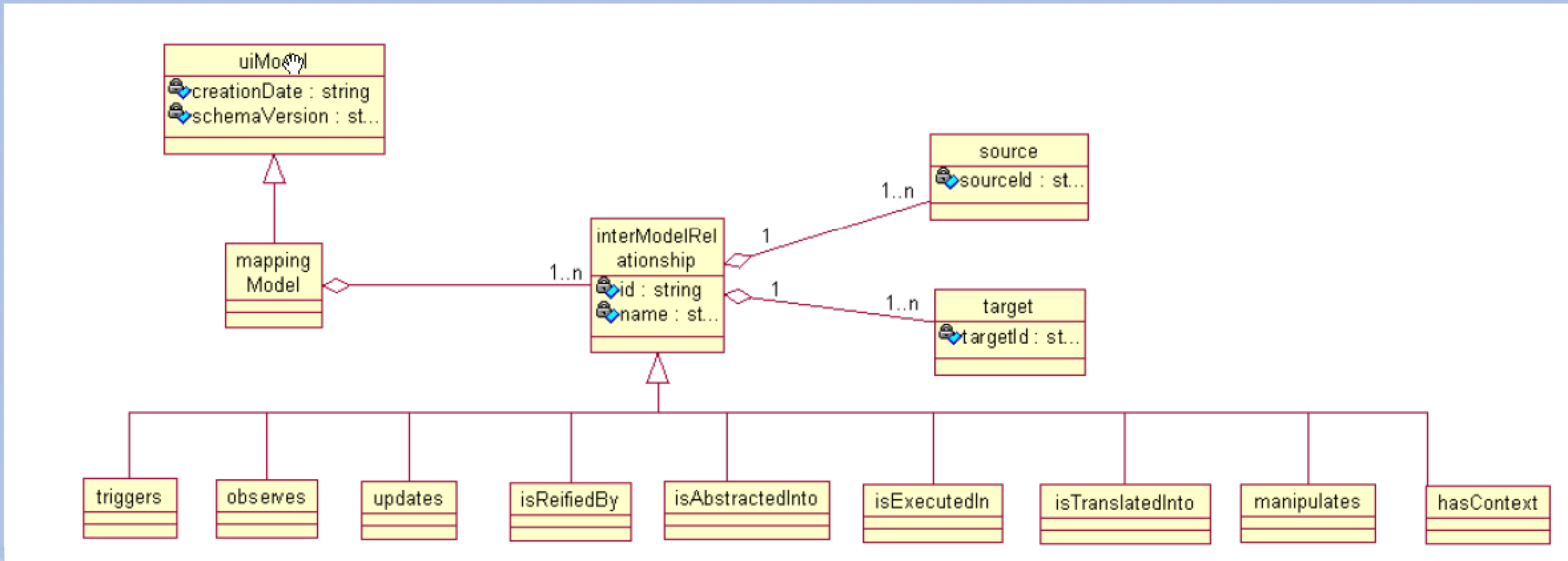


These mappings can be established:

- **triggers (tg):** { , } x
- **updates (up):** x
- **observes (ob):** x
- **isExecutedIn (ex):** x
- **manipulates (ma):** { , } x

Mapping the models

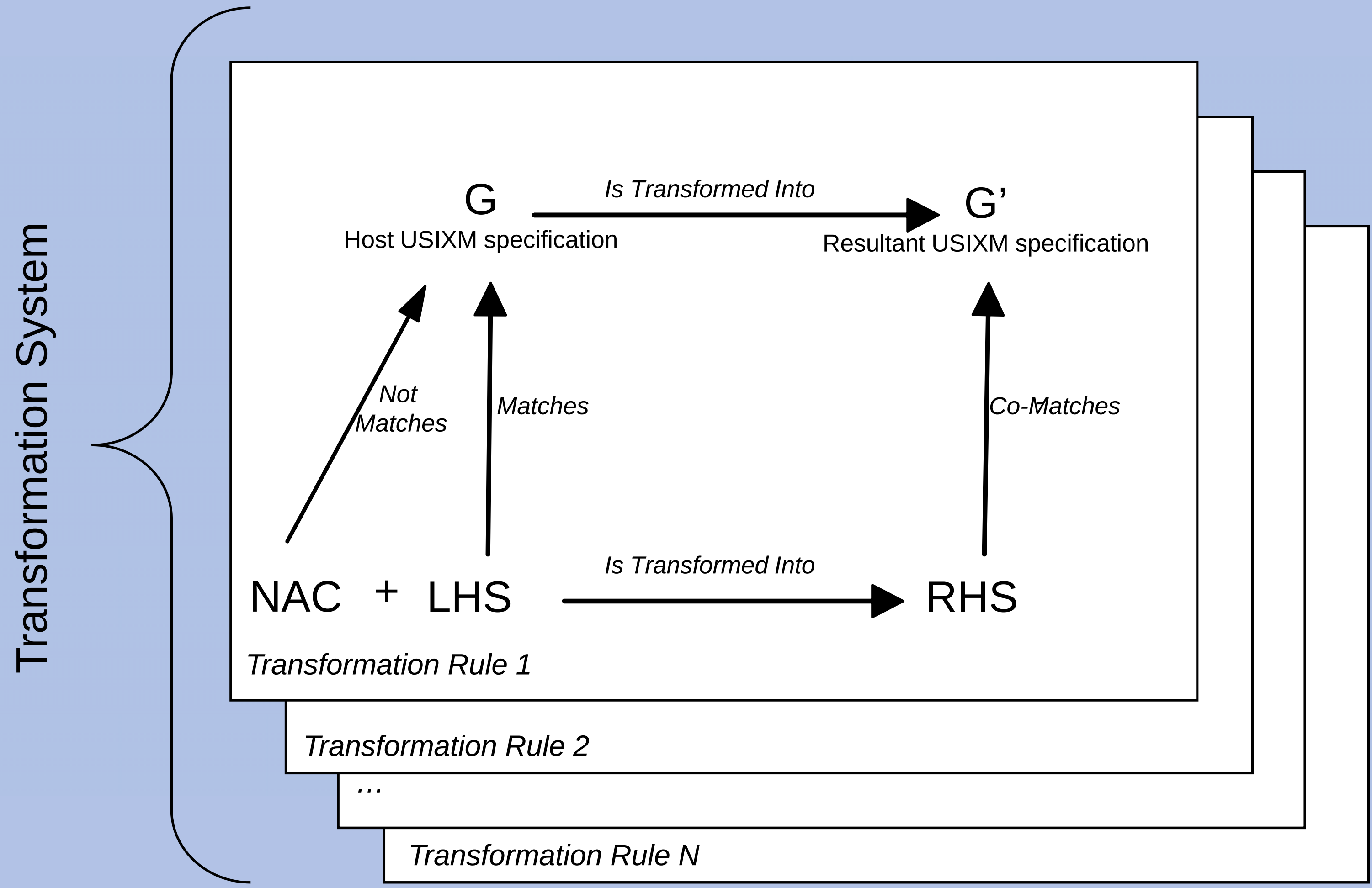
- Mapping the models with a mapping model (!!)



Expression of models as graphs

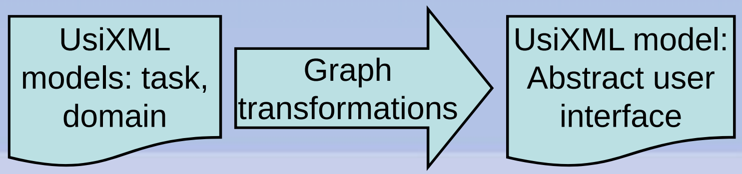
- All transformations are in UsiXML
  - Each model = instance of meta-model
  - Each model = graph as instance of graph type
  - Each model transformation =
    - graph transformation
    - Set of productions

Transformation system



PIM step: task+domain to AUI

- Abstract UI (AUI) = UI independent of any interaction modality
- Definition of AUI structure in terms of Abtract Containers (AC)
  - Which tasks should be logically grouped?
- Definition of Abstract Individual Components (AIC) types
  - Which « functionnality » should assume AICs and what data do they manipulate ?
- Definition of spatio-temporal arrangement
  - How should AIC be arranged in space and time ?
- Definition of dialog control
  - What is the valid flow of action on AICs ?



PIM step: task+domain to AUI

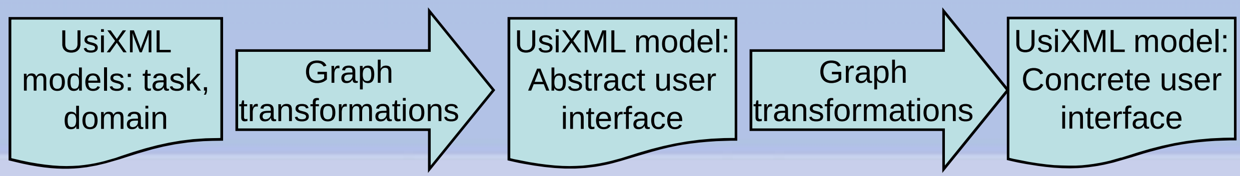
STEP : From Task & Domain to AUI

SUB-STEPS

1. Identification of AUI structure
2. Selection of AIC
3. Spatio-Temporal Arrangement of AIOs
4. Definition of Abstract Dialog Control
5. Derivation of AUI to Domain Relationships

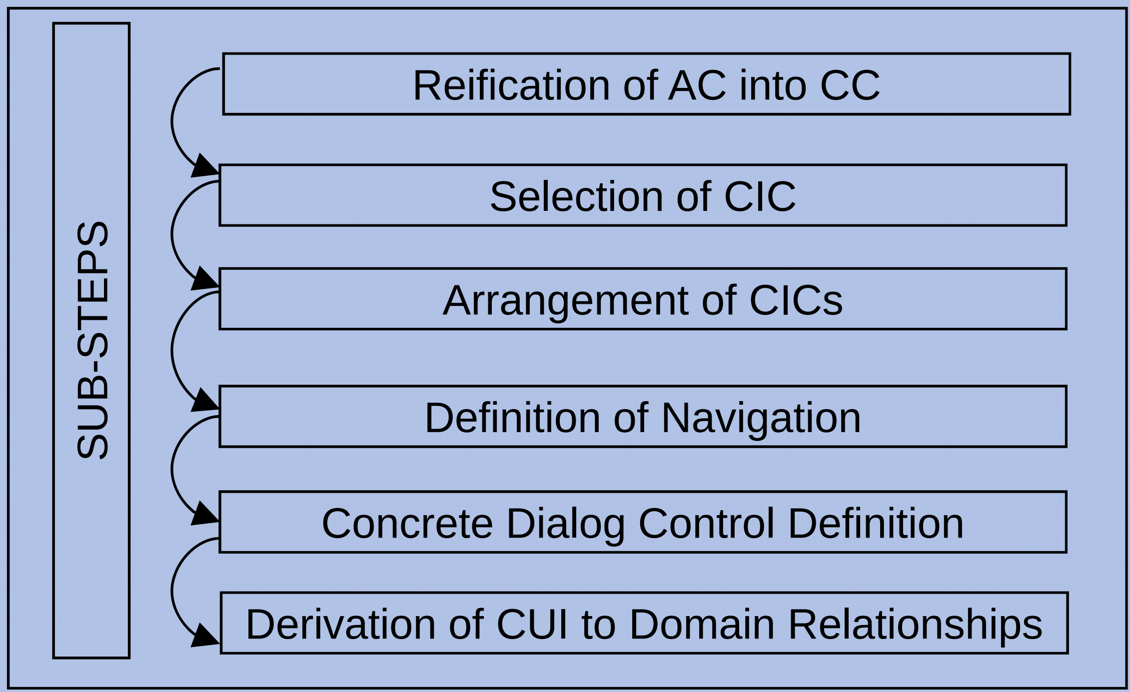
PSM Step: AUI to CUI

- Concrete UI (CUI) = UI independent of toolkit
- Definition of CUI structure
  - Which AIC is a window?
- Definition of Concrete Interaction Component (CIC) type
  - Which « widget » should represent which AIC ?
- Definition of placement
  - What layout can be specified between CICs,...
- Definition of navigation
  - Which container can be started or closed from which container?
- Definition of dialog control
  - What is the valid flow of action on AIOs

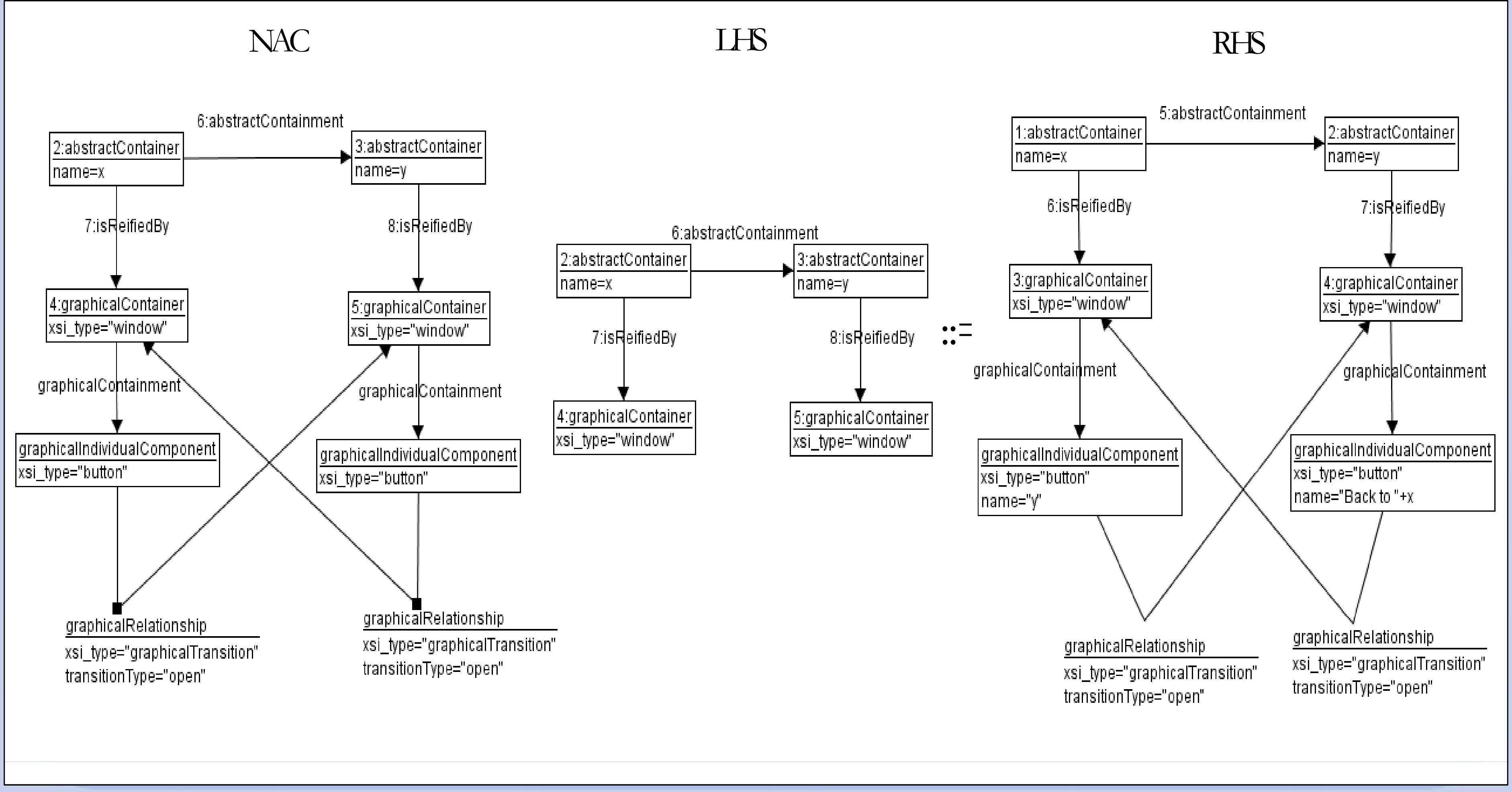


PSM Step: AUI to CUI

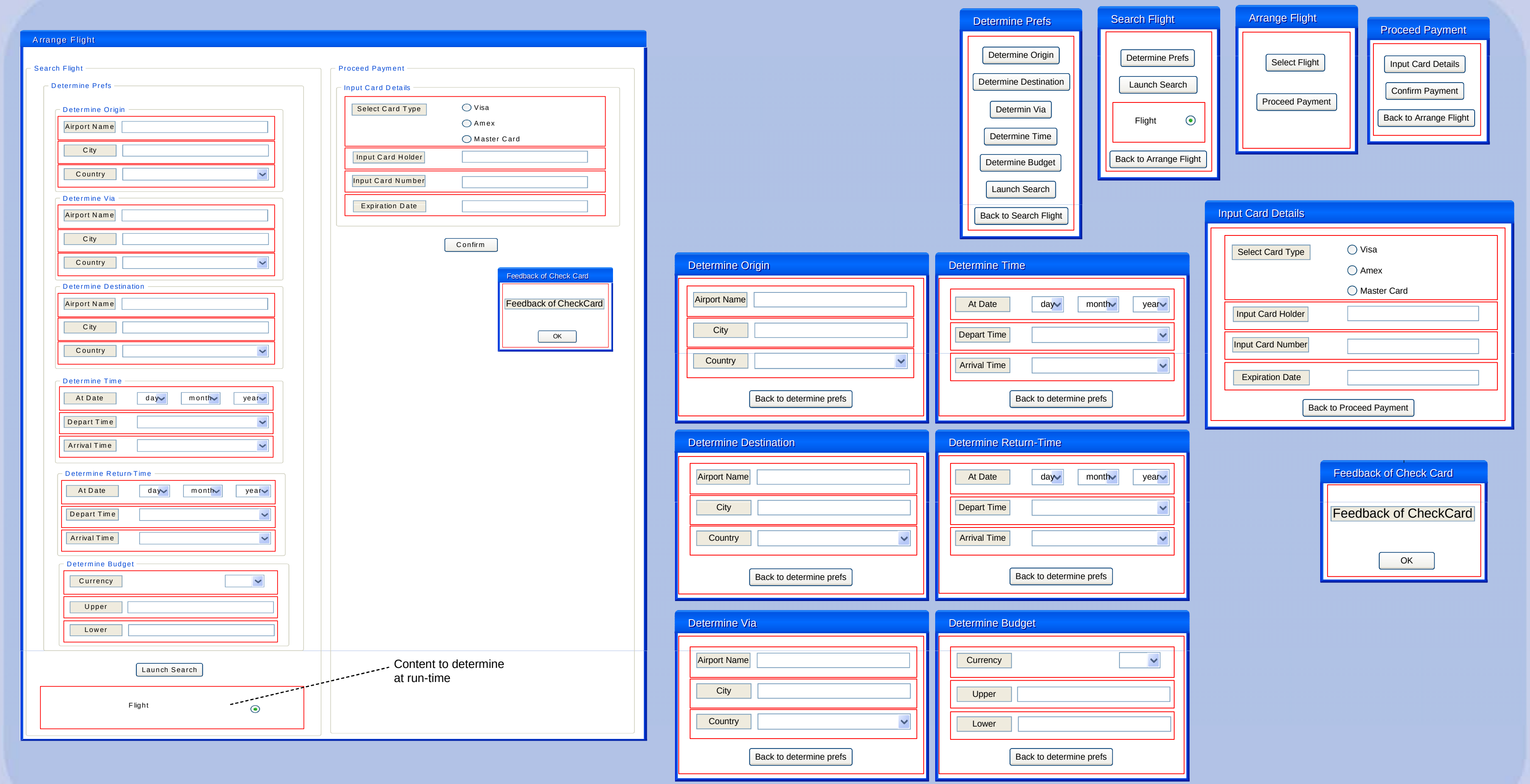
STEP : From AUI to CUI



PSM sub-step 3: definition of navigation
An example of a complex rule



PSM: Concrete User Interface

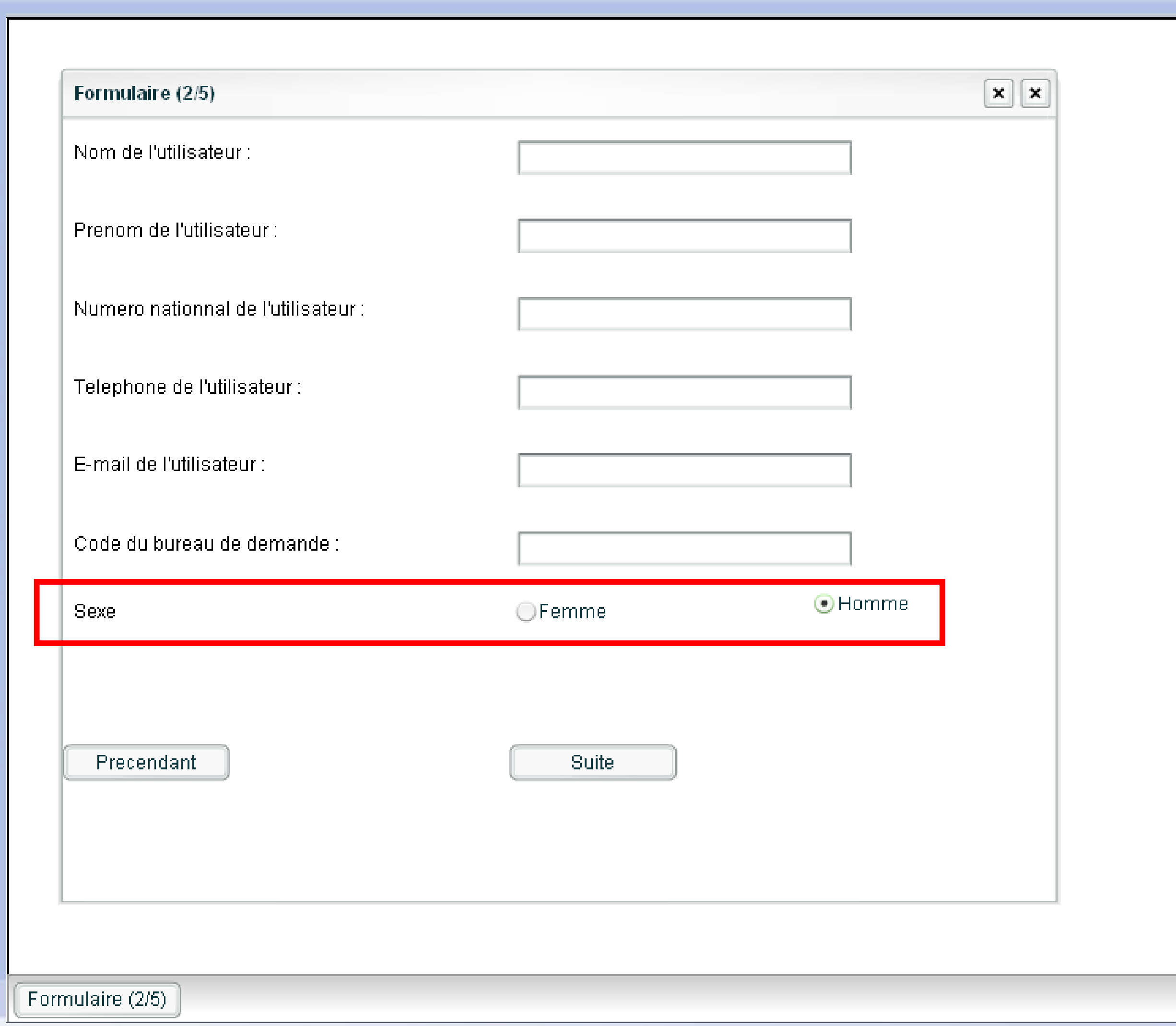


Example: Platform adaptation
widget substitution (1)

Excerpt for a UsiXML CUI specification

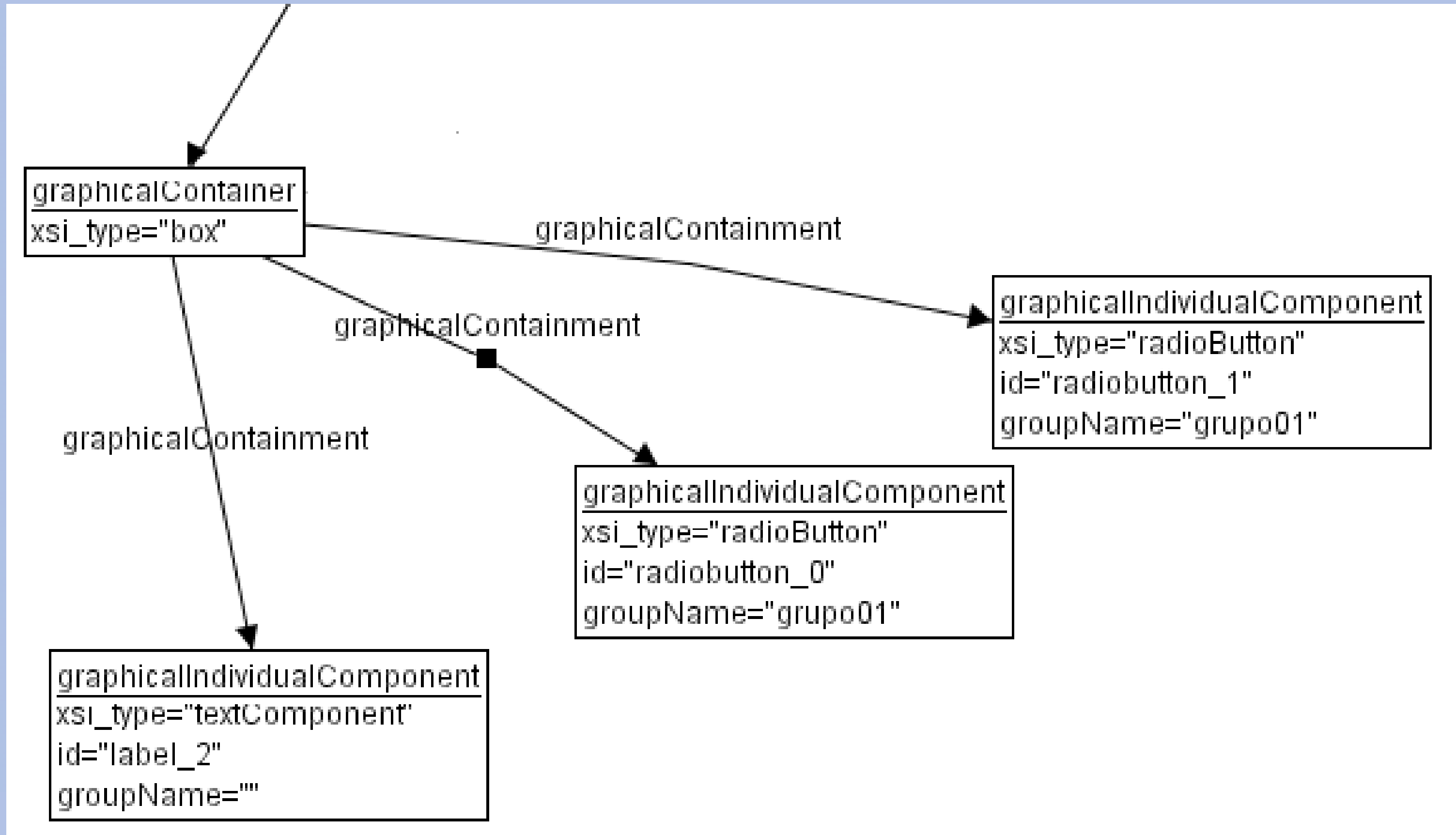
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cuiModel  name="MyModel">
  <version modifDate="2004-03-24T17:09:17.402+01:00" xmlns="">7</version>
  <authorName xmlns="">Youri</authorName>
  <window height="500" width="600" name="Formulaire (2/5)" id="window_1">
    <box relativeHeight="100" name="box1_0" id="box1_0">
      <box type="vert" name="boxTodo" id="boxTodo">
        ...
      <box type="horiz" name="box_2_2_2_1" id="box_2_2_2_1">
        <textComponent defaultContent="Sexe" isBold="true" id="label_2"/>
        <radioButton groupName=“grupo01" defaultContent="Femme"
                     defaultState="false" id="radiobutton_0"/>
        <radioButton groupName="grupo01" defaultContent="Homme"
                     defaultState="true" id="radiobutton_1"/>
      </box>
        ...
      </box>
    </box>
  </window>
</cuiModel>
```

Example: widget substitution (2)



Example: widget substitution (3)

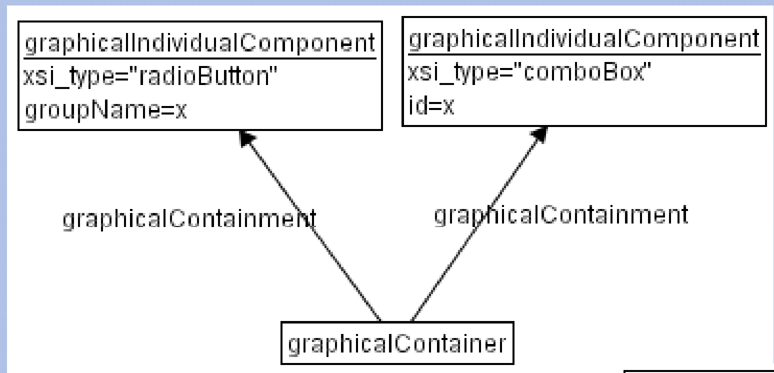
The UsiXML graph before applying any rule



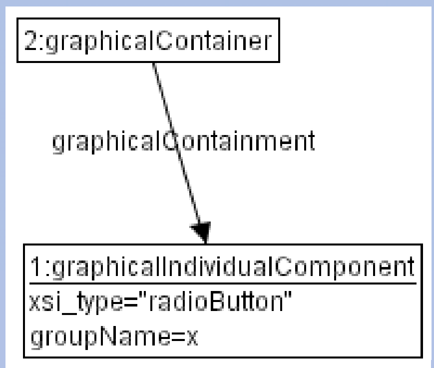
Example: widget substitution (4)

Rule 1: Create a new comboBox with the same *id* and *name* as the name of the group of radioButtons.

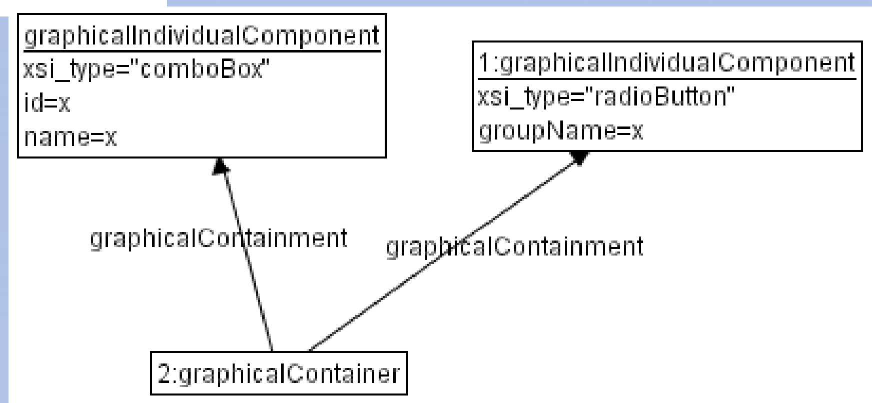
NAC



LHS



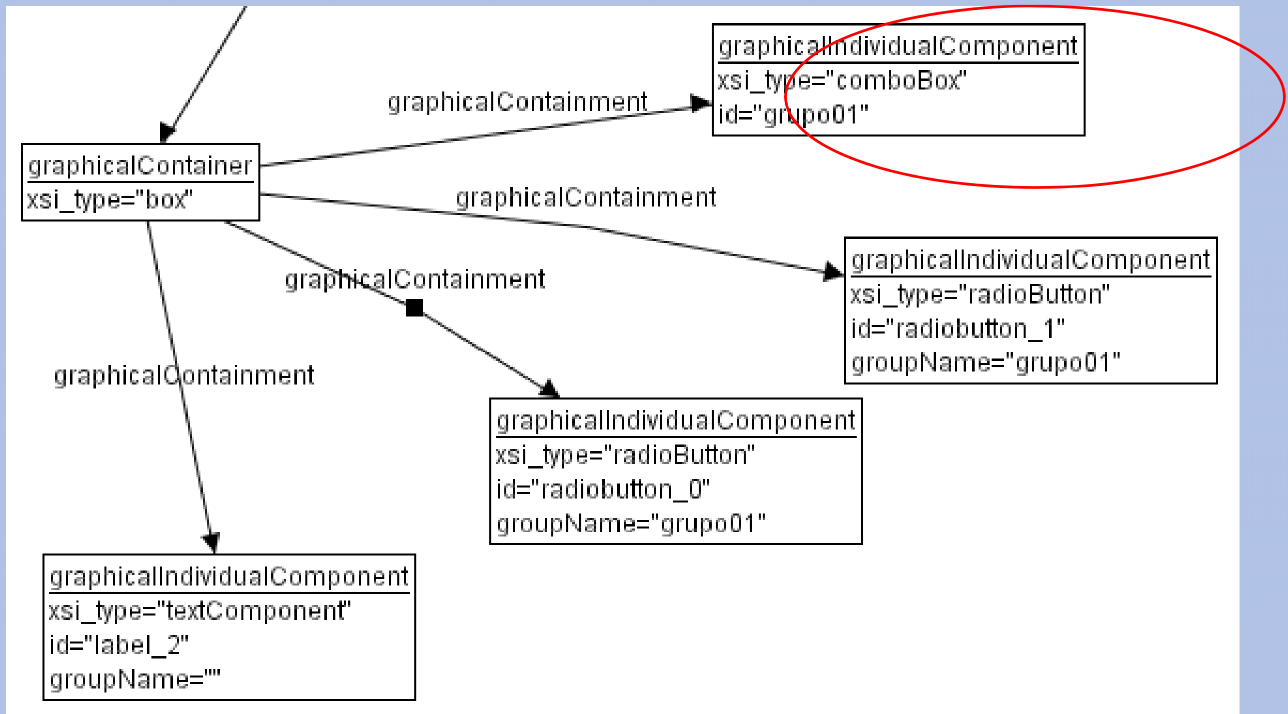
RHS



Example: widget substitution (5)

The UsiXML graph after applying the first rule

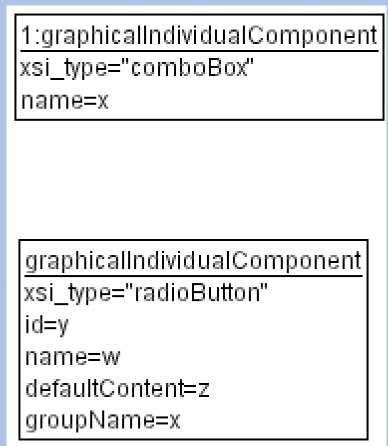
Rule 1: Create a new comboBox with the same *id* and *name* as the name of the group of radioButtons.



Example: widget substitution (6)

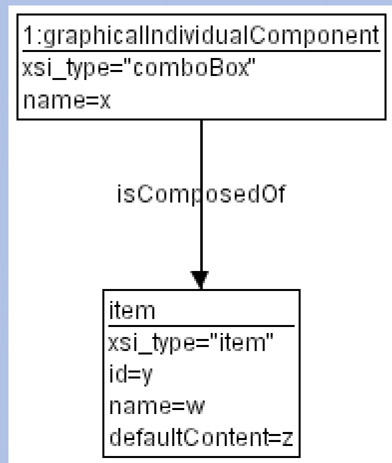
Rule 2: Convert every radio button within the group “x” into an item for the comboBox “x” that we have just created thanks to Rule 1

LHS



::=

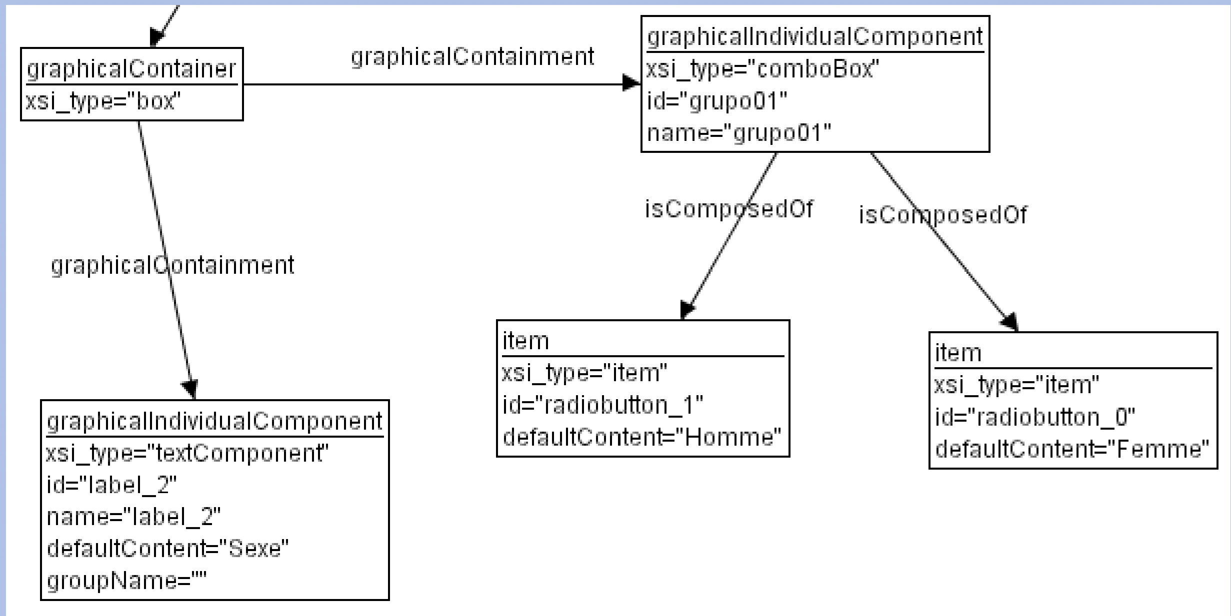
RHS



Example: widget substitution (7)

The UsiXML graph after applying the second rule: radio buttons disappeared

Rule 2: Convert every radioButton within the group “x” into an item for the comboBox “x”, we have just created.

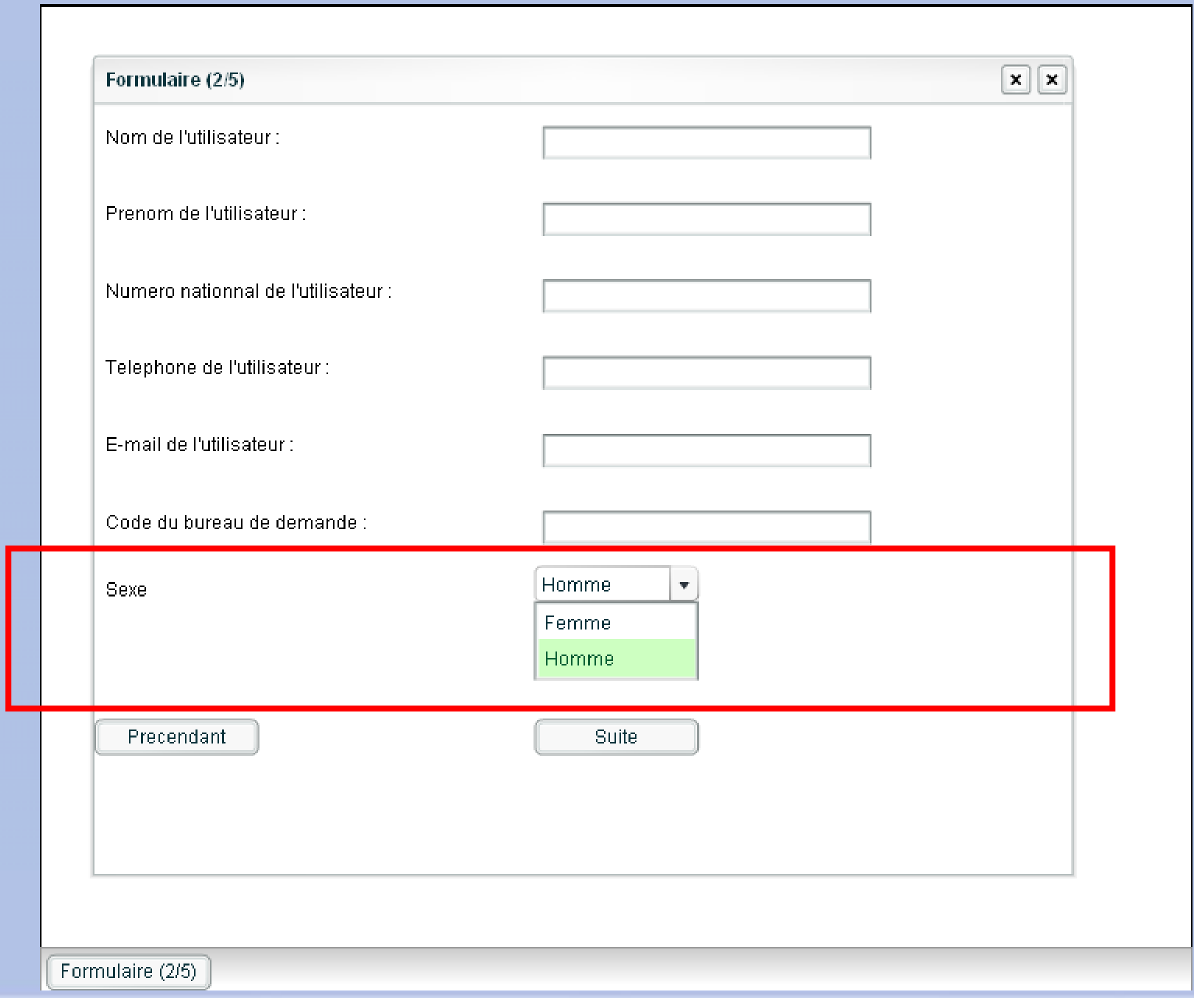


Example: widget substitution (8)

Excerpt from the final transformated UsiXML specification

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cuiModel  name="MyModel">
   <version modifDate="2004-03-24T17:09:17.402+01:00" xmlns="">7</version>
   <authorName xmlns="">Youri</authorName>
   <window height="500" width="600" name="Formulaire (2/5)" id="window_1">
      <box relativeHeight="100" name="box1_0" id="box1_0">
          <box type="vert" name="boxTodo" id="boxTodo">
              ...
          <box type="horiz" name="box_2_2_2_1" id="box_2_2_2_1">
              <textComponent defaultContent="Sexe" isBold="true" id="label_2"/>
              <comboBox id="comboBox001" name="label_3" isDropDown="true">
                 <item id="radiobutton_0" name="radiobutton_0" defaultContent="Femme"/>
                 <item id="radiobutton_1" name="radiobutton_1" defaultContent="Homme"/>
              </comboBox>
             ...
          </box>
      </box>
   </window>
</cuiModel>
```

Example: widget substitution (9)



Thank you very much for your attention

For more information and downloading,
http://www.usixml.org
User Interface eXtensible Markup Language

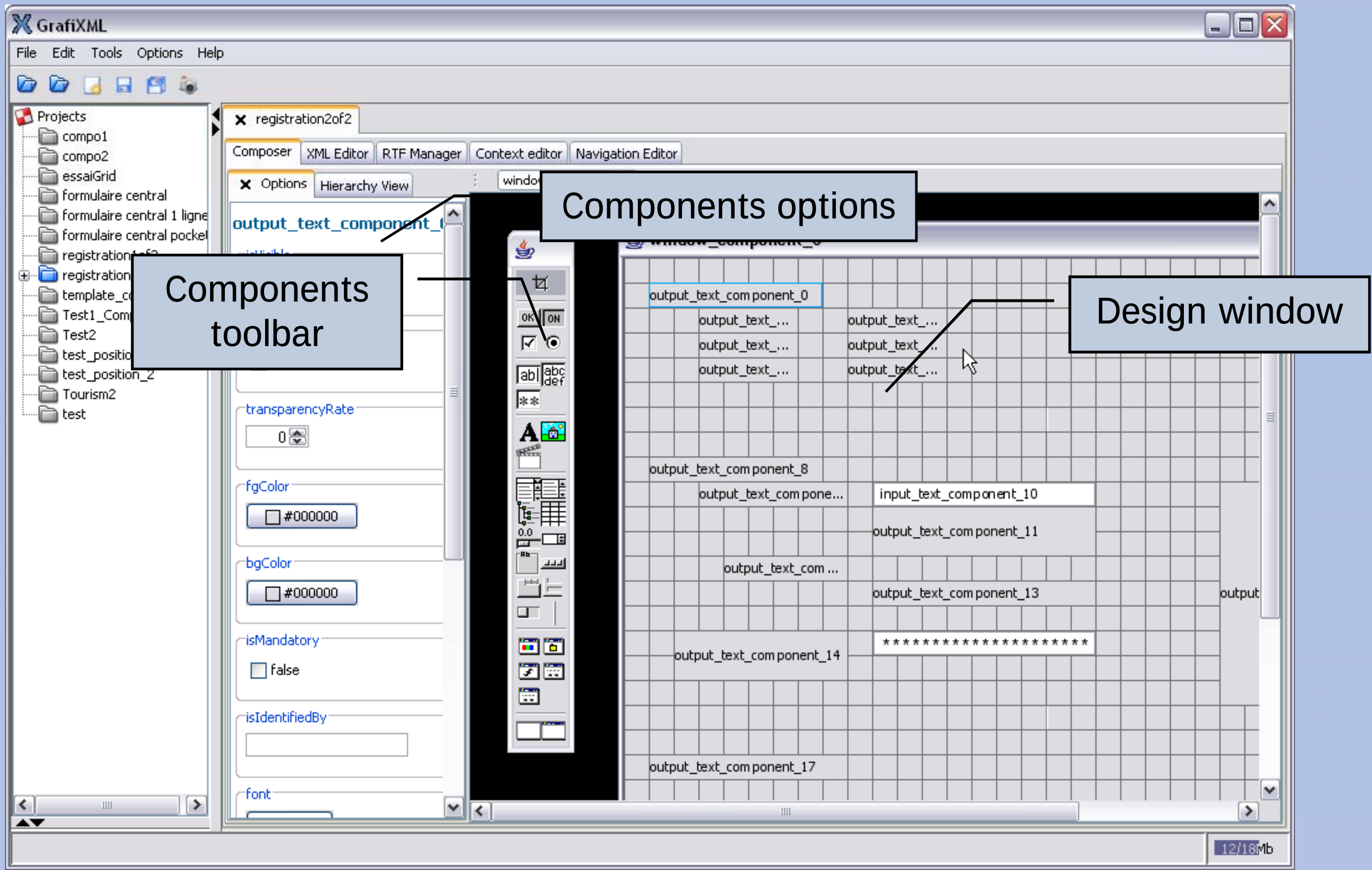
http://www.similar.cc
European network on Multimodal UIs

http://www.isys.ucl.ac.be/bchi

Special thanks to all members of the team!

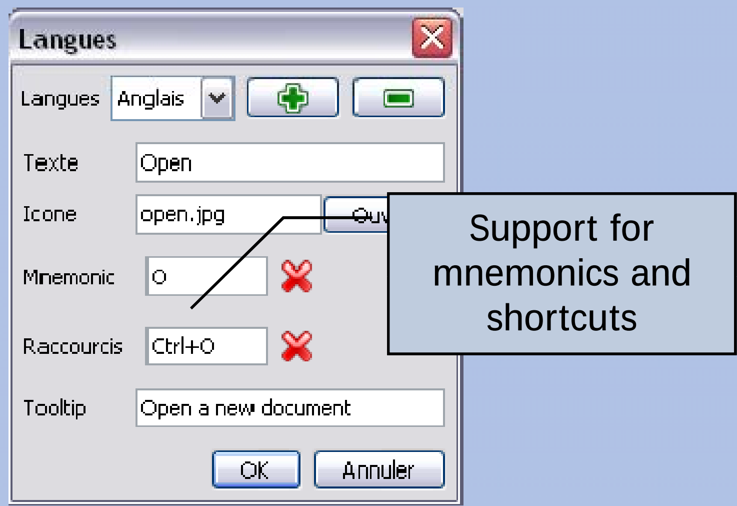
Tools

GrafiXML Design Tab



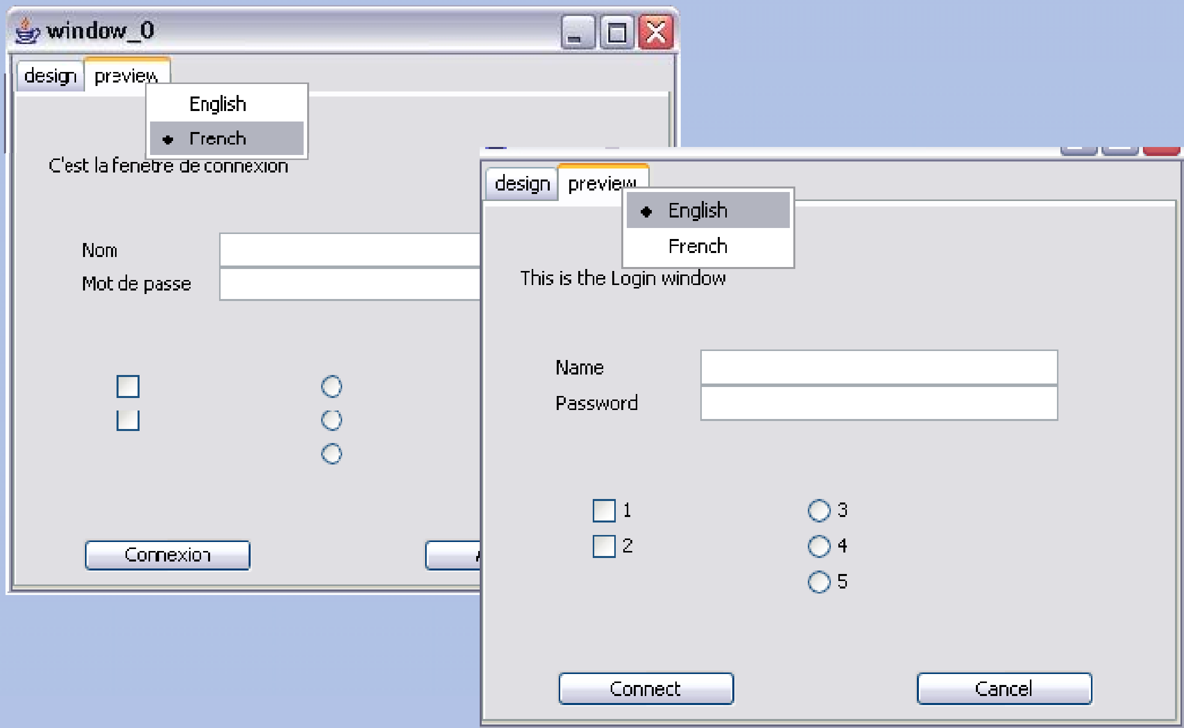
Language

GrafiXML allows the user to create multi-language GUI



Preview

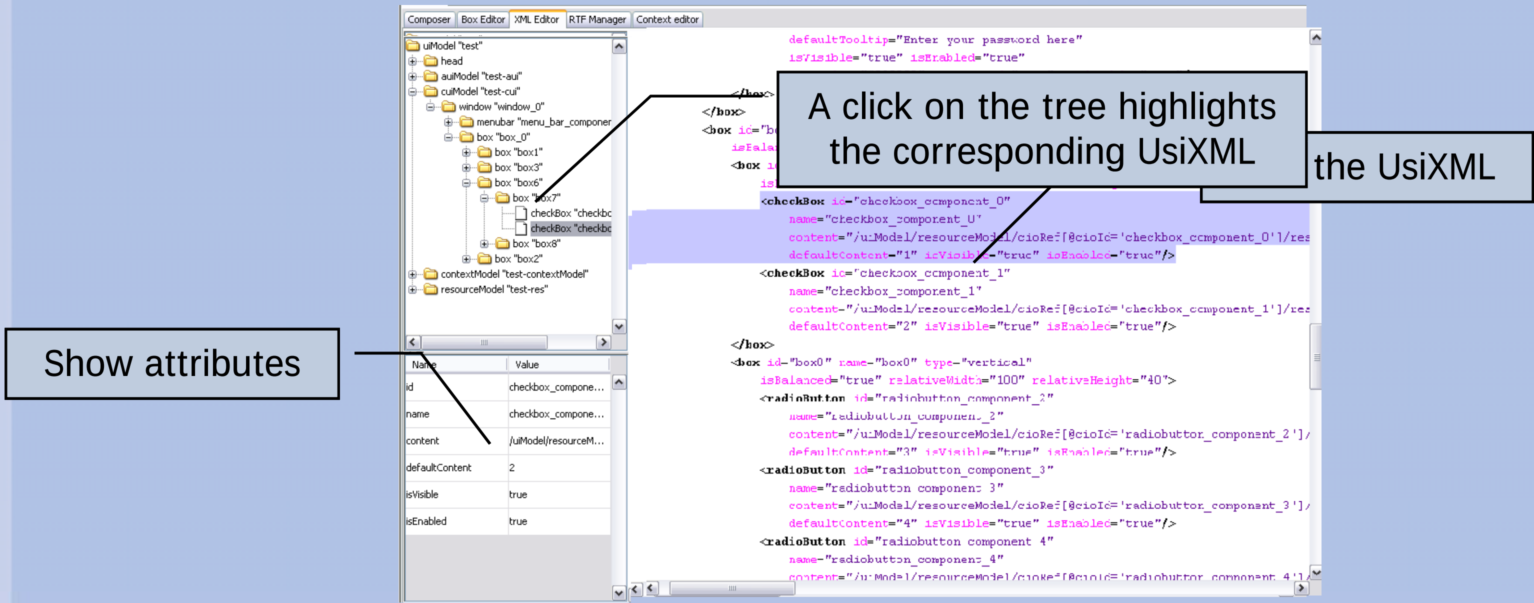
At any time, you can preview the ui in the language you want



XML Editor

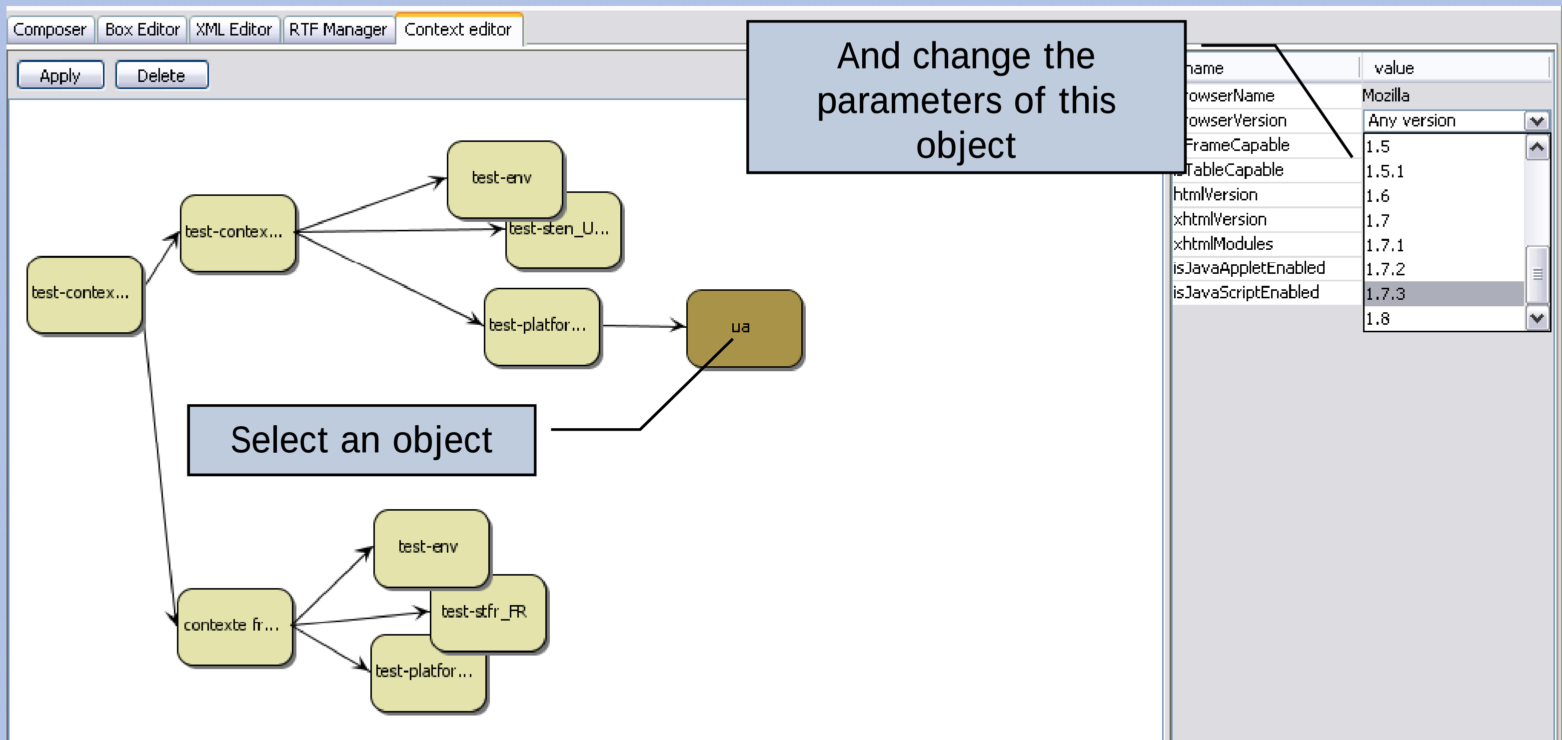
GrafiXML contains a XML editor which shows the UsiXML specification of your work

- You can edit yourself some part of the XML

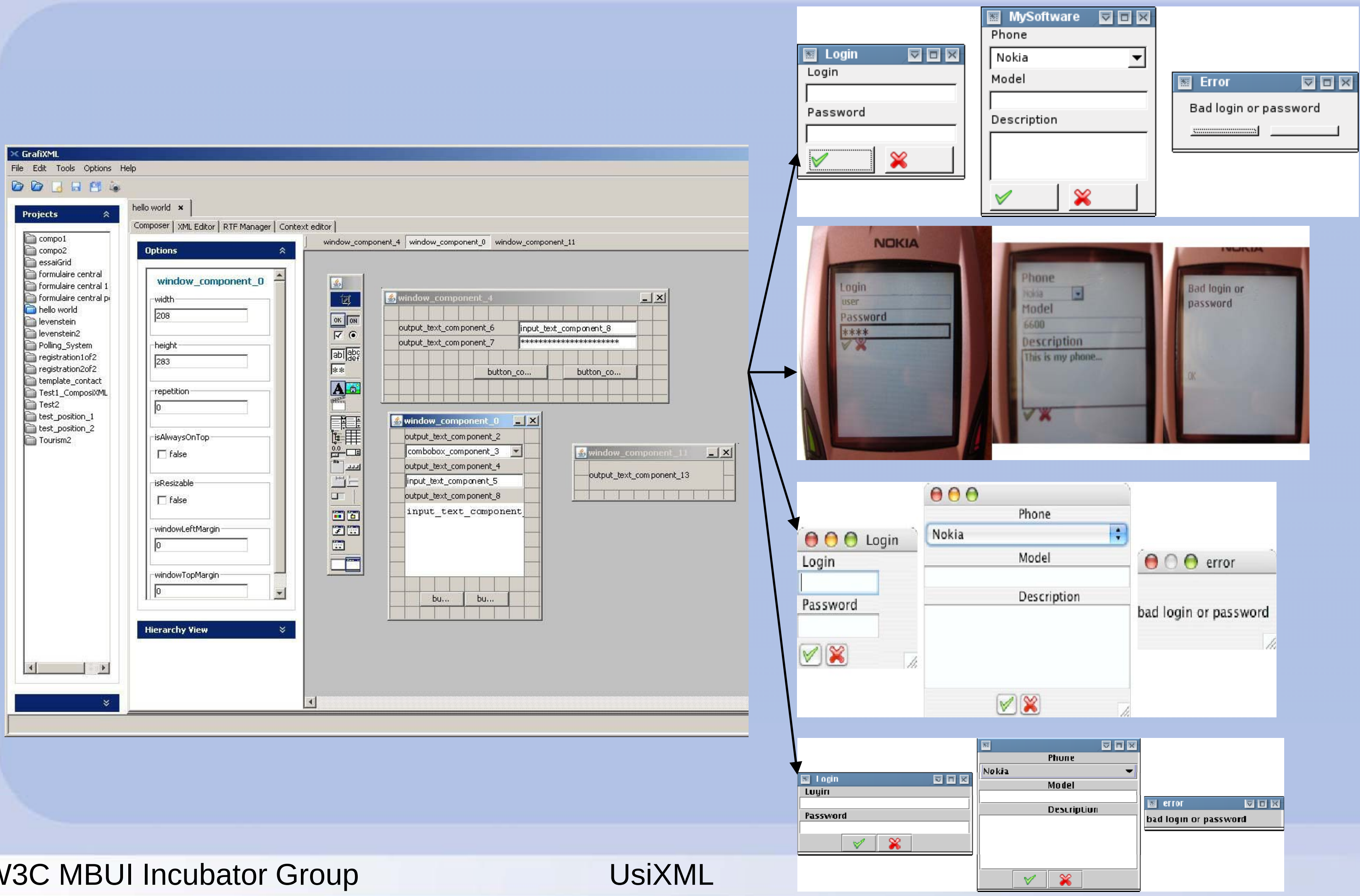


ContextModel Editor

You can create a contextModel using Drag&Drop



Example


GrafiXML
File Edit Tools Options Help
Projects
compo1
compo2
essaiGrid
formulaire central
hello world
levenstein
levenstein2
Polling_System
registration1of2
registration2of2
template_contact
Test1_ComposiXML
Test2
test_position_1
test_position_2
Tourism2
hello world
Composer
XML Editor
RTF Manager
Context editor
Options
window_component_0
width
208
height
283
repetition
isAlwaysOnTop
false
isResizable
windowLeftMargin
windowTopMargin
Hierarchy View
window_component_4
output_text_component_6
output_text_component_7
input_text_component_8
button_co...
output_text_component_2
combobox_component_3
output_text_component_4
input_text_component_5
output_text_component_8
input_text_component
window_component_11
output_text_component_13
Login
Password
MySoftware
Phone
Nokia
Model
Description
Error
Bad login or password
NOKIA
user
6600
This is my phone...
error
bad login or password