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
An example – Task Model
<?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

W3C MBUI Incubator Group
UsiXML
April 20, 2009
From task to AUI

Abs.container
Abs. component
Input
Output
Navigation
Control
Select
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)
<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>
</abstractContainer>

Name: 
Zip Code: 
Gender: M F 
Age : 18-25  25-45  45+ 
From CUI to FUI

GUI Version

Virtual Polling System

Participate to Opinion Poll

Provide Personal Data

- create name
- create zipCode
- select sex
- select age category

3D Version

Vocal

Name: [ ] Zip Code: [ ] Gender: [ M ] [ F ] Age: [ 18-25 ] [ 25-45 ] [ 45+ ]

Vocal+ Graphical

Name: Juan Manuel Gonzalez Zip Code: 81348

Gender

- Male
- Female

Age Category

- 18-35
- 35-45
- 45+
MDE based on UsiXML

MDA Components

- Computing Independent Model (CIM)
- Platform Independent Model (PIM)
- Platform Specific Model (PSM)

Model to Model

Source code

Techniques proposed based on UsiXML

- UsiXML models: task, domain
- Graph transformations
- UsiXML model: Abstract user interface
- Graph transformations
- UsiXML model: Concrete user interface
- Rendering
- Final user interface

W3C MBUI Incubator Group

UsiXML

April 20, 2009
The big picture of MDA supported in UsiXML

UsiXML model: task, domain

UsiXML model: Abstract user interface

UsiXML model: Concrete user interface

KnowiXML

Graph transformations

Graph transformations

GrafiXML
VisiXML
SketchiXML
FormiXML

FlashiXML
QtkXML
JaviXML

Rendering

Generative programming

Final user interface

Derivation rules

ReversiXML

IdealXML

TransformiXML

MethodiXML
• Three level of description
  – Semantics
  – Syntax
  – Stylistics
Excerpt for a UsiXML CUI 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"/>
          <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

- \( T_1 >> T_2 \) Enabling
- \( T_1[ ]>>T_2 \) Enabling + information passing
- \( T_1 > T_2 \) Suspend/resume
- \( T_1 [ ] T_2 \) Non-deterministic choice
- \( T_1 \pi T_2 \) Deterministic choice
- \( T_1 [ > T_2 \) Disabling (e.g. Form submit)
- \( T_1 | =| T_2 \) Independence (any order, but finished)
- \( T_1* \) Iteration
- \( T_1\{n\} \) Finite iteration
- \( T_1 ||| T_2 \) Concurrency
- \( T_1 [\times] T_2 \) Concurrency + information passing
- \([T]\) Optional
- \( T \) Recursion
## Operators selection

<table>
<thead>
<tr>
<th>AMBOSS</th>
<th>ANSI/CEA</th>
<th>Diane +</th>
<th>GOMS</th>
<th>GTA</th>
<th>HTA</th>
<th>TKS</th>
<th>TOOD</th>
<th>UsiXML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decomposition</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hierarchy</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Sequence</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ SEQ</td>
<td></td>
<td>✓ Ordered = true, information passing (Postcondition)</td>
<td>✓ Ordered sequence</td>
<td>✓ Sequence</td>
<td>✓ Seq</td>
<td>Fixed sequence</td>
<td>Sequence</td>
<td>Sequence</td>
</tr>
<tr>
<td>✓ DEC</td>
<td></td>
<td>✓ Order of alternatives</td>
<td>✓ Order of alternatives</td>
<td>✓ Order of alternatives</td>
<td>✓ Order of alternatives</td>
<td>✓ Order of alternatives</td>
<td>✓ Order of alternatives</td>
<td>✓ Order of alternatives</td>
</tr>
<tr>
<td>✓ SORT</td>
<td></td>
<td>✓ Ordered sequence</td>
<td>✓ Sequence</td>
<td>✓ Seq</td>
<td>Fixed sequence</td>
<td>Sequence</td>
<td>Sequence</td>
<td>Sequence</td>
</tr>
<tr>
<td>✓ ALT</td>
<td></td>
<td>✓ Required choice, free choice</td>
<td>✓ Or (If, then, else)</td>
<td>✓ Or (If, then, else)</td>
<td>✓ Or (If, then, else)</td>
<td>✓ Or (If, then, else)</td>
<td>✓ Or (If, then, else)</td>
<td>✓ Or (If, then, else)</td>
</tr>
<tr>
<td>✓ OR</td>
<td></td>
<td>✓ Optional (If, then, else)</td>
<td>✓ Optional (If, then, else)</td>
<td>✓ Optional (If, then, else)</td>
<td>✓ Optional (If, then, else)</td>
<td>✓ Optional (If, then, else)</td>
<td>✓ Optional (If, then, else)</td>
<td>✓ Optional (If, then, else)</td>
</tr>
<tr>
<td>✓ ZERO+</td>
<td></td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
</tr>
<tr>
<td>✓ ONE+</td>
<td></td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
</tr>
<tr>
<td>✓ MANY+</td>
<td></td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
<td>✓ Optional</td>
</tr>
</tbody>
</table>

Enabling, enabling with information passing

Iteration, finite iteration

Deterministic choice, undeterministic choice, inclusive choice

Optional
## Operators selection (Cont…)

<table>
<thead>
<tr>
<th></th>
<th>AMBOSS</th>
<th>ANSI/CEA</th>
<th>Diane +</th>
<th>GOMS</th>
<th>GTA</th>
<th>HTA</th>
<th>TKS</th>
<th>TOOD</th>
<th>UsiXML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interruption</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>±</td>
<td>±</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>√ Interruption</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>±</td>
<td>±</td>
<td></td>
<td></td>
<td></td>
<td>Suspend-resume, disabling, enabling with information passing</td>
</tr>
<tr>
<td><strong>Concurrency</strong></td>
<td>✓</td>
<td>±</td>
<td>✓</td>
<td>±</td>
<td>✓</td>
<td>±</td>
<td>X</td>
<td>✓</td>
<td>✓ Concurrency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ordered =</td>
<td>unordered</td>
<td>±</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Independent concurrency, concurrency with information passing, order independence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
<td>sequence</td>
<td>±</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>±</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cooperation</strong></td>
<td>±</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓ Cooperation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precondition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Collaboration(FKS extension)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Collaboration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cooperation</td>
</tr>
<tr>
<td><strong>Parallel</strong></td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓ ParallelSplit (process model)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAR, SIM</td>
<td>Parallel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

W3C MBUI Incubator Group  | UsiXML  | April 20, 2009
New Abstraction: the task metamodel
### Task Identification criteria

<table>
<thead>
<tr>
<th>Time</th>
<th>Space (location)</th>
<th>Resource</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workflow</strong></td>
<td>Series of time periods</td>
<td>Different locations; same organization</td>
<td>Same or different groups of resources</td>
</tr>
<tr>
<td><strong>Process</strong></td>
<td>Series of time periods</td>
<td>Different locations</td>
<td>Within groups, group as a whole, or among groups</td>
</tr>
<tr>
<td><strong>Task</strong></td>
<td>Same time period</td>
<td>Same location</td>
<td>One or two types of resources</td>
</tr>
</tbody>
</table>
Resource task allocation patterns

```
<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" Stateless_reallocation="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" Stateless_reallocation="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

[Tactile listbox] [Sound feedback] [Synthetic speech]

- ([description of tactile listbox])
- ([description of sound feedback])
- ([description of synthetic speech])

[Vanderdonckt & Bodart, 1993]
Abstraction: the abstract UI

- Notation: based on L. Constantine’s notation for canonical abstract prototypes
  [Constantine, 2003]
Example of AUI produced
Mapping the models

These mappings can be established:

- triggers (tg): \{\,\} x
- updates (up):
- 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

Transformation System

G \rightarrow G' 

Host USIXM specification  Resultant USIXM specification

Not Matches  Matches  Co-Matches

NAC + LHS \rightarrow RHS

Transformation Rule 1

Transformation Rule 2

... 

Transformation Rule N

PIM step: task+domain to AUI

- Abstract UI (AUI) = UI independent of any interaction modality
- Definition of AUI structure in terms of Abstract 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?
STEP : From Task & Domain to AUI

- Identification of AUI structure
- Selection of AIC
- Spatio-Temporal Arrangement of AIOs
- Definition of Abstract Dialog Control
- 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

UsiXML models: task, domain
Graph transformations
UsiXML model: Abstract user interface
Graph transformations
UsiXML model: Concrete user interface
PSM Step: AUI to CUI

STEP: From AUI to CUI

SUB-STEPS:
- Reification of AC into CC
- Selection of CIC
- Arrangement of CICs
- Definition of Navigation
- Concrete Dialog Control Definition
- Derivation of CUI to Domain Relationships
PSM sub-step 3: definition of navigation
An example of a complex rule

NAC

LHS

RHS

::=

::=

::=

::=

W3C MBUI Incubator Group                              UsiXML                                              April 20, 2009
PSM: Concrete User Interface

Arrange Flight

Determine Prefs
Input Card Details
Select Card Type
Input Card Holder
Input Card Number
Expiration Date

Select Flight
Proceed Payment

Search Flight
Dt
i
P
f

Determine Origin
Determine Destination
Determine Via
Determine Time
Determine Budget
Launch Search

Determine Prefs
Airport Name
City
Country

Feedback of Check Card

OK

Back to Search Flight
Back to Arrange Flight
Back to determine prefs
Back to Proceed Payment
Example: Platform adaptation
widget substitution (1)

Excerpt for a UsiXML CUI 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">
                ...
                <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>
    </window>
</cuiModel>
```
Example: widget substitution (2)
Example: widget substitution (3)

The UsiXML graph before applying any rule
Rule 1: Create a new comboBox with the same id and name as the name of the group of radioButtons.
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.
Rule 2: Convert every radio button within the group “x” into an item for the combo Box “x” that we have just created thanks to Rule 1
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 transformed 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>
    </box>
  </window>
</cuiModel>
```
Example: widget substitution (9)
Thank you very much for your attention.

For more information and downloading, go to [http://www.usixml.org](http://www.usixml.org).

User Interface eXtensible Markup Language (UI eXtensible Markup Language, XUL) is a user interface markup language that is used in web browsers, mainly in Mozilla Firefox. It is designed to provide a rich user interface for web applications.

[http://www.similar.cc](http://www.similar.cc) is the European network on Multimodal User Interfaces (UIs). It is a network of excellence that focuses on the development of multimodal user interfaces.

[http://www.isys.ucl.ac.be/bchi](http://www.isys.ucl.ac.be/bchi) is a special thanks to all members of the team!
Tools
GrafiXML Design Tab

Components toolbar

Components options

Design window
GrafiXML allows the user to create multi-language GUI

Support for mnemonics and shortcuts
At any time, you can preview the UI in the language you want.
GrafiXML contains a XML editor which shows the UsiXML specification of your work

- You can edit yourself some part of the XML

A click on the tree highlights the corresponding UsiXML

Show attributes
You can create a contextModel using Drag&Drop

Select an object

And change the parameters of this object