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
Who are we?
Belgian Lab of Human-Computer Interaction (BCHI) The BCHI Lab has **20 years of experience** in the domain of user interface engineering, which combines techniques from Human-Computer Interaction, Software Engineering, and Usability Engineering.

Model based User Interface Development
- Multi Modal
- 3D UIS
- 2D UIS (Web, desktop, ...)
- Migration
- Context adaptation
- ....
BCHI-Past Projects

- **Cameleon** (Context Aware Modelling for Enabling and Leveraging Effective interaction)
- **Envir3D** (Automatic Generation of Virtual Reality Scenes)
- **Kwaresmi** (Knowledge-based Web Automatic Reconfigurable evaluation with guidelines optimization)
- **MetroWeb** (METROlogy of WEB sites)
- **Salamandre** (User Interfaces for Mobile and Multi-platform Interactive Systems)
- **Visme** (VISual Scene composition with multi-resolution and modulation for a Multi-sources Environment dedicated to neuro-navigation)
- **Similar** (The European taskforce creating human-computer interfaces SIMILAR to human-human communication)
- **Destine** (Design and Evaluation STudio for INtent-based Ergonomic web sites)

W3C Model Based UI '2008 – Mandelieu, October 24, 2008
BCHI-Current Projects

- **UsiXML** (USer Interface eXtensible Markup Language)
- **Vitality** User Interface for Medical data visualization
- **HUMAN** Model-Based Analysis of Human Error During Aircraft Cockpit System Design
What are we doing?
UsiXML-the Problem

- To develop user interfaces (UIs) simultaneously for multiple contexts of use
- A context of use = triple
  - User
  - Computing platform
  - Surrounding environment
    - Organisation
    - Socio-psychological factors
• What is UsiXML?
  – It is a XML-compliant User Interface Description Language
  – Publicly available from http://www.usixml.org
  – Free to use, open for access, easy to expand
  – Definition of the language

UML Class Diagrams ➔ UsiXML Reference manual

XSD XML Schema Descriptions ➔ UsiXML Models
• **UsiXML = USeR Interface exTensible Markup Language**
  - [http://www.usixml.org](http://www.usixml.org)
  - Join the UsiXML Consortium by registering on line
What do we have so far?
Any development method (or methodology) is decomposed into 4 axes:

- **Models**: explicitly capture knowledge about UI and Interactive Applications with appropriate abstractions
- **Language**: In order to specify different aspects and related models, a specification language is needed that allows designers and developers to exchange, communicate, and share fragments of specifications and that enables tools to operate on these specifications.
- **Method**: structures the definition and use of underlying models in a stage-wise approach
- **Supporting tools**: support the use of the method by providing tools for models and their related operations. Ideally, one model should be supported by at least one tool
• Goal: to integrate all three facets
Models
The collection of models for specifying a user interface
The language
Language Engineering approach

- UsiXML is different from a pure UI authoring language in that it could also be used as a specification language.
- The ultimate goal is not only to generate code, but also to have the capability to reason 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
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

- Low
- Medium
- High
Abstraction: the abstract UI

- Notation: based on L. Constantine’s notation for canonical abstract prototypes

[Constantine, 2003]

[Montero et al., 2005]
The method
MDE based on UsiXML

MDA Components

- Computing Independent Model (CIM)
  - Model to Model
- Platform Independent Model (PIM)
  - Model to Model
- Platform Specific Model (PSM)
  - Model to Code
  - 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 Model Based UI '2008 –Mandelieu, October 24, 2008
MDE based on UsiXML

UI Development methodology
- Software tools
- Step-wise method
- Models
- UsiXML Language

Industrial applications
Public applications

is applied on

S=Source context of use
T=Target context of use

User S Platform S Environment S
Task and Domain S
Abstract User Interface S
Concrete User Interface S
Final User Interface S

Cameleon Reference Framework [Calv03]

Computation Independent Model (CIM)
Platform Independent Model (PIM)
Platform Specific Model (PSM)

Reification
Abstraction
Reflexion
Translation

Industrial applications
Public applications

Standardisation actions: W3C, OASIS, MAUSE

Industrial applications
Public applications

is applied on

W3C Model Based UI '2008 –Mandelieu, October 24, 2008
Typed Model-to-Model Transformation

- Meta-Meta-Model
  - Graph Structure
  - is instance of
  - Meta-Model
    - Our Meta-Model
      - Meta-Model Subset 1
        - e.g., Task+Domain Model
          - is instance of
      - Meta-Model Subset 2
        - e.g., Concrete UI Model
          - is instance of
      - Initial UI Model
        - e.g., MyTaskAndDomainModel
      - Transformation Rule
        - Our transformation catalog
      - Resultant UI Model
        - e.g., MyConcreteUIModel
          - Uses language

[Limbourg, 2004]
• 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
Example of the method
In practice

S=Source context of use

T=Target context of use

User S | Platform S | Environment S

User T | Platform T | Environment T

Task and Domain S

Task and Domain T

Abstract user Interface S

Abstract user Interface T

Concrete user Interface S

Concrete user Interface T

Final user Interface S

Final user Interface T

UsiXML supported model

UsiXML unsupported model

Reification

Abstraction

Reflexion

Translation

W3C Model Based UI '2008 –Mandelieu, October 24, 2008
In practice
In practice
In practice
One description - Many representations

Task & Concepts
- Task
- Classes

Abstract User Interface (AUI)
- Modality-independent AIO type
- Modality-dependent AIO

Concrete User Interface (CUI)
- Platform-independent CIO type
- Platform-independent CIO

Final User Interface (FUI)
- Code
- Rendering

Method triggered: download file
Object: computer file

Control AIO
Software control AIO
Physical control AIO

Graphical 2D push button
Graphical 3D push button

HTML pushbutton
Windows push button
OSF/Motif XmButton
VRML97/X3D button
Software key
Function key

W3C Model Based UI ’2008 – Mandelieu, October 24, 2008
Tools
The big picture of MDA

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

IdealXML

TransformiXML

MethodiXML

UsiXML

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

KnowiXML

- GrafiXML
- VisiXML
- SketchiXML
- FormiXML

ReversiXML

FlashiXML
- QtkXML
- JaviXML

Generative programming
- Rendering

VisualiXML

Derivation rules
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
You can create a contextModel using Drag&Drop

Select an object

And change the parameters of this object
Example
Thank you very much for your attention.

For more information and downloading, see:
- 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!
More Details
MDE based on UsiXML

MDA Components

- Computing Independent Model (CIM)
- Model to Model
- Platform Independent Model (PIM)
- Model to Model
- Platform Specific Model (PSM)
- Model to Code
- 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 Model Based UI '2008 –Mandelieu, October 24, 2008
CIM Step 1: Task model

W3C Model Based UI '2008 –Mandelieu, October 24, 2008
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: LOTOS operators (in CTTE)**
  - $T_1 >> T_2$ Enabling
  - $T_1[~] >> T_2$ Enabling + information passing
  - $T_1 \triangleright T_2$ Suspend/resume
  - $T_1 [~] T_2$ Non-deterministic choice
  - $T_1 \pi T_2$ Deterministic choice
  - $T_1 \triangleright T_2$ Disabling (e.g. Form submit)
  - $T_1 \triangledown T_2$ Independence (any order, but finished)
  - $T_1^*$ Iteration
  - $T_1\{n\}$ Finite iteration
  - $T_1 ||| T_2$ Concurrency
  - $T_1 [\mathbf{x}]| T_2$ Concurrency + information passing
  - $[T]$ Optional
  - $T$ Recursion

[Markopoulos, 1992]
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 task meta-model

[Limbourg, 2004]
CIM Step 3: Task-domain mappings

[Image of diagrams and XML code]

W3C Model Based UI '2008 –Mandelieu, October 24, 2008

[Limbourg,2004]
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

\[\text{Vanderdonckt & Bodart, 1993}\]
Abstraction: the abstract UI
Abstraction: the abstract UI

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

W3C Model Based UI '2008 – Mandelieu, October 24, 2008

[Montero et al., 2005]
Example of AUI produced
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 (!!)
Uses language

Meta-Meta-Model
Graph Structure

is instance of

Meta-Model
Our Meta-Model

is instance of

Meta-Model Subset 1
e.g., Task+Domain Model

is instance of

Initial UI Model
e.g., MyTaskAndDomainModel

Uses language

Transformation Rule
Our transformation catalog

is instance of

Meta-Model Subset 2
e.g., Concrete UI Model

is instance of

Resultant UI Model
e.g., MyConcreteUIModel

[Limbourg, 2004]
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
• Find an occurrence of LHS in G (this occurrence is called a match). If several occurrences exist, choose one non-deterministically.
• Check preconditions of both type PAC and NAC. If not verified, then skip.
• Remove the part of G which corresponds to (LHS – K), where K is the morphism specified between LHS and RHS.
• Add RHS – K into G – (LHS – K) as it is given by the corresponding relation between RHS – K and K.
• Check postconditions of both type PAC (and notably that the resulting graph is properly typed) and NAC. If not verified, then undo the transformation rule.
Transformation system

Transformation System

G Host USIXM specification

G' Resultant USIXM specification

LHS

RHS

Matches

Co-Matches

Transformation Rule 1

Transformation Rule 2

... Transformation Rule N

Transformation System

NAC + LHS

Is Transformed Into

RHS

Is Transformed Into

[Limbourg, 2004]
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?
PIM step: task+domain to AUI

STEP : From Task & Domain to AUI

SUB-STEPS

- 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

W3C Model Based UI '2008 –Mandelieu, October 24, 2008
PSM sub-step 3: definition of navigation
An example of a complex rule

W3C Model Based UI ’2008 –Mandelieu, October 24, 2008
PSM: Concrete User Interface

W3C Model Based UI '2008 –Mandelieu, October 24, 2008
How to read a graph transformation?

Node type (Attribute, value)

Node

Edge type

Edge

(Node, Edge)

Player

Id="12"
name="Jonvonderdong"
salary=2500

isPartOf
entryDate="05/04/02"
recruiter="Emile"

Team

Id="43"
name="The Broken Arms"
city="Louvain-la-Neuve"
What do we have so far?

S = Source context of use

- User S
- Platform S
- Environment S

Task and Domain S

Abstract user Interface S

Concrete user Interface S

Final user Interface S

UsiXML supported model

UsiXML unsupported model

Reification

W3C Model Based UI ’2008 –Mandelieu, October 24, 2008
Multiple development paths

1. Task and Domain
   - T1: Abstract User Interface
   - T2: Concrete User Interface
   - T3: Rendering
   - Final User Interface

2. Task and Domain
   - T1: Abstract User Interface
   - T2: Concrete User Interface
   - T3: Rendering
   - Final User Interface

3. Task and Domain
   - T1: Abstract User Interface
   - T2: Concrete User Interface
   - T3: Rendering
   - Final User Interface

4. Task and Domain
   - T1: Abstract User Interface
   - T2: Concrete User Interface
   - T3: Rendering
   - Final User Interface

5. Task and Domain
   - T1: Abstract User Interface
   - T2: Concrete User Interface
   - T3: Rendering
   - Final User Interface

6. Task and Domain
   - T1: Abstract User Interface
   - T2: Concrete User Interface
   - T3: Rendering
   - Final User Interface

7. Task and Domain
   - T1: Abstract User Interface
   - T2: Concrete User Interface
   - T3: Rendering
   - Final User Interface
A development library stores (in usiXML textual format) paths, steps and sub-steps definition and their associated transformation systems and transformation rules.
Multiple development paths

- Transformation System 1
  - Rule 1
  - Rule 2
  - ...
  - Rule n

- Transformation System 2
  - Rule 1
  - Rule 2
  - ...
  - Rule n

- Transformation System ...
  - Rule 1
  - Rule 2
  - ...
  - Rule n

- Transformation System n
  - Rule 1
  - Rule 2
  - ...
  - Rule n

- Development Step α

- → : when source terminates apply target
- ● → : execute development step

W3C Model Based UI '2008 –Mandelieu, October 24, 2008
• **API = set of transformations**

```
<window>
  <button>
    ....
  </window>
```

**USIXML specification (initial)**

```
::=
```

**Transformation rules expressed in USIXML**

```
::=
```

**USIXML specification (resultant)**

```
<window>
  <button>
    ....
  </window>
```

**Transformation API**

**rules applied**
From T&D to AUI

- TransformiXML

[Bouillon et al., 2005]
Two forms of UI rendering

- Interpretation
  - By run-time static analysis and direct rendering (InterpiXML & FormiXML)
- Code generation
  - By program synthesis (GrafiXML)
  - By generative programming (Angie)
    - Feature model
    - Components assembling