

USIXML: a User Interface Description for Specifying Multimodal User Interfaces

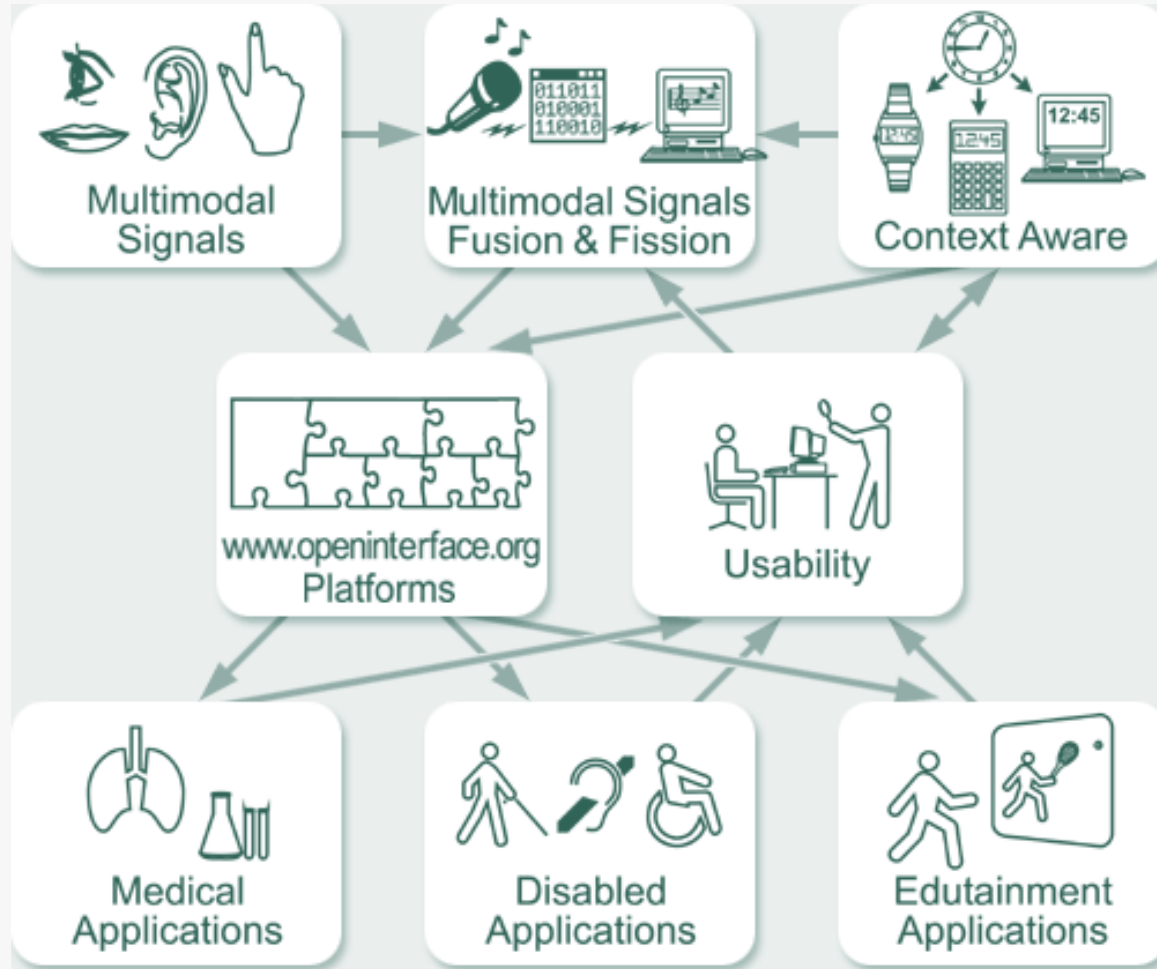
Jean Vanderdonckt, Quentin Limbourg, Benjamin Michote, Laurent Bouillon, Daniela Trevisan, Murielle Florins



*¹Belgian Laboratory of HCI
University Catholic of Louvain
Louvain-la-Neuve, Belgium*



www.similar.cc



UIs ARE RUNNING FAST ... AFTER CHANGE

- Task redefinitions
- Tasks reallocation
- Organizational adaptation
- Domain evolution
- Obsolescence of languages
- New languages
- New platforms
- ...

DEVELOPMENT PATHS

To face these challenges several development paths may be identified:

- Forward engineering
- Reverse engineering
- Adaptation to context of use
- Middle-out approach
- Widespread approach

MULTI-PATH DEVELOPMENT

To support these approaches in a single framework we need:

- An ontology of concepts valid for all paths.
- A central storage of models.
- A mean to express model transformations.
- An execution mechanism for performing transformations.

ONTOLOGY

- Task (CTT + minor improvements).
- Domain (Class + Object diagram + improvements)
- Abstract User Interface (vocabulary independent of the modality)
- Concrete user interface (vocabulary independent of the platform)
- Context of use (subset of CC/PP standard)
- Inter-model relationship mappings (traceability, integration of all views)

SYNTAX

Abstract syntax

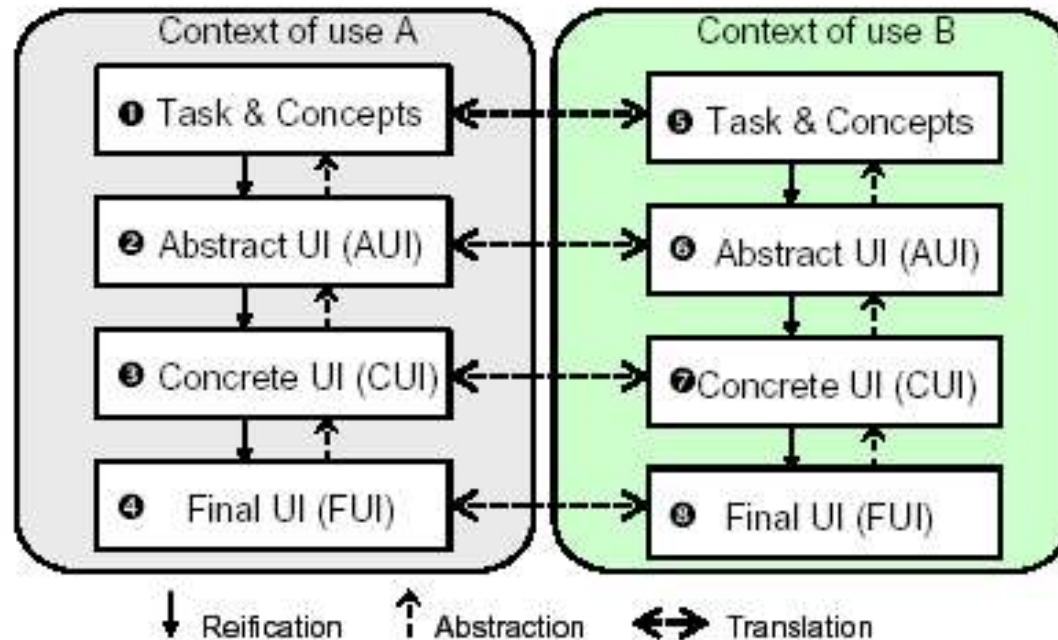
- Directed, labelled, attributed, typed graphs.
- Nodes are concepts.
- Edges are relationships between these concepts.
- Result: a UI specification is a BIG WHOLE graph.

- Concrete syntax : USIXML

- User Interface eXtensible Mark-Up Language
- (graph structure is a ed by defining explicitly relationships)



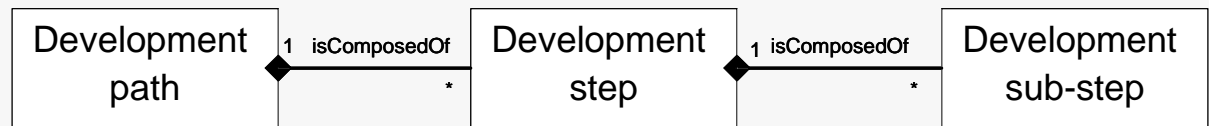
REFERENCE FRAMEWORK



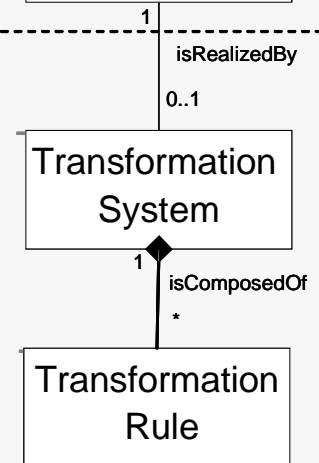
Multi-Directional UI development

DEVELOPMENT PATH CONNECTION TO TRANSFORMATIONS

Methodological World



Graph Transformation World



A development library and transformation models are available to store and reuse the defined development paths and transformations.

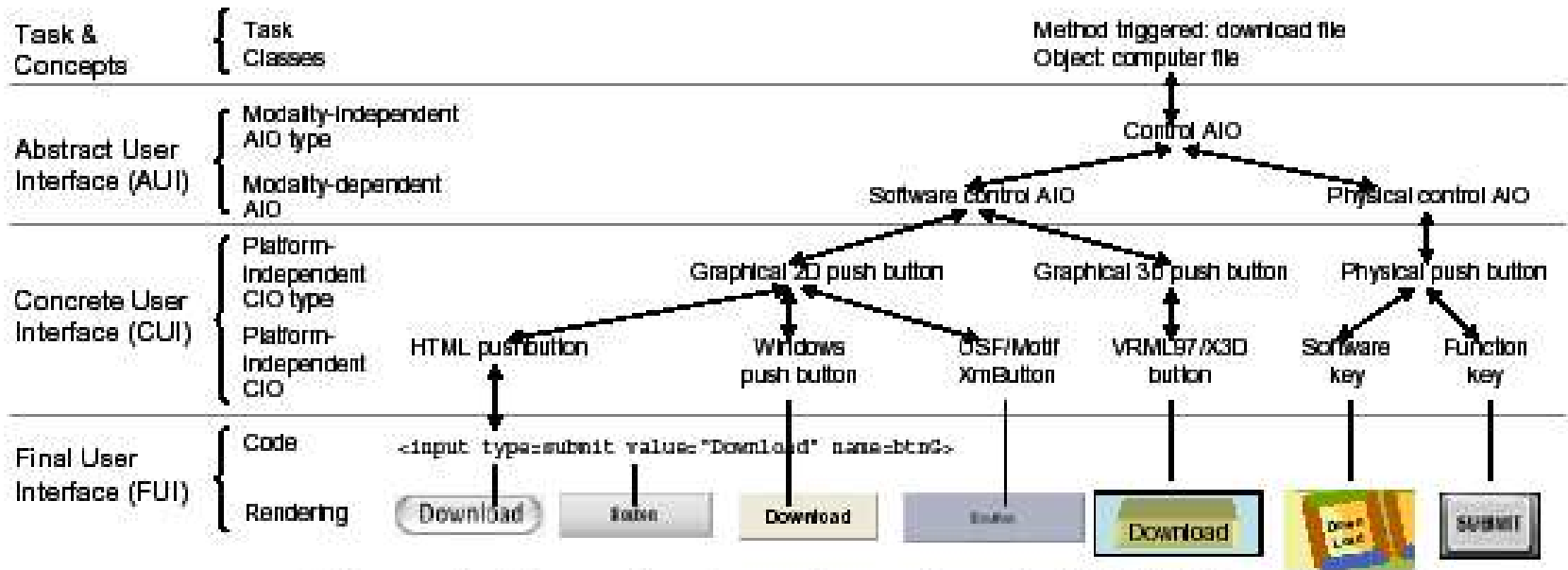
Graph Transformation

AGG – Attributed Graph Grammars

Generalization of string grammars.

- Grounded execution semantics (pushout construction).
- Side-effect free.
- Attractive syntax.
- Declarativeness.
- Seamlessness with ontological world (rules manipulate patterns of specification).
- The rules are applied in a pure sequential programmed graph rewriting manner.

Example of transformations

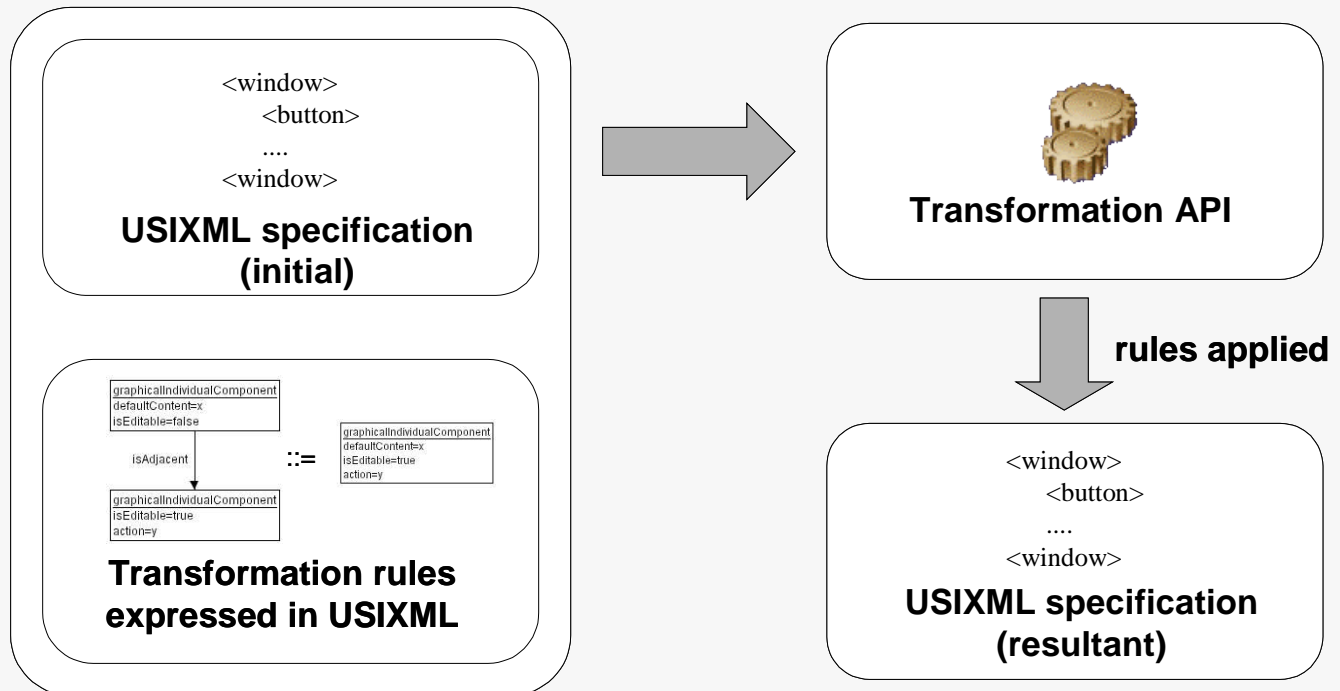


TOOL SUPPORT

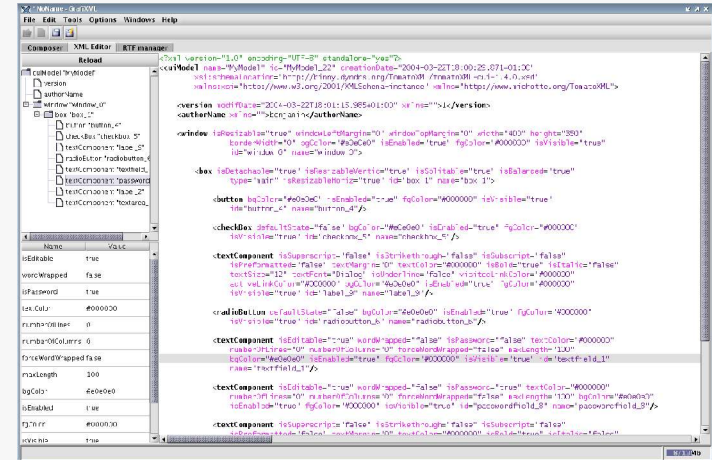
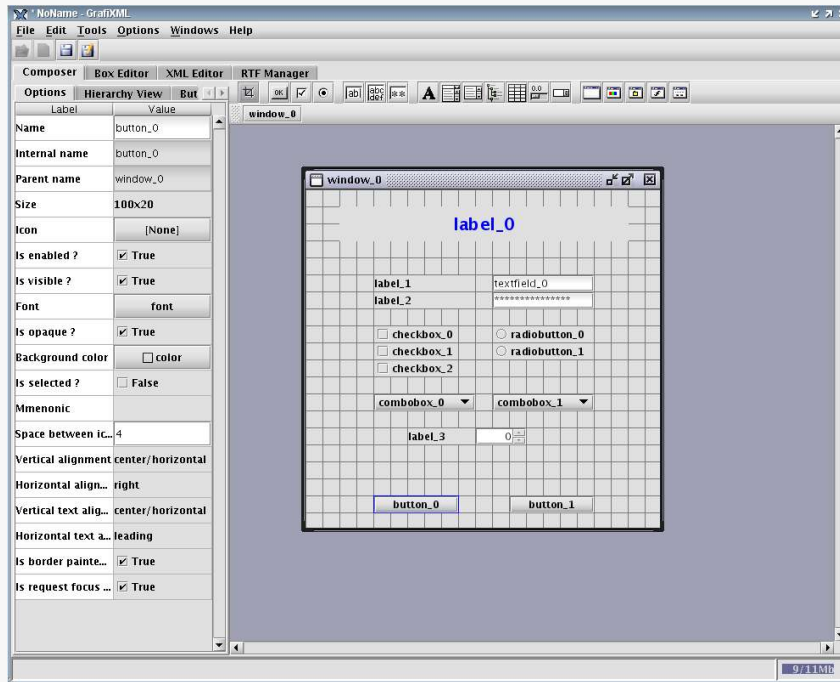
Running prototypes

- TransformiXML API : transformation tool
- GrafiXML : CUI Hi-Fi + Code Generator (Java Swing, XHTML)
- SketchiXML : CUI Sketching Lo-Fi
- VisiXML : CUI Lo-Fi, MS Visio Plug-in
- FlashiXML : flash renderer
- ReversiXML : reverse engineering from HTML to CUI
- **In development:**
 - TransformiXML GUI : transformation tool
 - Task and AUI editors
 - Tcl/Tk renderer
- **In cooperation :**
 - Teresa (F. Paterno, CUI level)

TRANSFORMIXML API



GRAFIXML



FLASHXML

window_0

Submission Form

Personnal information

label_1

Sexe yes
 no

Family sisters
 brothers

Login Information

Login

Password

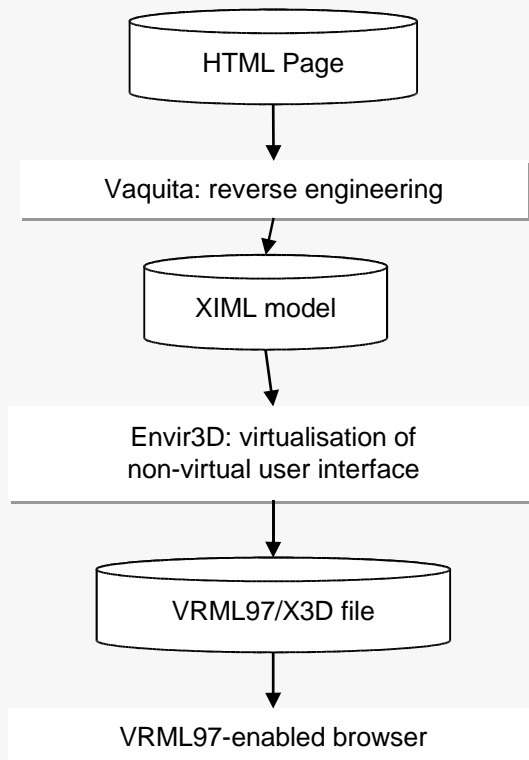
Personnal Information

WebSite

Comment

Virtualisation of UIs

To ensure the UI transition process



Transforms any HTML page into a XIML presentation model

VRML file based on a presentation model expressed in XIML

Example: transforms an existing 2D UI into its 3D equivalent.

TRANSFORMIXML GUI

The screenshot displays the TRANSFORMIXML GUI interface, which is organized into several functional panels:

- Enter Title Here:** A text input field at the top left.
- Select Input Models:** A panel containing a list box, an "Add" button, and a "Delete" button.
- Choose Path:** A panel with two dropdown menus labeled "Choose a starting point here" and "Choose a destination point here", and a checked "Shortcut" checkbox.
- Transformation Composer:** A central panel with a list of transformation systems (e.g., TransSys23, TransSys20, TransSys12, TransSys66, TransSys20, TransSys21, TransSys89), a "system edition" section with buttons for "New Transformation System", "Edit system in grafiXML", and "Edit system in AGG", and an "Attach to current substep" button.
- Rule edition:** A panel on the right with a list of rules (Rule 76, Rule 34, Rule 43) and buttons for "Edit rule in grafiXML" and "Edit Rule in AGG".
- Development steps and sub-steps:** A panel at the bottom with a diagram showing relationships between steps and sub-steps, and buttons for "Apply Transformation" and "Apply step by step".
- Save Result:** A button at the bottom right.
- Back to GrafiXML:** A button at the bottom right.

Mock-up

CONCLUSIONS

Key ideas:

- usiXML represents specification models as BIG WHOLE graphs, it allows the expression of (1) multiple levels of abstraction of UI models (2) development steps (of all sorts) by using conditional graph rewriting rules.
- **Advantages of our approach:**
 - Ontological commitment: our language can be criticized as it is defined in all its dimensions, from concepts to concrete syntax, from task and domain until concrete user interface.
 - Opens the black box of transformation.
 - Decomposes transformation into meaningful chunks: separation of concern at methodological level.
 - Capitalization on transformational heuristics.
 - Multiple-entry points and multiple exit points = flexibility.
 - Model exchange formalism -> tool interoperability.
 - Extendibility, usiXML was planned to receive contributions (3D, multi-modal, multi-surface interaction).
 - Tracaebility of design decisions .

FUTURE WORK

- Pattern expression using usiXML chunks.
- Extension to other modalities (e.g., 3D, multi-modal).
- Integration of other models in the framework (e.g., workflow models?).
- Continue the development of ongoing tools ...

THANK YOU!

See you on www.usixml.org !

