Authoring Transformations by Direct Manipulation

Lionel Villard
Opéra Project – INRIA Rhône-Alpes
Lionel.Villard@inrialpes.fr
Outline

- Motivation
- Transformations authoring
- Incremental transformations
- Conclusion and perspectives
Motivation

- Authoring multimedia presentations for classes of document
- Authoring adaptable presentations
Multimedia presentation architecture

- Negotiation
  - Result of negotiation (Transformation Sheets)
  - Transformation
    - Document
    - Medias
    - Metadata

- Execution
- Formatting
- Presentation Document

Doc'Eng 01 - Atlanta 2001
Author skills

- Classes of document
  - Generally for professional
  - Need heavy infrastructure (database, schema)
- Adaptable presentation
  - For any authors (novice, professional, etc.)
  - Optional schema

No transformation coding => direct manipulation
Outline

- Motivation
- Transformations authoring
- Incremental transformations
- Conclusion and perspectives
Transformation sheets authoring

- Difficulties
  - Programming language (XSLT)
  - Multimedia: multiple dimensions
- Current tools: text editing + debugger

- Our solutions
  - For general purpose transformations: Visual language (cf. Emmanuel Pietriga)
  - For multimedia: direct manipulation through several views
Main principles

Source Views
Hierarchical View

Transformation Views
Template View
Transformation sheets

← General
Mixed Views
Specific →

Target Views
Execution View

XML Document

Transformation

Presentation Document

Doc'Eng 01 - Atlanta 2001
Authoring by direct manipulation

- Build expressions (XPath) in a source view
- Drag and Drop expressions/templates in a target view
- Select and modify the presentation in a target view: "classic" multimedia authoring by direct manipulation
  - Move media object, add new media, set style, set temporal relation, etc.

⇒ Which transformation rules to modify or generate?
Input parameters for XSLT generation

- Target context: author selection
  - Target node, position in parent
  - Template/for-each node
  - Source node

- Resulting nodes of expression
  - Arity (0, 1 or many)
  - Type (element, attribute, etc.)

- Editing mode for the presentation part: local or global
Examples

- **Arity of the resulting node set**
  - 1 : generates an if instruction
  - Many : generates a for-each instruction

- **Target node type**
  - Image : generates a src attribute
  - Text : generates text content
Editing mode: local

- The author expects to have a local modification
  => Produce code to identify the selected node
- Example: insertion of an image in the transformation sheet
Example: image insertion
XSLT modification

<xsl:template match="album" mode="sp">
  <s:region id="title"/>
  <apply-templates select="photo" mode="sp"/>
</xsl:template>

<xsl:template select="photo" mode="sp">
  <s:region>
    <xsl:if test="position()=3">
      <s:region id="rPhoto-3"/>
    </xsl:if>
  </s:region>
</xsl:template>
Editing mode: global

- XSLT production for the nodes selected by an expression
- Must deal with all multimedia dimensions
Example
XSLT modification

<xsl:template select="photo" mode="te">
   <s:image region="rPhoto{generate-id()}">
</xsl:template>

<xsl:template select="photo" mode="sp">
   <s:region>
      <s:region id="rPhoto{generate-id()}">
      </region>
   </region>
</xsl:template>
Outline

- Motivation
- Transformations authoring
- Incremental transformations
- Conclusion and perspectives
Incremental transformation

Two step process

- Preprocessing: transformation sheets analysis
  - Build templates and variables dependency graphs
  - Build re-evaluation rules
    (Editing operation, Instructions to be re-evaluated)

- Incremental processing
  - Execute the instructions computed during the first step
Conclusion

- First experiments
  - for-each/apply-templates instructions
  - Target view: execution view

- Implementation
  - In an existing multimedia authoring tool: Kaomi
  - Incremental processor implemented inside Xalan (Apache Software Foundation)
Perspectives

- At the implementation level
  - Consider more editing operations (remove, modification, etc.)
  - Consider other views (Timeline, etc.)

- At the theoretical level
  - Generation of relative expressions
  - Generation with Schema awareness
  - Take into account transformation rules that modify processor context (param, variable)