



## VXT: Visual XML Transformer

Emmanuel Pietriga\*      Jean-Yves Vion-Dury  
*Xerox Research Centre Europe*  
6 Chemin de Maupertuis 38240 Meylan - France  
{emmanuel.pietriga, jean-yves.vion-dury}@xrce.xerox.com

### Abstract

The ever growing amount of heterogeneous data exchanged through the Internet, combined with the popularity of XML, make structured document transformations an increasingly important application domain. Most of the existing solutions for expressing XML transformations are textual languages, such as XSLT or DOM combined with a general-purpose programming language. Several tools build on top of these languages, providing a graphical environment and debugging facilities. Transformations are however still specified in a textual way using the underlying language (often XSLT), thus requiring users to learn it.

We believe that visual programming techniques are well-suited to representing XML structures and make the specification of transformations simpler. We propose a visual language for the specification of XML transformations in an interactive environment based on a zoomable user interface toolkit and on two target languages specialised in structure transformations: Circus and XSLT.

### 1. Overview

VXT is a visual programming language combined with an interactive environment, written in Java, and specifically designed for programming XML document transformations. It aims at providing high quality visual support for data structure representation and manipulation, so that the user will be freed from maintaining complex mental models of data structures, which play a central role in the transformation specification.

VXT tries to simplify the complexity brought by the different levels of abstraction (transformations rules, document instances, schemas) by unifying them in a single visual representation system (formally studied in [2]) focused on tree structures and based on a variation of *treemaps*

[1]. XML document instances and Document Type Definitions (DTD) can be visualised as background images which can be smoothly zoomed and panned. Transformations are specified mainly visually and consist of a set of rules which can be cascaded (thus following a model close to the one found in XSLT). The left-hand sides of all transformation rules, in charge of selecting nodes and extracting data, are specified in the same window on a transparent layer<sup>1</sup> which can be navigated and zoomed independently of the background. As opposed to XSLT, in which nodes to select and information to extract w.r.t them are expressed in separate statements, VXT uses the multidimensionality of visual languages to merge them in a single expression, called Visual Pattern-Matching Expression (VPME). A VPME resembles the source structure it is supposed to select and extract information from, thus giving the impression to match “visually”. Extracted nodes are filled with a semi-transparent color, conveying the idea of a visual filter, which is close to the mental model associated to the considered transformations.

To improve scalability, each rule’s production is specified in its own window, which contains a synchronised copy of the associated VPME on its left-hand side, and the result-tree fragment produced when this rule is fired on the right-hand side. Links between the two sides indicate the type of operation applied on extracted nodes.

Transformations can be run from the environment or exported as XSLT or Circus source code (Circus is a language specialised in structure transformations positioned at a level of abstraction between XSLT and a language such as Java associated with DOM). The interactive environment also provides mechanisms to assist the user in specifying transformations, such as the automatic generation of VPMEs from source document fragments and the ability to test them on document instances at any time directly from the environment (*progressive evaluation* from Green’s cognitive dimensions framework).

<sup>1</sup>The legibility problem has been addressed by rendering the background image using shades of grey with minimum contrast while VPMEs are rendered in the foreground with highly-contrasted vivid colours.

\*in partnership with INRIA Rhone-Alpes (Projet OPERA), 655 Av de l’Europe, 38330 Monbonnot Saint-Martin - France

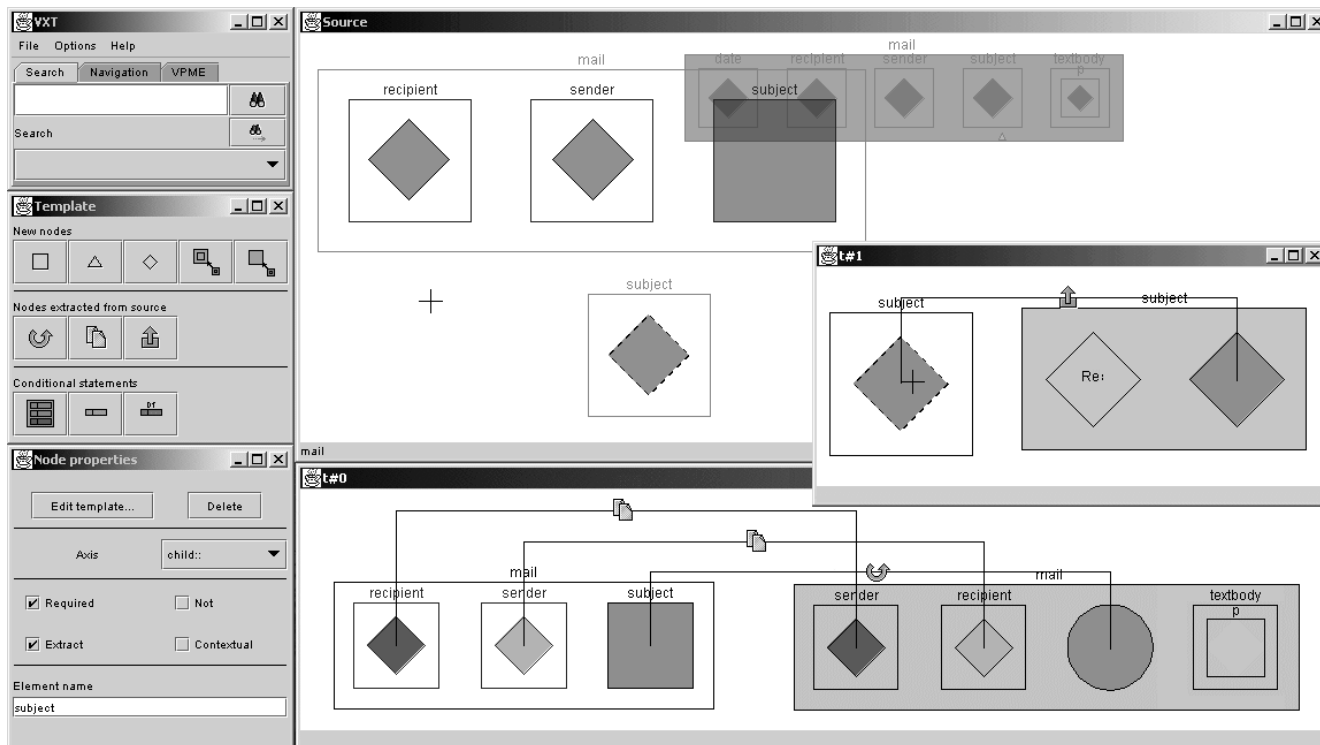


Figure 1. Preparing a reply to a mail

## 2. Example of transformation

Figure 1 shows a transformation that prepares a reply to an e-mail document represented in XML such as the one below:

```
<mail ref="ep47">
  <date>26-01-2001</date>
  <recipient>vion-dury@xrce.xerox.com</recipient>
  <sender>pietriga@xrce.xerox.com</sender>
  <subject>Jazz</subject>
  <textbody><p>Do you know John Coltrane?</p></textbody>
</mail>
```

This transformation is made of two rules, whose *VPMEs* are specified in the Source window (the background layer contains a mail document which can be used to progressively evaluate those *VPMEs*). The two rules are respectively represented in windows #0 and #1. Rule #0 matches mail elements on the condition that they contain recipient, sender and subject children. The textual content of recipient and sender are inverted while transformation rules are applied again on the subject element. The equivalent XSLT template rule is given below:

```
<xsl:template match="mail[recipient and sender and subject]">
  <mail>
    <sender> <xsl:copy-of select="recipient"/></sender>
    <recipient> <xsl:copy-of select="sender"/></recipient>
    <xsl:apply-templates select="subject"/>
    <textbody><p></p></textbody>
  </mail>
</xsl:template>
```

Rule #1 matches subject and appends "Re:" to its content (the dashed octagon indicates that the text node is not required inside subject for the *VPME* to match, but its content is extracted if present):

```
<xsl:template match="subject">
  <subject>
    Re: <xsl:value-of select="text()"/>
  </subject>
</xsl:template>
```

Icons attached to links between nodes extracted from source documents and the associated ones in the result-tree represent the type of operation to apply:

- = copy nodes
- = apply transformation rules
- = extract textual content

## References

- [1] B. Johnson and B. Shneiderman. Treemaps: a space-filling approach to the visualization of hierarchical information structures. *Proceedings of the 2nd International IEEE Visualization Conference, San Diego*, pages 284–291, 1991.
- [2] E. Pietriga and J.-Y. Vion-Dury. A formal study of a visual language for the visualization of document type definition. *IEEE Symposium on Visual Languages and Formal Methods (HCC'01)*, 2001.