Proxima - a generic presentation-oriented XML editor

Martijn Schrage, Johan Jeuring, Lambert Meertens, Doaitse Swierstra

March/April 2003




Editing structured documents

All editing is structure editing. Sometimes the structure is very simple, for example when editing text.

In 1999 we started on the Proxima project, with the goal to build a generic, presentation-oriented editor for XML documents.

With XML documents we mean structured documents, and with generic we mean that the editor can handle structured documents of arbitrary type (for XML documents: with a DTD or a Schema).

Presentation-oriented means that we always see a presentation of the document. The presentation might be the 'raw' XML presentation, but also a more advanced presentation of a document.

This talk

In this talk I will

- Discuss example instances of Proxima.
- Give a list of requirements for Proxima.
- Briefly give the architecture of Proxima.

A Program Editor

Suppose we want an editor for a programming language such as Haskell. Then we want:

- Syntax highlighting on a semantic level (distinguishing for example type variables from expression variables);
- Derived information, such as the type of functions (including type errors), and variables in scope, to appear in the presentation of the document;
- Automatic layout, but also user-specified layout;
- Structured edit operations, such as cutting and pasting declarations, as well as text input, without mode switching or new windows.

```
let x=1;
    y=2
in (x+y)/2
```

- Rename a variable.

Word processing

We want an improved MS Word.

- Structural view on the document: *How* **do I edit** this? versus `<bf><it>How</it> do I edit</bf> this?`
- Edit operations on derived information: edit a title in the table of contents, or even move a chapter in the table of contents.

An Equation editor

Equations usually possess a lot of structure (compared with texts).

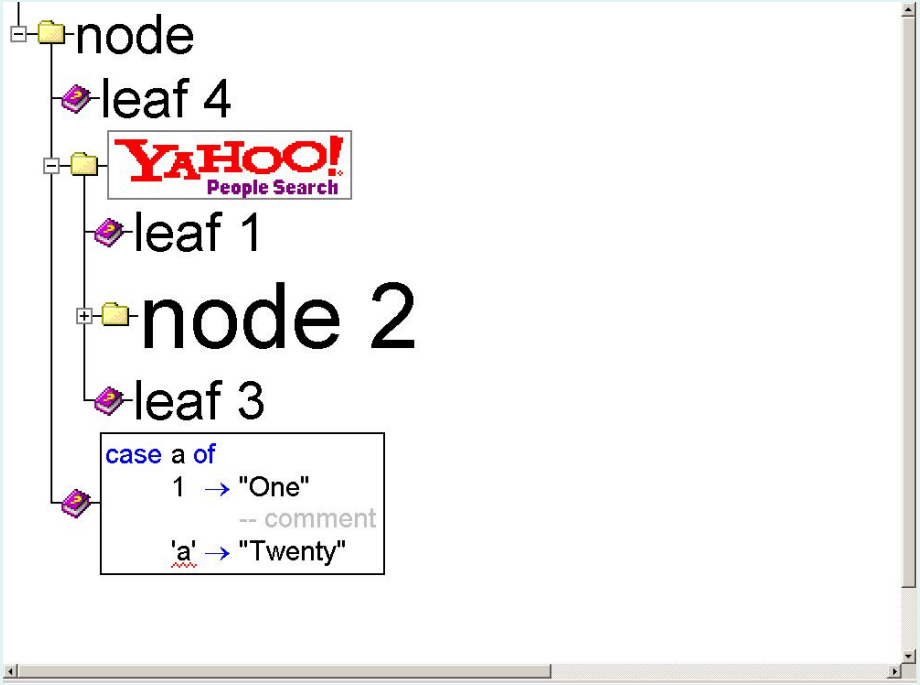
- Advanced two-dimensional presentations.
- Both text input as well as structure edit operations, without mode switching. For example, typing 2+3*8 has the same effect as constructing this expression via selecting +, *, and the appropriate integers.

  It is not clear to us (yet) how to textually input two-dimensional formulae.
- Easy drag and drop of subexpressions.
- Domain specific transformations: `a*(b+c)` $\Rightarrow$ `a*b+a*c`.

The equation editor can be seemlessly integrated in any desired editor.

A Tree Browser I

A tree browser is a hierarchical view on tree structures. The Java Swing library has an implementation. We want to be able to specify the presentation of a tree browser.



A Tree Browser II

- For program sources, the tree view can be used in a separate window to navigate through code.
- The 'state' of the tree view is kept locally at presentation level, and saved on exit.
- Drag and drop are supported.
- The tree view is customizable: it can be changed to horizontal instead of vertical, for example.

A Tax Form

A tax form is designed by the tax office, and filled out by tax payers. Both kinds of users use the same document type, but different presentations.

A tax form requires a table-oriented layout, with support for user interface widgets such as text fields, radio buttons, and selection lists.

- It should be possible to let the structure of the presentation depend on values in the document. For example, if you fill out 2 in the text field for number of jobs, two text fields for job particulars should appear.
- Only the tax office can edit the structure of the tax form, tax payers can only fill out text fields etc.

Requirements for Proxima

Proxima

- is a generic structure editor for XML documents that are valid with respect to some DTD;
- supports computations over the document;
- has a graphical presentation language with a powerful mapping formalism;
- supports edit actions on all levels, including the level of derived structures;
- supports modeless editing;
- supports local state.

An Architecture for Proxima

Proxima has a layered architecture:

- Document
- Document extended with computations
- Presentation
- Arrangement

Each of the layers has a local state, and a computation sheet.

The computation sheet is used to either compute the extended document from the document, or the presentation from the extended document.

A layer

Between each pair of layers there is a mapping from the upper layer to the lower layer using the computation sheet, and a mapping that translates edit actions on the lower layer to edit actions on the upper layer. Document nodes have a name, which is preserved throughout the layer structure.

Edit actions are applied at the level where they should be applied: an edit action that changes the layout of a program never reaches the document level, a structure transformation skips all levels except the document level.

Related work

A lot!

- Syntax-directed editors: Synthesizer generator, ...
- Syntax-recognizing editors: Pan, Ensemble, ...
- Editor toolkits: Amaya, Thot, Visual Studio, ...
- XML editors: XMetal, XML Spy, ...

Proxima improves upon these editors on important requirements.

Conclusions and Future work

A prototype version (Windows only) is available on CD.

A lot of implementation work remains to be done (platform independence).

We want to produce a number of example editors, to obtain a domain-specific language for specifying (particular kinds of?) editors.

Explore the transformation capabilities of the editor.