χ–IDE

XML Integrated Development Environment
Specifications Document

Colin Hartnett (cphartne) – 7 February 2003

1 Project Description

There exist many integrated development environments that make large projects with many interdependent files easy to manage. Currently, there is no full-featured noncommericial Linux IDE for XML (Extensible Markup Language). Many specialized XML text editors exist, but there is no editor that incorporates file editing, project management, and testing features.

AUDIENCE

A relatively robust IDE would be warmly welcomed by the XML community at large. A good local test user base could be found within the Computer Science Department – specifically, students in CS196–9: Document Engineering.

2 Specifications

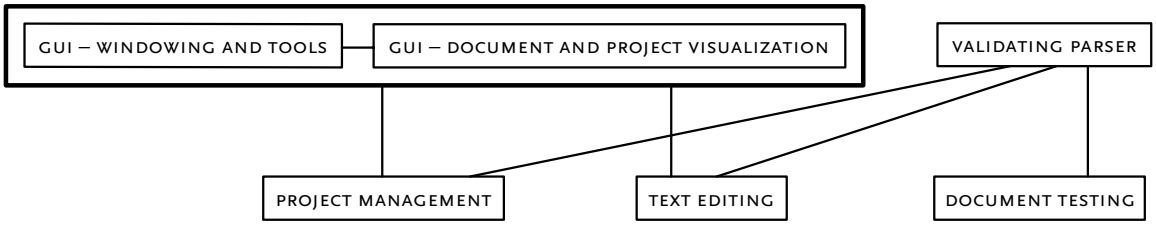
An integrated development environment is a complex program. It is not used linearly; thus, the specifications will be organized by module.

GENERAL DESCRIPTION OF MODULES

User Interface – Tools and Windowing. Provides the window, toolbar, and menu widgets.

User Interface – Document and Project Visualization. Produces a collapsible graphical tree representation of a document or project.

Validating Parser. A W3C–compliant validating XML parser. Also provides basic



information about documents such as DTD, character encoding, &c.

Text Editing. Provides text editing capabilities. Handles fancy editing features such as syntax highlighting and auto indenting.

Project Management. Groups files into projects. Handles organization of files and saving of projects.

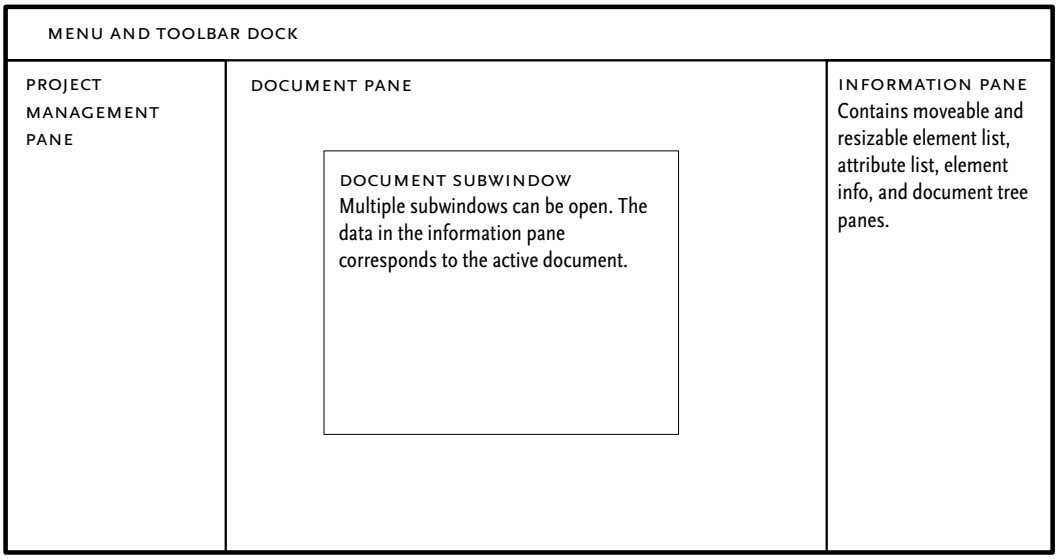
Document Testing. Allows execution of XPath commands and XSLT transformations.

USER INTERFACE — TOOLS AND WINDOWING

The user interface follows standard WIMP protocol and has a menu and toolbar across the top of the main window.

The main toolbar contains buttons to manipulate single documents. Buttons are: new, open, save, and validate. A separate project toolbar includes buttons to save and open projects, add files to and remove files from projects, sort the project view in various ways, and add project files to 'soft' groups.

The body of the window is divided horizontally into three areas: the project pane, the document pane, and the information pane. The project pane displays the graphical representation of the current open project. The document pane is where subwindows containing open documents reside. These documents can be edited. Windows in the document pane can be arranged in standard ways: cascade, tile vertically, and tile horizontally. The information pane contains four main items, all of which should be moveable and resizable: The element information pane displays information about the current element (determined by the cursor position). The element list displays a list of all the



elements in the specified DTD. The elements that are not valid at the current position should be 'greyed out', that is, displayed in such a way that it is obvious to the user that they are invalid. The attribute list displays all the valid attributes of the current element. The document tree shows a tree-like graphical depiction (outline in next subsection) of the current document.

USER INTERFACE — DOCUMENT AND PROJECT VISUALIZATION

Both documents and projects will be graphically represented as trees much like those found in common file manager applications. An XML document is a tree, therefore this module simply takes the parsed document and produces a tree representation where each node of the tree is an element in the document. Attributes of elements are represented on a line underneath each element prefixed with a marker (icon) that indicates the data on that line is an attribute. The XML document cannot be modified through this tree. (*Version 2:* Actually editing the document with the graphical tree.)

Projects are represented by file much like those in a file manager. They can be sorted by filename, type (DTD) , and character encoding. When a file is sorted by one of these properties, each unique value is displayed as a folder, and each file is displayed as a file in that folder. files can also be 'soft' grouped, that is, files can be added to zero, one, or many specialized groups (e.g., a computer science paper in a project that represents a journal could belong to both the "computer graphics" and "computational biology" groups). files can also be sorted by group. When a file is double-clicked, it should be opened in an editor window.

This module also graphically represents lists of valid elements and attributes obtained from the parser. These elements should be clickable and insert the tag into the document at the current cursor position.

VALIDATING PARSER

The parser module needs not only to parse and store an internal representation of the document, it must also return basic information about the document, such as its type. The parser must also return information about the document type. The parser must not die unless a document is not well-formed. In its internal representation, each node (element or attribute) must include information on whether it is valid, invalid and not in the DTD, or in the DTD,

but in an invalid position. It must also include basic element data, such as namespace.

Given a physical point in the document, the parser must be able to return information obtained from the DTD about the current element consisting of its namespace, name, datatype, enum values, occurrence, and annotation. The parser must also be able to return a list of valid elements at the given point.

TEXT EDITING

The text editing module provides basic text editing capabilities. It also includes syntax highlighting. Valid tags should be highlighted with one color. Invalid tags should be highlighted using two different colors: one for tags that are not anywhere in the DTD and another for tags that are not in a valid position. Attributes should be highlighted using one color for valid attributes and another for invalid attributes. Tags should be automatically closed (e.g., once I type the ">" in "<foo>", "</foo>" should be inserted and the cursor should be placed between the two tags).

PROJECT MANAGEMENT

This module supports adding and removing files to projects. It must store basic data about each file including its name, type, encoding, and 'soft' groups. It saves projects as XML files. It should also be able to open (with the help of the parser) any previously saved project files.

DOCUMENT TESTING

The testing module allows a user to execute XPath expressions and XSLT transformations and view the results as a new XML file in an edit window. This is relatively straightforward.

3 Version 2 Possibilities

More automated editing features could and should be added to version 2. In addition to the aforementioned addition of graphical editing, tools could be added to reverse engineer a DTD from on a prototype document, and wizards could be added to assist in creating DTDs. Project management could be fleshed out with features such as extended search and replace. More technologies could be supported for testing, such as the XML Schema and RELAX languages, which are more robust ways of specifying document types,

and the XML Query Language, which allows an XML document to be queried much like a database.

4 Implementation Suggestions

There are good, free implementations of many of the components of this system. The Apache XML Project provides Xerces, a validating parser, and Xalan, a stylesheet processor. Both are available in C++ and Java versions. These components could be used as core elements in the parser and testing modules.

5 Nonfunctional Requirements

The goal of any development environment is to increase productivity. This project should provide an environment that allows users to quickly and easily work with XML–based projects. This program have the potential to go above and beyond even powerful text editors like Emacs with its ability to present multiple perspectives of the document to the developer simultaneously. In order to enhance productivity, the user interface must be well-designed, the program must execute quickly, files must open and close quickly, and editing must be intuitive. These are not concrete, testable concepts, but a general consensus can be reached by analyzing the performance of commonly used programs.

6 Updated Requirements

The list of basic features was trimmed to a more manageable level. The requirements are the same in content, but the priorities have been dramatically adjusted.

USER INTERFACE

- "Unlimited" open files
- More than one document visible at once
- Collapsible tree representation of project
- Collapsible tree representation of current document
- Lists of elements valid in current position
    - Grey out elements not permissible at current position (*medium*)
    - Visually indicate required elements at current position (*medium*)
    - List of attributes for current element (*medium*)

- Display information obtained from DTD about current element
- Automatically generate DTD from a document (*low*)
- Wizard-like support for specifying DTDs (*low*)

TEXT EDITOR

- Basic syntax highlighting
- Syntax highlighting based on DTD (*medium*)
- Smart indenting
- Auto tag completion
- Unlimited undo (*medium*)
- User-definable macros for tag insertion (*low*)
- Unicode support (*low*)

PROJECT MANAGEMENT

- Group multiple files together in a project
- Save projects in an XML format
- Group files together (*medium*)
  - Files can belong to multiple groups (*medium*)
- Display files sorted by:
  - Name
  - Type
  - Date modified
  - Groups (*medium*)

TESTING

- Method of executing XPath expressions
  - Return results as XML
  - Return results graphically (*low*)
- Execute XSLT transformations (*medium*)
- Validate against:
  - DTD
  - XML Schema (*low*)
  - RELAX (*low*)
- Realtime validation
- Step-by-step XSLT debugging (*low*)
- Execute XML Queries

7 Risks and Dependencies

Writing an XML parser is a large task – one that could seriously stall the project. On the other hand, if the project is implemented using an external parser and stylesheet processor, a huge external dependency is introduced. Luckily, XML is a public standard, and most parsers – including Xerces – implement the DOM (Document Object Model), which is simple, robust way of retrieving information from XML documents. Since the standards and specifications were written well before any parsers were developed, the language and protocols are concretely defined and well-documented.

This project would require the members of the team to at least have a basic knowledge of XML. Those developing the parser and testing modules need to have a much more intimate knowledge of the language and protocols.

8 Definitions

XML (*Extensible Markup Language*) – A markup language that allows users to represent data in plain text. Its syntax is similar to that of HTML, but authors can create their own elements ("tags" in HTML lingo) and attributes. Some common uses for XML include representing mathematical formulae, vector graphics, remote function calls, and syndicated news feeds.

DTD (*Document Type Definition*) – A method of specifying the legal structure of an XML document.

Validating parser – An XML parser that, in addition to parsing and allowing access to the document, indicates whether or not the document conforms to its DTD.

XPath (XML *Path Language*) – A method of selecting information from an XML document. XPath's syntax is similar to Unix paths.

XSLT (*Extensible Stylesheet Language Transformations*) – A method of transforming one XML document into another of a different format.

9 Further Reading

The official World Wide Web Consortium (W3C) XML site
http://www.w3.org/XML/

The official W3C XSL and XSLT site
http://www.w3.org/Style/XSL/

The official W3C XPath site
http://www.w3.org/TR/xpath/
The Apache XML Project
http://xml.apache.org/