XEBRA: The Design and Implementation of Integrated Programming Environment for XML Processing and Browsing

Norio Touyama[†], Yasuyuki Hirakawa[†], Takashi Hattori[‡], Tatsuya Hagino[‡]
[†]Graduate School of Media and Governance, Keio University
[‡]Faculty of Environmental Information, Keio University
5322 Endoh, Fujisawa, Kanagawa 252-8520, Japan
{next,chibao,hattori,hagino}@tom.sfc.keio.ac.jp

ABSTRACT
We present the design and implementation of XEBRA system. XEBRA is an integrated programming environment for XML processing and browsing on which users can build their own XML processing applications. XEBRA has a lisp interpreter as its main control system. The lisp interpreter provides programming interface for users. It also provides XML manipulation routines and data structures devoted to XML. We designed the XEBRA architecture to be extensible by modules. We implemented some modules which have functionalities for XML and web related common technologies such as XSLT, XSL-FO, HTTP and CSS to XML conversion. Users may build customized programs by combining functionalities provided by these modules. Finally, we show an HTML browser as an example application of the system. We demonstrate that this browser shows potential of our system and its design.

Keywords
browsers and tools; language; XML; XSLT; XSL Formatting Object

1. INTRODUCTION
After appearance of XML [1], the world of World Wide Web is now getting into the second generation based on XML and XML family technologies like XSLT [2]. Compared with the first generation web computing that is based on HTML, the second generation web computing that is based on XML will provide openness, flexibility and extensibility for users by using XML as a meta language for all documents and data. Our main interest of this paper is how we can bring potential of XML into user's processing and browsing environment.

Existing web environment for users consists of HTML browser and its extension. However, handling of XML documents is more than just browsing. For example, we may want to gather XML files from various sites, combine these files into an XML tree, transform this tree into an XSL Formating Object [3], and then browse it. In such cases, we require programmable environment on which we can build our own XML processing applications.

In order to satisfy our interest, we developed XEBRA (eXtensible Environment for BRowser Architecture) system. We designed this system as integrated programming environment for XML processing. Users of XEBRA have ability to create their own customized XML application.

2. DESIGN AND IMPLEMENTATION OF XEBRA

2.1 Main system as a lisp interpreter
In our system XEBRA, we introduced a lisp interpreter as its main control system. We employed modular extensible architecture in the main system. We implemented some functions of XML family technologies as modules. Figure 1 shows the system overview of XEBRA.

The lisp interpreter provides interactive programming environment for users. With the lisp interpreter, users can build and execute their own program in an interactive session. Our lisp interpreter has both functions and data structures devoted to XML processing. Furthermore, users can freely combine modules for their own applications by calling modules as lisp functions.

The lisp interpreter and modules are written in C language. Because we wanted to make free both lisp users and module writers from memory management, we introduced memory allocation system with mark and sweep garbage collection.

2.2 Modules
We implemented modules described below. In order to make C functions of modules available to lisp programs, each module has glue functions for lisp. Users can build their own applications by combining these modules in their lisp programs.

1. **XML parser module**

   Our parser has ability to parse well-formed XML documents [1] and produce XML trees which can be handled by lisp programs. Our parser also can process XML namespaces correctly.

2. **HTML parser module**

   Our HTML parser module can parse legacy HTML [5] documents and simultaneously convert them into XHTML [4] compatible structure. Its input is an HTML document and output format is the same as XML parser's one.

3. **XSLT (XSL Transformation) module**

   In addition to ability of flexible XML tree processing by writing lisp programs, users can utilize XSLT

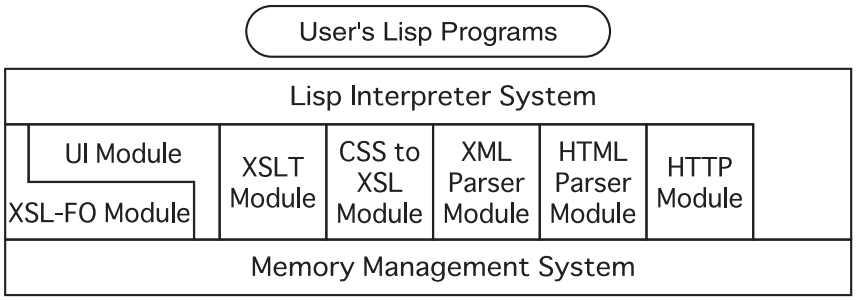

Figure 1: XEBRA system overview.

stylesheets [2] for general transformations of XML trees with this module. XSLT stylesheets and lisp programs are complementary to each other.

4. **HTTP module**

   Lisp programmers can use our HTTP [6] module for collecting XML documents and data from web servers.

5. **XSL-FO (XSL Formatting Object) rendering module**

   XSL-FO is *an XML vocabulary for specifying formatting semantics* [3]. We employ XSL-FO as basis for rendering and displaying XML data. Our XSL-FO module converts XML trees described with XSL-FO vocabulary into device dependent drawables and displays it directly on screens.

6. **CSS to XSL stylesheet conversion module**

   With this module, users can convert arbitrary CSS stylesheets into XSLT stylesheets which convert XHTML trees into XSL-FO trees.

7. **User interface module**

   For implementing basic browsing functionality, XEBRA provides GUI interfaces for browsing. This module is created with GTK+ toolkit. Currently, XEBRA provides basic window user interface just enough to implement a simple HTML browser.

3. APPLICATION EXAMPLE — HTML BROWSER

XEBRA is an integrated environment for XML processing and it is not intended only for implementing web browsers. However, in order to show basic ability of the system, we implemented an HTML browser with XEBRA. Main part of this simple HTML browser is expressed in 14 lines of lisp program. It consists of roughly following three steps.

1. It loads an HTML document from a specified URL.

2. Then, it transforms the loaded document tree into an XSL-FO tree by calling the XSLT module with an XSLT stylesheet which converts HTML trees into XSL-FO trees.

3. Finally, it calls the XSL-FO rendering module.

Customizing this browser is relatively easy. For example, it is easy to insert a lisp program which call the XSLT module with a stylesheet which summarizes H1 element of HTML documents and display their summarized versions.

After implementing the basic part of XEBRA system, we implemented each module independently. Then, we tried to write the HTML browser as a lisp program. Nevertheless the design of each module was left to implementor of each module, writing the lisp program of the HTML browser was far easier than we anticipated. With this experience, we proved the effectiveness of our basic design of XEBRA and potential of the system.

4. ACKNOWLEDGMENT

This research was supported by IPA (Information-technology Promotion Agency, Japan).

5. REFERENCES

[1] T. Bray, et al. , Extensible Markup Language (XML) 1.0 (Second Edition) , W3C Recommendation 6 October 2000 , http://www.w3.org/TR/REC-xml

[2] James Clark , XSL Transformations (XSLT) Version 1.0 , W3C Recommendation 16 November 1999 , http://www.w3.org/TR/xslt

[3] S. Adler, et al. , Extensible Stylesheet Language (XSL) Version 1.0 , W3C Working Draft 18 October 2000 , http://www.w3.org/TR/xsl/

[4] Steven Pemberton, et al. , XHTML 1.0, The Extensible HyperText Markup Language - A Reformulation of HTML 4 in XML 1.0 , W3C Recommendation 26 January 2000 , http://www.w3.org/TR/xhtml1/

[5] Dave Raggett, et al. , HTML 4.01 Specification , W3C Recommendation 24 December 1999 , http://www.w3.org/TR/html4/

[6] Fielding, R. et al , RFC 2616 , HyperText Transfer Protocol, HTTP/1.1 , June, 1999 , http://www.ietf.org/rfc/rfc2616.txt