Persona: a Framework to provide Adaptive Presentation for Web Documents

Junichi Suzuki, Gaku Tashiro, Yutaka Abe and Yoshikazu Yamamoto

Department of Computer Science,
Graduate School of Science and Technology,
Keio University
Yokohama City, 223-8522, Japan
+81-45-563-3925 (Phone & FAX)
{suzuki, manabu, yutaka, yama}@yy.cs.keio.ac.jp

Abstract

The current issue in the web document management is that every document has only one presentation and/or content across its entire readership. An important aspect in this issue is how documents can be customized context-sensitively. This paper addresses the document customization with the environmental context, and describes how we can provide the adaptive presentation for HTML documents. We developed the Persona system, which is a framework allowing to mine an appropriate document presentation in a given context. It inspects the client-side environment information and generates a presentation suited to the environment. Persona makes it easy to process and distribute web documents on the fly.

Keywords

Adaptive presentation, Web documents, Document management, Personalization

1. Introduction

It is now recognized that the Web is a huge distributed document repository. Various approaches to retrieve, traverse or process Web documents have been proposed in both academic and commercial areas. For every approach, the most important aspect is the context-sensitive customization of the presentation and/or content. It means documents can be customized so that they are suited to the context in users' accesses. The context includes:

- what kind of environment a client application uses,
- who accesses a web document,
- which document the user wishes to browse,
- and how the user behaves, or has behaved.

This paper focuses on the first context: the client-side environment information, and discusses how web documents can be customized with the context. Our system named Persona provides the environment independent presentation of web documents.

The remainder of this paper is organized as follows: Section 2 describes the motivation and existing approaches to provide environment independence of web documents. Section 3 outlines the conceptual framework of the Persona system. Section 4 describes design of Persona. Section 5 presents applications using Persona. We conclude with a note on the Persona's the current status of the project and future work in Section 6 and 7.

2. Environment Independence of Web Document Presentations

Currently, we can access web documents with various devices: from personal electronic devices such as PDAs (Personal Digital Assistants), pagers and cellular phones into computers including palmtop, laptop and desktop ones. Most documents, unfortunately, are designed for the display on desktop computers that have color monitors with at least 640x480 resolution. The disconnect can lead to 4-to-1 (or greater) ratio of designed v.s. available screen area [1]. In such a situation, documents are not navigable and difficult to browse. The document presentation also depends on the current window size or gray/color difference of monitors. It is required to provide means to display arbitrary web documents independently of client-side environments.

In general, there are four approaches to display HTML documents on multiple client-side environments [1]:

- device-specific authoring
- multiple-device authoring
- client-side navigation
- automatic re-authoring.

Device-specific authoring means users access a pre-defined content specific to a client-side environment. This approach provides the accurate presentation suited to the client, but content providers have to prepare a set of contents manually.

Multiple-device authoring identifies a range of devices and provides mappings from a single source document to a set of rendered documents. Typical example of this approach is to use stylesheets such as CSS (Cascade Style Sheet) [4], XSL (eXtensible Stylesheet Language) [5] and DSSSL (Document Style Semantics and Specification Language). A stylesheet defines a set of display attributes. They can be chosen by users selectively, or chosen by content providers statically [2]. This approach eliminates for content providers to author multiple contents for different environments. It reduces manual work per a document than the device-specific authoring approach, but still requires more work than simply authoring for a single client environment.

Client-side navigation allows users to navigate a single document by changing its displayed portion at any given time. Typical examples include JavaScript-enable or Dynamic HTML (DHTML) enable contents that can control the portion that a document is displayed. This approach has an advantage that users can control the presentation based on their own preferences. However, it cannot always apply to arbitrary documents. It is likely that client-side scripts should be written only for any set of documents.

Automatic re-authoring involves developing software that can take an arbitrary documents and transform the document so that it can be displayed on the target client environment. This process can be performed either on the client, server [2] or an intermediary place like a HTTP proxy server [3]. There are two possible re-authoring policies: syntactic and semantic. The

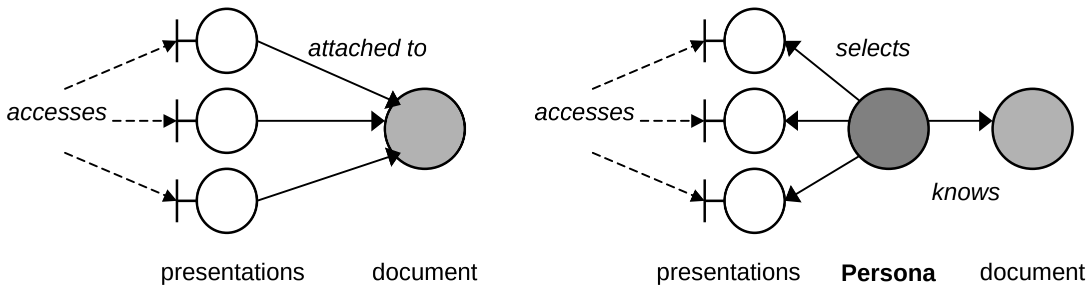

Figure 1: In general, multiple presentations can be attached to a single document. Users access the document through a selected presentation (left). In Persona, an interface agent intermediates between a document and its presentations (right). It adapts the presentation for a specific user.

syntactic conversion operates on the structure of a document, while semantic conversion relies on some understandings of the document. The automatic re-authoring approach can reduce the manual work by users or content providers. However, providing appropriate presentations depends on the intelligence of the re-authoring software.

As described above, each approach has advantages and drawbacks. Our Persona system takes a hybrid approach using the multiple-device authoring, client-side navigation and automatic re-authoring techniques.

3. Persona Overview

This section presents Persona's conceptual overview.

In general, documents can have multiple presentations, and users access them through a selected presentation (left of Figure 1). The presentation can be determined statically at the time when the document is authored, or dynamically after it is published. Also, the presentation may be selected automatically or manually. Current stylesheet mechanisms intend the static and manual selection. A content provider has to assign a specific stylesheet when they author a document. The stylesheet cannot be changed at runtime. In order to provide multiple presentations for a document, it is required to prepare multiple documents that apply different stylesheets, as described in the previous section. This solution potentially have a content inconsistency problem when the original document is updated.

In contrast, Persona intermediates between client applications (i.e. HTML browsers) and documents (right of Figure 1), and allows to rearrange the presentation of web documents dynamically and automatically. Content providers only have to prepare a single document and, the decision which presentation should be applied can be delayed until runtime. The selection process is performed by Persona using its automatic re-authoring technique. Persona can transform HTML documents with the following strategies:

- Outlining
- First sentence replacement
- Client-side text manipulation
- Image reduction and elision
- Dynamic CSS attachment

The outlining strategy transforms documents so that the content of each section is elided from the document and every section header is converted into a hyperlink to the content. The

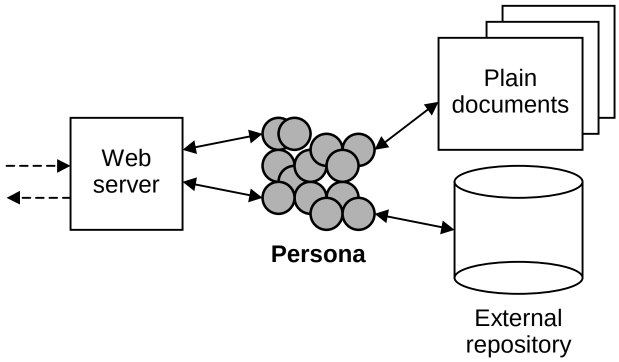

Figure 2: Persona receive a request through a Web server and generates an appropriate presentation of the documents.

converted document seems to be a table of contents. This strategy can be applied well for structured documents such as technical papers and manuals. Persona extracts HTML header tags, from H1 to H4 tags, and outlines the document.

The first sentence replacement strategy transforms documents so that the content of each section is elided from the document except from the first sentence. This strategy can be applied even when no section headers are present in the document. It can be also used with the outlining strategy. Persona replaces the first sentence of a text block between <P> and </P> tags with the entire block.

The client-side text manipulation strategy transforms static documents into dynamic ones. This strategy is an example of the client-side navigation approach described in Section 2. It can be used together with the outlining and first sentence elision. Persona can re-author documents so that the text scrolls automatically, and it can shrink and stretch with the user's mouse manipulation. These capability is performed on HTML browsers using JavaScript.

The image reduction and elision strategy transforms images by pre-defined scaling factors, and makes a hyperlink from the reduced image to original one. Persona can reduce images with 75%, 50%, 25% and 0% scaling factors. 0% means the image elision. An elided image is replaced with the text of its ALT attribute.

The dynamic CSS attachment strategy replaces a document's CSS definitions dynamically. This strategy is a dynamic variant of the multiple-device authoring approach, described in Section 2.

4. Persona's Design

This section describes the design of Persona to achieve the concepts introduced in Section 3.

As depicted in Figure 2, Persona is intended to run at the back-end of a web server. It can be implemented with server-side components like a CGI (Common Gateway Interface) program and Servlet component [6]. Our initial prototype system was implemented as a CGI scripts with Perl. The current Persona is written with Java and integrated into OpenWebServer [7, 8], which is our reflective and extensible Web server.

When a user accesses a Persona-enable web server, the client-side environment information passed to the web server. The information is obtained mainly by a script with JavaScript that is run on a HTML browser, and then embeded into a URL description. The script can check the name and version of browsers, the resolution of monitors and the Java

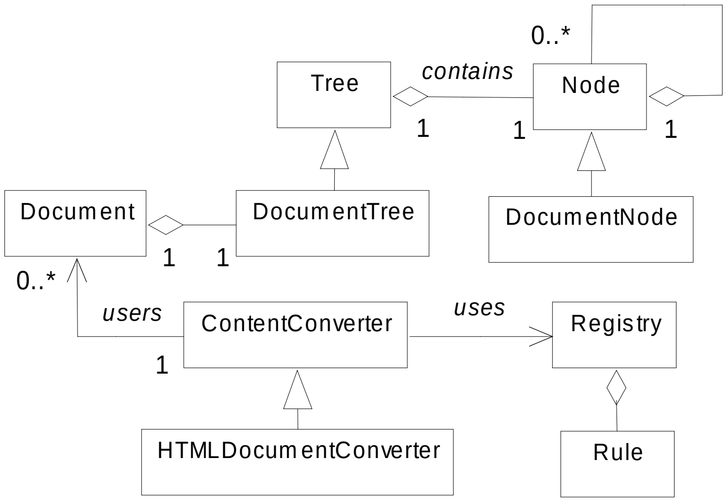

Figure 3: Class structure of core components in Persona.

enable/disable configuration. When Persona accepts a request for a HTML document through a web server, it send a dummy HTML document that includes the JavaScript script. Then, the script automatically requests the target HTML file again using the URL that contains the client environment information. A sample URL description is like:

```
http://www.yy.cs.keio.ac.jp/index.html?resolution=800*600
```

Users can explicitly specify the URL with environment information, instead of using the script. Users may wish to specify the re-authoring strategy explicitly like:

```
http://www.yy.cs.keio.ac.jp/index.html?policy=outlining
```

Other environment information including the name of operating systems can be inspected with HTTP headers. Persona can be configured to obtain the information from HTTP headers.

Persona re-authors the requested document based on the obtained client-side environment information. It maintains the mappings (rules) between the environment information and re-authoring strategy. This rules are defined with a configuration file based on XML (eXtensible Markup Language), and read by Persona at the system startup time. A re-authored document is named with a temporal name other than original one, and sent to a HTML browser. It is cached in the system using LRU algorithm.

Figure 3 depicts the class structure of core components in Persona. `Document` represents each HTML document and has its syntax tree, which is created when the document is requested from clients. A syntax tree is designed with a recursive association between `Node`s and created with a HTML parser. Every syntax trees are cached in the system for the later use. `ContentConverter` and `HTMLDocumentConverter` are classes that re-author documents. These classes use `Rule` and `Registry` to maintain re-authoring rules.

5. Applications

This section presents applications using Persona, and describes the future work.

Figure 4 shows a sample re-authoring with the outlining strategy (see also Section 3). This example uses an technical paper that has a clean document structure. Persona extracts section

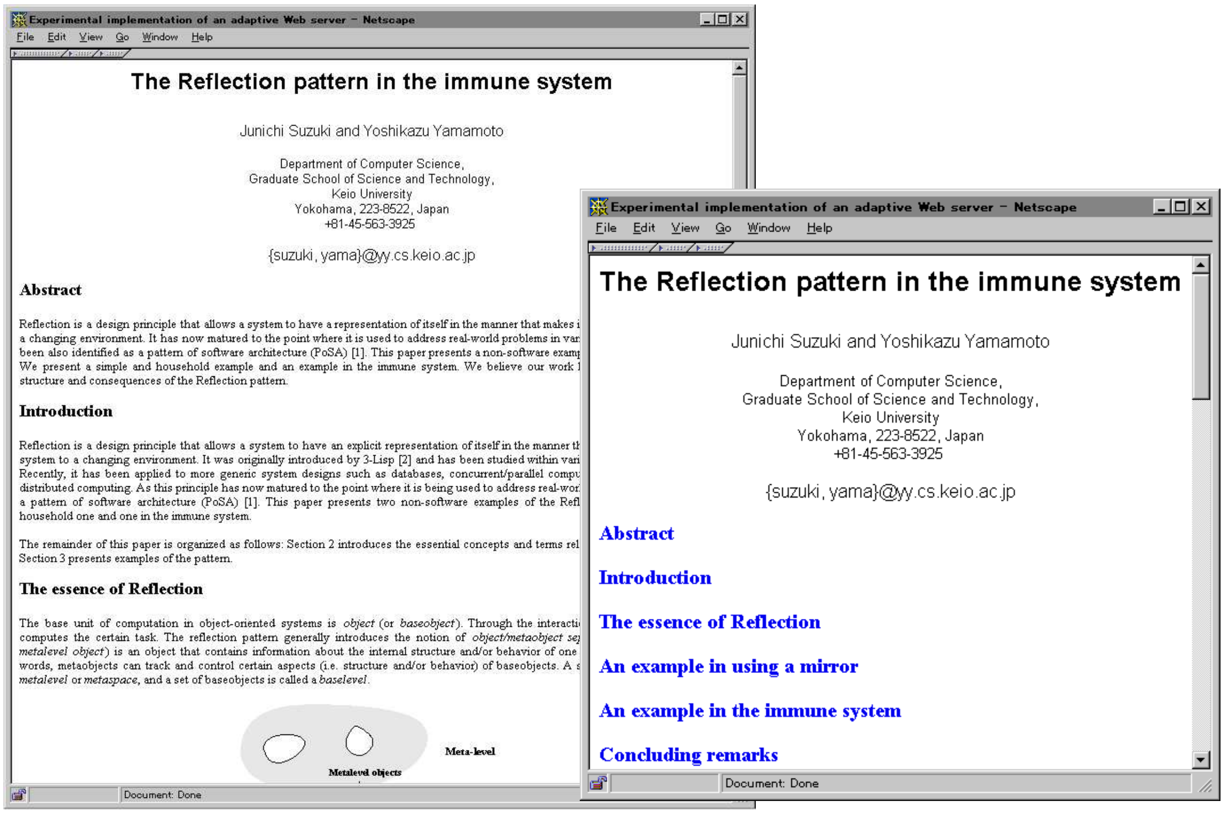

Figure 4: A sample re-authoring with the outlining strategy.

headers tagged with <H3>, and every header is converted into a hyperlink to the content.

Figure 5 shows a sample re-authoring with the client-side text manipulation (see Section 3). This example uses a software design documents [9-12]. Persona transforms a textual presentation into DHTML-enable one in which the content stretches and shrinks when users point headers with mouse. This DHTML content can be displayed only on Microsoft Internet Explorer (MSIE), because it uses incompatible scripts with other browsers. Therefore, Persona generates this presentation only when a browser is MSIE. Also, Persona can generate a Java-enabled graphical presentation using ConceptMap applet [13] (Figure 6). This presentation is created only when a browser is Java-enable.

We are working for further applications with Persona to improve its functionality. Persona is being extended for the area of distant education and virtual training. To develop such a deployment, we are adding a capability to recognize users accessing documents [14]. Also, we are integrating the capabilities that generate presentations for HTML and XML documents [2, 14]. The future version of Persona will be able to deals with other contexts than the environmental context.

6. Future Work

We are developing an alternative mechanism to inspect the client-side environment information using cookies in HTTP headers [15]. It can eliminate to use a dummy HTML document to check the environment (see also Section 4) and reduce the communication overhead. An document repository, shown in Figure 2, is being developed with an object-

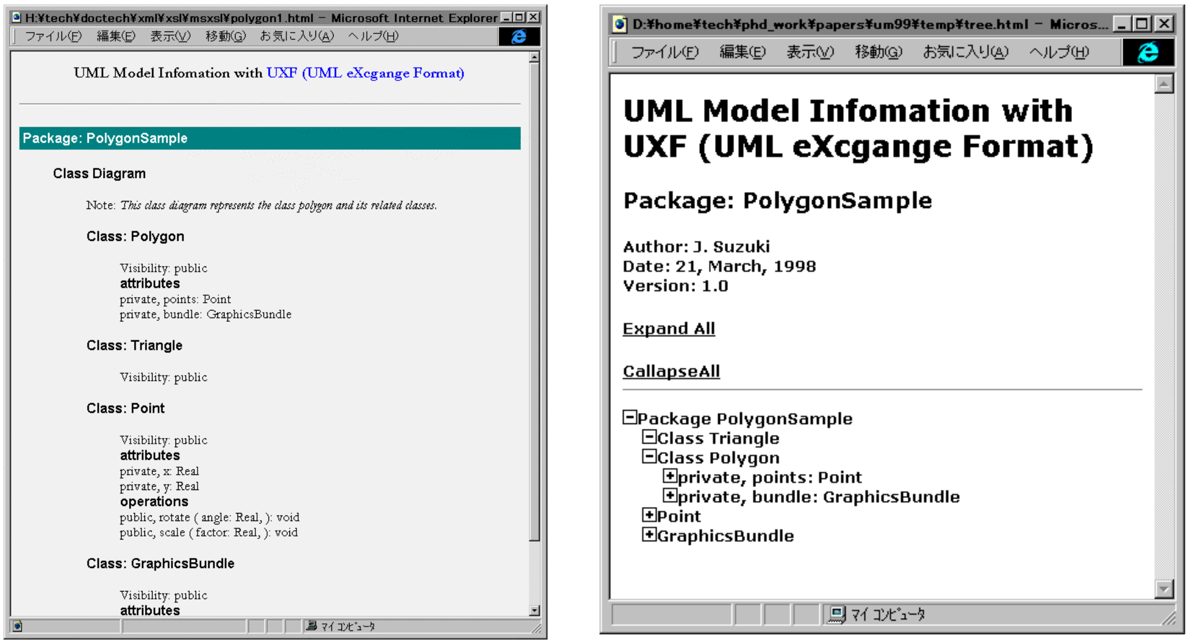

Figure 5: A sample re-authoring with the client-side text manipulation strategy.

oriented database. It allows the persistence of documents and their syntax trees. An new format to define re-authoring rules is planned with RDF (Resource Description Framework) specification [16] of World Wide Web Consortium (W3C). RDF allows us to describe the metadata of web documents and interchange its semantics across them.

7. Concluding Remarks

This paper addressed the adaptive presentation of HTML documents, and discussed how they can be customized with the environmental context. The Persona system intermediates between users and documents and provides the environment independent presentation for them. It makes it easy to dynamically process and distribute documents. We also experimented applications using Persona. The information on our project can be found at [17].

Reference

[1] T. W. Bickmore. Digestor: "Device-independent access to the World Wide Web". In *Proceedings of the Sixth International World Wide Web Conference*, 1997.

[2] J. Suzuki and Y. Yamamoto. Document brokering with agents: Persona approach. In *Proceedings of the Sixth Workshop on Interactive Systems and Software*, December, 1998.

[3] C. Brooks, M. Mazer, S. Meeks, J. Miller. Application-Specific Proxy Servers as HTTP Stream Transducers. In *Proceedings of Fourth International World Wide Web Conference*, 1995.

[4] H. W. Lie et al. Cascading Style Sheets, level 1. W3C, 1996.

[5] J. Clark et.al. (ed.). Extensible Stylesheet Language (XSL) version 1.0. W3C Working Draft 18-August-1998.

[6] JavaWebServer and Servlet. at http://jserv.java.sun.com/products/webserver/docs/

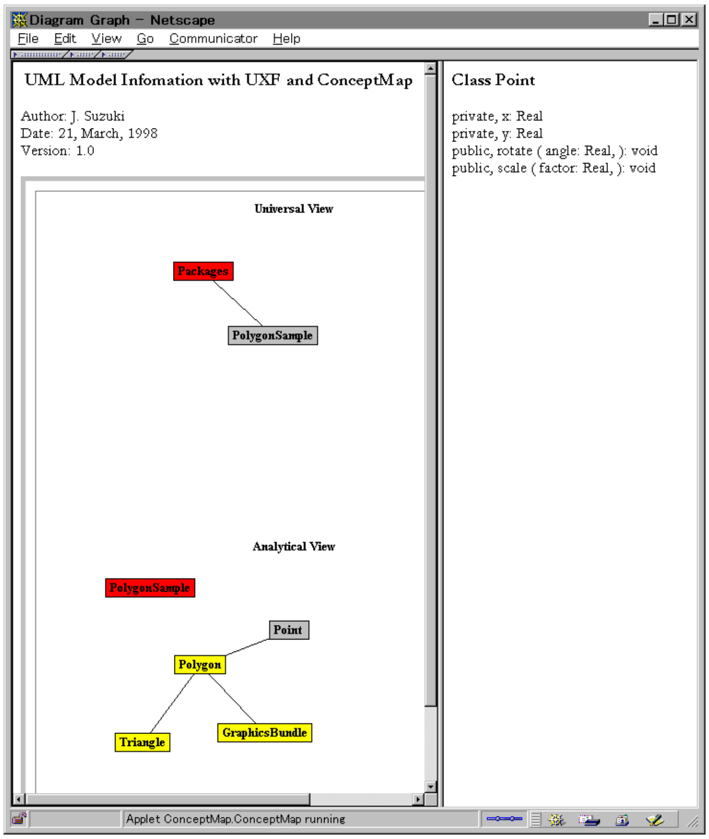

Figure 6: Sample re-authoring with the client-side text manipulation strategy.

[7] J. Suzuki and Y. Yamamoto. Building an Adaptive Web Server with a Meta-architecture: AISF approach. In *Proceedings of JSSST SPA'98*, March, 1998.

[8] J. Suzuki and Y. Yamamoto. OpenWebServer: an Adaptive Web Server using Software Patterns. Submitted to the special issue on design patterns of IEEE Communications magazine.

[9] J. Suzuki and Y. Yamamoto. Managing the software design documents with XML. In *Proceedings of ACM SIGDOC'98*, 1998.

[10] J. Suzuki and Y. Yamamoto. Toward the interoperable software design models: quartet of UML, XML, DOM and CORBA. Submitted to IEEE ISESS'99, 1999.

[11] J. Suzuki and Y. Yamamoto. Making UML models exchangeable: UXF approach. In *Proceedings of UML'98*, 1998.

[12] J. Suzuki. Making UML models interoperable with UXF. LNCS, Springer, 1999 (to be appeared).

[13] J. Suzuki. Design and coding of ConceptMap. Submitted to Java Report, SIGS publication.

[14] J. Suzuki and Y. Yamamoto. Metadata Management in Personalizing Web Presentations. Submitted to UM'99, 1999.

[15] Netscape Corporation. Persistent Client State: HTTP Cookies. at http://www.netscape.com/newsref/std/cookie_spec.html

[16] O. Lassila and R. R. Swick (eds.). Resource Description Framework (RDF) Model and Syntax Specification. W3C Working Draft 19 August 1998.

[17] Persona project web site: http://www.yy.cs.keio.ac.jp/~suzuki/project/persona/.