# An extensible architecture for web adaptability

José Paulo Leal and Pedro Silva

DCC-FC & LIACC, University of Porto
R. Campo Alegre, 1021/1055 – 4169-007 Porto, Portugal
zp@dcc.fc.up.pt psilva@alunos.dcc.fc.up.pt

**Abstract.** This paper presents an architecture for web adaptability based on service oriented principles. The major feature of this architecture is the loose coupling of its components, including the adapted web site and the adaptation components, that enables its use by existing web sites and provides a convenient test bed for new adaptation strategies. Another important feature is that adaptation is based on interaction data collected directly on the web browser rather than on web server's logs as in most frameworks.

**Keywords: adaptability framework, SOA, service orientation**

## 1 Introduction

Our research goal is the creation of an infrastructure for web adaptability that integrates with the development tools used by most sites. Moreover, this infrastructure is intended to be extensible, in the sense that it should not have a fixed set of adaptation strategies. In this paper we present an architecture for such an infrastructure based on service oriented principles.

In his seminal paper on web adaptability [7], Perkowitz distinguishes two forms of web site adaptation: the optimization of a site for the benefit of a group of users; and the customization of the site to adjust to preferences and needs of a specific user. Perkowitz defined optimization as an off-line process that improves the site as a whole for everyone, while customization is on-line process that improves a single page for a specific user. Using this classification our approach to adaptation should be classified as a customization.

Some frameworks for web customization, as most of those reviewed by Eirinaki and Varzirgiannis [6], integrate several aspects of web site management, including those necessary for managing a traditional, non-adaptable web site. That approach has two important drawbacks: it forces the web site to be developed within the adaptation framework and relies on a fixed set of adaptation features.

Content management, and sometimes web site publishing, are some the tasks that must be implemented within these frameworks to benefit from their adaptation service. These tasks are not specific to web adaptation and are more effectively implemented by specialized systems. In fact, most existing web site already use their own set of tools and would have to be re-implemented in order to be adapted by these frameworks.

Moreover, most adaptation frameworks are designed towards a fixed set of adaptation features. Since the integration of new types of customizations is difficult, these systems are not good test beds for new adaptation strategies and thus are not the ideal adaptation framework, at least from a research point of view. As production systems, this lack of flexibility is also a disadvantage since the adapted web sites cannot incorporate customizations developed by a third party.

These observations lead us to define the following set of requirements of an architecture for web customization:

- support existing web sites, independently of their platform;
- require no changes (or minimal changes) in the adapted web sites;
- delegate high level adaptation features in pluggable components;
- provide only the basic services required for customization;
- avoid reimplementing features available in other systems.

To support customization in existing websites, developed in different platforms, the several components that assemble the adaptation framework must have a loose coupling and rely on communication to achieve adaptation.

As well as web sites, the components responsible for adaptation should be pluggable parts in such an architecture. These components, that we call *adapters*, should be independent from web sites and web formatting. They should receive adaptation requests, have access to the web site content, to end users' usage data or other relevant content, and produce a customization response.

With this view of an architecture, where adaptable website and adapters are pluggable components, we identified three basic services that should be provided by the underlying framework:

1. connect web sites pages with adapters;
2. apply the customization to web sites pages;
3. collect usage data to feed adapters.

The rest of this paper describes as architecture meeting these requirements and is organized as follows. The next section introduces the main concepts of the proposed architecture. Section 3 describes the messages exchanged between components and the following sections analyzed each component type: client, broker and adapters. Finally, we present some open issues and future work.

## 2 An Architecture for adaptability

In this section we introduce the main concepts of an architecture for web adaptability focused on extensibility. This architecture proposes a framework with two types of extensions points: adaptation consumers and adaptation providers. Adaptation consumers are any kind of web sites, including static HTML files, dynamic content generated by web applications or content management systems. Adaptation providers, or simply *adapters* are pluggable components that implement a particular strategy for web adaptability.

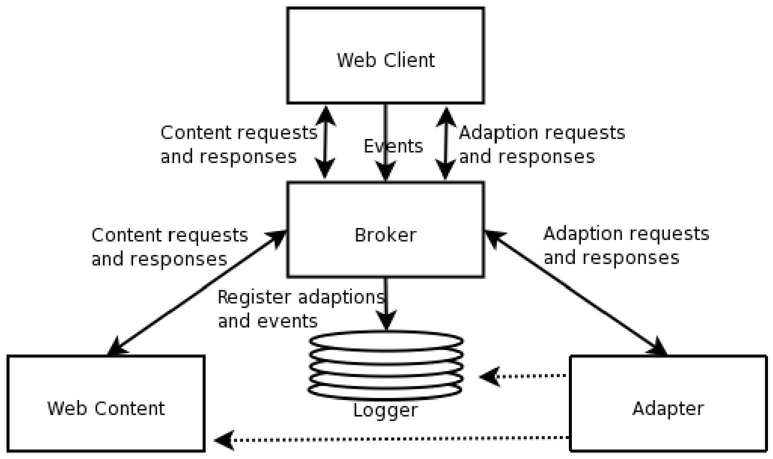
Web Client
Content requests and responses
Events
Adaption requests and responses
Broker
Content requests and responses
Adaption requests and responses
Register adaptions and events
Web Content
Logger
Adapter

**Fig. 1.** An architecture for web adaptability

Every type of web site, ranging from a simple collection of static HTML pages to a n-layered web application, relies on a web client (browser). For that reason, the web client is the ideal place to apply customization, directly over web content formatting. The browser is also the ideal place for collecting usage data based directly on user's interaction, rather than on web servers logs.

This framework places itself between the web client and web server as shown in Figure 1. A central broker mediates the communication between web clients, web server and the adapters. Page requests are simply forwarded to the HTTP server while adaptation requests are forwarded to adapters. When the user follows a link this results in a page request. The adapted page is loaded to the browser and a component is automatically activated that performs the adaptation of the client side. The client side adaptation is based on adaptation requests that are channeled throw the broker to one or more adapters connected to the infrastructure. The client component is also responsible for collecting user interaction data and reporting it to the broker. All communication with the client – adaption and notification messages – is recorded in a logger component and made available to adapters. Using data from the logger (and possibly from content) the adapters can improve their response to adaptation requests.

This architecture is inspired in Service Oriented Architectures (SOA) and follows some of its main guidelines [4]:

1. component interaction is based on implementation independent messages;
2. there is an prearranged agreement on communication content;

3. components are autonomous, i.e each component is a fully working piece of software;
4. components hide implementation details and the framework focus on communicated data.

Communication between heterogeneous components is the corner stone of this architecture. Thus, in the next session we start by introducing the messages exchanged between components before detailing the role of the different component types.

## 3 Messages

To communicate components exchange messages in an XML [2] dialect that acts as an adaptation language. This language was designed to encode all data communication in this framework, with emphasis on page adaptation, but including also messages for notification of user activity and broker administration.

Adaptation messages are used for requesting the customization of a web page and returning the changes computed by an adapter. These messages identify the parts of the web page to be customized, the adapter type needed to perform the customization, and what is the customization – how the page can be rearranged, what new content can be inserted – while avoiding the communication of formatted content (e.g. XHTML).

The language supports two kinds of actions as customization: recommend, when new content is inserted; reformat, when content is preserved but formatting is changed. For instance, if the user searches a product and other related products are displayed then it is a recommendation. If the list of products is altered, highlighting or promoting some of them, or even hiding other, then it is reformatting.

A web page requiring adaptation may have several customization points. Each *customization point* is given an unique XML identifier and is associated with a collection of items. In recommendations, responses return a list of items to be inserted in the customization point. If the customization action is reformat then a list of items is sent in the adaptation requests. In both cases the adaptation response associates to each item a set of attributes that influences formatting of the client side.

In these messages a type of adapter is encode as an Uniform Resource Name (URN) [8], which is a kind of Uniform Resource Identifier, just as the popular URL used for referring web pages, that does not require the availability of the resource that is identified.

After adaptation, notification is the most important message type. These messages are issued by client components to report user activity, they are processed by the broker and stored in logger. This information will later be used by adapters. A notification message may report on several events to reduce the communication with the broker.

In the current version of the language, notification messages cannot categorize event types. All events types supported by the implementation language of the client component (such as JavaScript) can be sent in notification messages. To further reduce the number of notification messages we may consider in the future to process events on the client side. Notification messages would report high level events, more adjusted to the needs of adapters. A radical approach would be to report only on user activity, indicating the number of consecutive seconds in which the user interacted with the page and the number of seconds the web interface was idle. In any event, this is still a topic of future research.

The messages' language includes also administration messages. This type of message is used for tasks such as registering adapters in the framework, to enable or disable registered adapters, to list adapters or query their status.

## 4 Broker and Logger

The broker has a central role in this architecture. It is responsible for registering other components, managing communications and logging data to support adaptation. Although close to the broker, the logger must be an autonomous component since it can be queried by adapters without interfering in the brokers performance and should be deployable in a separate server, if necessary.

Efficiency is a very important aspect of the broker since it processes all requests coming from clients. It should be noted that, for each web page request, several requests will hit the broker: one web page request, introducing modifications on-the-fly to connect it to adapters, one adaptation request for each adapter involved (each page can use more than one), and several notification requests, depending on the user interaction on that page.

On the initial request of a web page the broker will fetch it from the web content server and apply a transformation on the HTML code. This transformation replaces each customization point with a call to a specific adapter, as defined in the broker's configuration for that web site. This configuration file relates an URL pattern, a customization point and an adapter. For efficiency sake, the transformed web pages can be cached by the broker.

To process adaption requests the broker must parse the messages described in the previous subsection. Adaptation requests are simply forwarded to a registered adapter with a compatible type. The response produced by the adapter is sent back to client. In the end of this process, the adaptation is recorded in the logger. Notification messages report on events that occurred in web interface. The broker unmarshals these messages and records their data in the logger. Administration messages are an application programming interface (API) exposing the commands of the broker. Using this API adapters may interact with the broker. This API may also be used to implement custom administration interfaces for managing the framework.

Using the administration commands adapters can register themselves in the broker, so that they can latter be invoked when a request is made for their type of adaptation. On registration adapters specify their IP addresses and their type. Several adapters with the same type may be registered in the same broker. In this case the broker will balance load among the adapters with the same type

## 5 Client component

In this architecture customization is performed preferentially on the web browser. The client component has a very important role in the process since it is responsible for requesting adaptation to the broker, modifying the web content and reporting user activity for logging, and subsequent processing by adapters. Other adaptation frameworks collect also usage data of the client [9] but as far as we know no other performs adaptation on the client.

As a design goal, we tried to keep this architecture as independent from specific technologies as possible. Nevertheless, the implementation language of client component must be able to communicate with the broker, independently from the HTTP request-response cycle, and modify the interface according to adaptation responses.

Several web technologies meet these requirements. For instance, Java applets or Flash movies, have the necessary features to implement the client component, although the vast majority of web pages uses only standard W3C technologies - such as HTML, CSS and ECMScript - for presentation and formatting. A client component for this kind of web interface can be implemented using a technique commonly called Ajax – standing for *Asynchronous JavaScript and XML* – that mixes communication of XML data outside the normal request-response cycle of HTTP, with the manipulation of the page structure and content using the DOM API. Although this was the chosen approach to implement the framework, it should be noted that this choice is independent from the architecture.

## 6 Adapters

Adapters are pluggable components that process adaptation requests and provide a response that is delivered to the client component. This architecture specifies only the role of adapters, not their design. An implementation of this architecture must specify the interfaces provided by adapters but should not place other constraints on programming languages, development platforms or other implementation tools. Adapters may run on different hosts and be implemented if several different platforms. Some adapters may be small programs experimenting a new algorithm while other may be full fledged components optimized for performance.

An implementation of this architecture should provide only a small library of simple adapters, including the null adapter and echo adapter. The null adapter is useful for registering events from a web page without actually performing any adaptation. The echo adapter will simply echo the message it receives. Both these adapters can used for debugging purposes and should be implemented as independent components. Although the broker could easily implement internally their functionality, implementing them as external components reduces the complexity of the broker and makes them more effective as debugging tools.

Adapters process adaptation requests based on user interaction data provided by the logger. They can also make use of web site content, or other external data (for instance, news feeds) but that is outside of the scope of this architecture.

As a rule, adapters do not deal with web formatting languages, such as HTML, neither in requests, nor in responses. Requests for adaptation refer to customization points on web pages but do not include the actual page; and responses from adapters are sets of modifications to a page, not the modified content. This separation between content and adaptation ensures that an adapter can be re-used in different web sites.

## 7 Concluding remarks

This paper describes an architecture for a web adaptation framework, designed to connect adaptation consumers – web sites – with adaptation providers – adapters – while feeding them with detailed usage data in which to base customization. Unlike other systems for web adaptability, this framework avoids implementing specific adaptation features that are delegated to adapters. This non-intrusive approach makes this framework applicable to existing web sites and does not prevent the integration of third party adaptations.

The major contributions of this work are an architecture based on service oriented concepts and particularly an adaptation language for exchanging messages between the several types of components.

A prototype of framework following this architecture was implemented and applied to an existing web site for testing these concepts. Since then the framework entered in production has been used as a test bed for complex adapters linked to a data warehouse [3] and is scheduled to test adaptation experiments in commercial sites.

There are some open issues in this work that we expect to address in the near future, including the following.

– In the current implementation of the framework the broker is just processing messages issued from the client component, and is not processing request for web content; the reverse proxy module of the Apache HTTP server is being used for that purpose. In the current implementation the transformation of web pages must be done "by hand" on the web content. We are currently working on the broker's transformation module.
– The interaction data collected in the web client could be processed locally before being logged, saving that effort to the adapters and reducing the communication with the broker. We hope to be able to define high level events, adjusted to the needs of web adaptation, and define these events types in notification messages.
– The current implementation of the framework has a single implementation of the client library in JavaScript, targeted for traditional HTML with CSS sites. It would be interesting to test this approach with other technologies.

## 8 Acknowledgments

Work partially funded by *Fundação para a Ciência e Tecnologia* (FCT) and *Programa POSC*, under project Site-O-Matic (contract POSC/EIA/58367/2004) and by LIACC through *Programa de Financiamento Plurianual*, FCT and *Programa POCTI*, co-financed by EC fund FEDER.

## References

1. Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin. Namespaces in xml 1.0 (second edition). Technical report, World Wide Web Consortium, August 2006.
2. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml) 1.0 (fourth edition). Technical report, World Wide Web Consortium, 2006. Available as `http://www.w3.org/TR/xml`.
3. Marcos A. Domingues, Alípio M. Jorge, Carlos Soares, José Paulo Leal, and Pedro Machado. A data warehouse for web intelligence. In *Proceedings of the Portuguese Conference on Artificial Inteligence*, 2007. *Accepted for publication.*
4. Thomas Earl. *Service-Oriented Architecture - Concepts, Technology and Design.* Prentice Hall, 2005.
5. David C. Fallside and Priscilla Walmsley. Xml schema part 0: Primer second edition. Technical report, World Wide Web Consortium, 2004. Available as `http://www.w3.org/TR/xmlschema-0/`.
6. M. Vazirgiannis M. Eirinaki. Web mining for web personalization. *ACM Transactions on Internet Technology*, 3(1):1–27, February 2003.
7. Mike Perkowitz and Oren Etzioni. Adaptive web site: an ai challenge. In *Proceedings of the IEEE*, 2004.
8. K. Sollins and L. Masinter. Rfc 1737: Functional requirements for uniform resource names. Technical report, The Internet Engineering Task Force, December 1994. available as `http://tools.ietf.org/html/rfc1737`.
9. Michal Tvarozek, Michal Barla, and Mária Bieloková. Personalized presentation in web-based information systems. In *SOFSEM 2007*, number 4362 in LNCS, pages 796–807. Springer, January 2007.