Adaptabilidade Web no SOM

Pedro Silva e José Paulo Leal

DCC-FC & LIACC, Universidade do Porto, Portugal
psilva@dcc.fc.up.pt
zp@ncc.up.pt
WWW: http://www.niaad.liacc.up.pt/Site-O-Matic

Resumo Neste artigo é descrita uma metodologia desenvolvida no âmbito do projecto Site-O-Matic (SOM) para introdução de adaptabilidade em sítios da internet. Esta baseia-se numa *framework*, com uma arquitectura orientada a serviços, constituída por um conjunto de módulos heterogéneos e que trocam entre si mensagens XML através de um *broker*.
A troca de mensagens recorre às tecnologias *Web Services* e XMLHttpRequest. Esta última é usada na comunicação com navegadores, sendo os *Web Services* usados para os restantes módulos. As mensagens trocadas representam uma linguagem de adaptabilidade onde se encontram os dados necessários aos diferentes passos da adaptação.
Neste artigo são focados diferentes aspectos da utilização das tecnologias XML na implementação da *framework*, incluindo: o desenho da linguagem de adaptação; a utilização de Ajax para construção e envio de mensagens e a sua conversão em HTML; a implementação de serviços *web* de manipulação de mensagens; a parametrização de módulos de adaptação.
É também apresentado um exemplo de aplicação da metodologia proposta e é avaliado o impacto da sua utilização.

1 Introdução

O Site-O-Matic[7] é um projecto de investigação do grupo de Inteligência Artificial e Análise de Dados[8] (NIAAD) que tem como objectivo, o desenvolvimento de uma plataforma e uma metodologia para automatizar actividades relacionadas com a gestão de sítios. Exemplos dessas actividades, são a gestão do conteúdo, a sua estruturação, recomendação e personalização. O Site-O-Matic propõe-se ainda a ter em conta o comportamento dos utilizadores e os objectivos do sítio na aquisição dos seus próprios objectivos.
O Site-O-Matic debruça-se sobre três aspectos:

- Definir uma plataforma flexível para sitíos *web* que permita a aquisição de dados de qualidade, assim como transformações e adaptações *online* da estrutura e interface dos sítios;
- Desenvolver técnicas para realizar adaptações *web* usando *data mining*;
- Obter conteúdos focados em tópicos, sendo essa obtenção guiada por objectivos bem definidos e monitorização de valores de desempenho sobre os dados da utilização.

Para atingir estes objectivos este projecto dividiu-se em várias tarefas, uma das quais deu origem a este trabalho com o objectivo de definir uma metodologia e uma infra-estrutura que facilite a introdução de adaptabilidade em sítios *web*. A infra-estrutura deve permitir a modificação automática de páginas *web* usando mecanismos de adaptação que funcionem sobre dados de utilização. Deve também recolher esses dados necessários à adaptação.

1.1 Adaptação de sítios *web*

Para realizar adaptações sobre páginas *web*, qualquer solução tem de reunir a capacidade de fazer um conjunto de tarefas[1] necessárias à adaptação, das quais a nossa abordagem destaca:

- Recolha de dados de adaptação;
- Integração de mecanismos de adaptação;
- Aplicação de transformações.

A adaptação em função do utilizador requer algum conhecimento sobre as suas preferências, de modo a criar páginas com conteúdo relevante para o utilizador. Para isso, é necessário recolher dados para adaptação, que dêem a conhecer algo sobre o utilizador e que possam ser levados em conta na adaptação. Os dados sobre o utilizador têm um papel preponderante para os processos de adaptação. Os mecanismos que produzem adaptações usam-nos para moldar os seus resultados, conseguindo assim um maior refinamento e uma maior orientação ao utilizador. Depois de obtidos os resultados é necessário transformá-los em visualizações, por forma a reflectirem-se de alguma maneira na página que o utilizador está a consultar. Outras tarefas propostas[1] estão relacionadas com a utilização directa de um gestor de conteúdos.
Independentemente destas tarefas foi possível identificar com este trabalho duas visões de adaptação[2]:

Recomendação Adaptações em que são inseridos novos conteúdos.
Reformatação Adaptações em que o conteúdo não é alterado mas reorganizado, seja do ponto de vista da sua ordem relativa ou da formatação.

Em ambos os casos o objectivo é melhorar a experiência do utilizador.

1.2 Recolha de dados de adaptação

Para adaptar páginas *web* é necessário obter informações sobre a pessoa a quem se destina a adaptação.
Tradicionalmente os dados usados para atingir estes objectivos são os que se encontram nos ficheiros de registos dos servidores *web* e nos sistemas de informação[1][4]. Usando estes dados relacionam-se por exemplo página vistas, artigos comprados e dados pessoais. Contudo, estes mecanismos apresentam algumas limitações[1][4]. Por exemplo, não é possível saber se um utilizador que acedeu a uma determinada página a leu ou não. O utilizador pode de facto ter

acedido à página mas nunca sabemos se se ausentou, se leu todo o seu conteúdo ou apenas uma parte. Naturalmente este vazio de informação pode levar a interpretações erradas aquando da geração de conteúdos adaptados. Se conseguirmos perceber determinados comportamentos do utilizador podemos aumentar a qualidade da informação coleccionada sobre este. Aquilo que propomos é acrescentar aos dados tradicionais, que estão do lado do servidor *web* e dos servidores de informação, dados que estão disponíveis junto do utilizador através do navegador, nomeadamente eventos.
Ao processarmos eventos de rato como *click*, *focus*, *scroll* ou *select*, podemos inferir algum contexto da navegação do utilizador. Se um utilizador, ao ler um texto, seleccionar uma parte desse texto, podemos traduzir esse evento como interesse nesse texto. Do mesmo modo, se o utilizador deslocou a página (*scroll*) para conseguir ver uma parte que não estava visível, podemos interpretar esse movimento como interesse sobre a zona oculta da página. Podemos também criar **meta-eventos** para agregar um conjunto de eventos com outras variáveis como o tempo, para interpretar interesse ou desinteresse. Por exemplo, se durante um determinado espaço de tempo um utilizador não gerou eventos ou mudou várias vezes de página podemos considerar que houve um desinteresse sobre o que estava a ser apresentado. O mesmo se podendo dizer no caso inverso. Se um utilizador gerar uma série de eventos numa determinada página durante um período alargado de tempo podemos interpretá-los como interesse no conteúdo apresentado.

2 Arquitectura da *Framework*

As diferentes soluções existentes[1] apresentam abordagens monolíticas, e mesmo algumas que se dizem ser uma *framework* estão orientadas a um determinado gestor de conteúdos. A proposta que apresentamos é uma *framework* que apresenta uma arquitectura com características de uma arquitectura orientada a serviços[5] (SOA). Os dois componentes que compõem a *framework* propriamente dita são: o **Cliente** e ***Broker***.
O Cliente é constituído por uma biblioteca JavaScript[6] responsável pelas operações a realizar no navegador. O *Broker* é um componente central único, responsável pelos processos de comunicação e registo de dados. Esta solução não define como componente os geradores de adaptações ou **Adaptadores**, pois entende-se que estes devem ser deixados ao critério dos criadores de adaptações. A única imposição é que cada adaptador implemente um serviço *web* segundo uma dada especificação WSDL[5], podendo assim receber, processar e enviar mensagens XML.
Como se trata de uma arquitectura de inspiração SOA uma das características da *framework* é a utilização de mensagens na comunicação entre os diversos componentes. São geradas mensagens nos Clientes com a informação de utilização que são enviadas aos Adaptadores e são geradas mensagens nos Adaptadores com os resultados das adaptações que são enviadas para os Clientes.
Independentemente dos meios usados para a circulação destas mensagens, elas

cumprem um conjunto de regras estabelecidas *a priori* e que definem uma linguagem de adaptação. Esta definição estabelece uma linguagem que representa a informação necessária à adaptação, abstraindo essa definição da implementação.

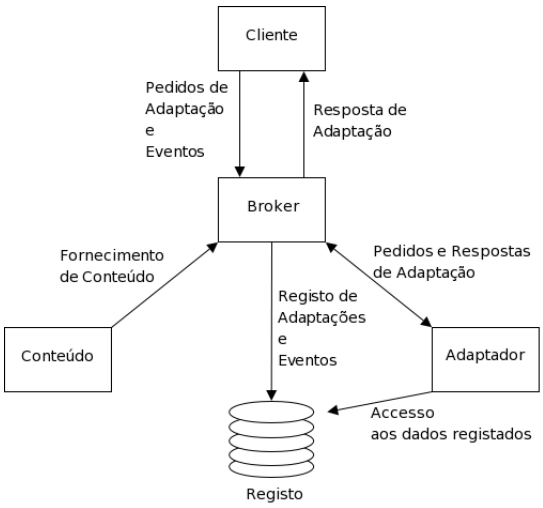

Figura 1. Proposta para arquitectura geral.

2.1 Cliente

O módulo Cliente tem duas funções principais: realizar operações sobre mensagens e realizar a reformatação e transformação de conteúdos. As operações sobre as mensagens incluem: a criação de mensagens que representam os dados a serem adaptados no Adaptador e enviadas através do *Broker*; a interpretação de mensagens que chegam do *Broker* provenientes dos Adaptadores, com os resultados da adaptação; e a criação de mensagens de notificação com os eventos gerados pelo utilizador. As operações de reformatação ou transformação são as operações sobre o conteúdo que visam reflectir as adaptações realizadas. Nas operações sobre as mensagens o Cliente deve ter a capacidade de construir e processar as mensagens mas também de as enviar e receber, se possível sem prejudicar a navegação do utilizador. No que diz respeito à manipulação do conteúdo o Cliente deve ter a capacidade de o manipular, se possível de forma simples e sem introduzir efeitos secundários. O Cliente é definido como **consumidor de adaptação**. Define uma API que permite invocar o processo de adaptação através do método `adapt(adaptor_id,adaption_parameters,element_id)`. Ao incluir este método no código HTML é iniciado o processo de adaptação. Os parâmetros deste

método indicam: a identificação do adaptador; eventuais parâmetros a passar ao adaptador; e o elemento da página sobre o qual será feita a adaptação.

2.2 *Broker*

O *Broker* é o ponto central da *framework*. Por ele passam todas as comunicações, tendo uma função assumidamente de *proxy* e *logger*. Mensagens de adaptação e notificação partem do Cliente, são analisadas e seguem os seus respectivos caminhos. As competências do *Broker* foram definidas da forma mais simples possível para não comprometerem a sua escalabilidade. Sendo o ponto central de toda a *framework* e processando todas as mensagens, é importante que o tempo gasto com cada uma seja o mais reduzido possível.
Num ambiente de produção, numa arquitectura distribuída como esta, irão existir n clientes e n adaptadores que irão contribuir para o escalar de mensagens no sistema. O *Broker* terá de ser capaz de lidar com diversas ligações, registos em base de dados e algum processamento de mensagens de forma eficiente para não prejudicar o sistema.

2.3 Adaptador

O Adaptador não é um componente da *framework*, mas é um componente fundamental no processo de adaptação. É ele que possui a capacidade de gerar adaptações com os seus algoritmos específicos. Por outro lado tem de ter a capacidade de comunicar usando a mesma linguagem que os restantes componentes da *framework* esperam e acordaram inicialmente. Cada adaptador tem definido através de uma especificação WSDL o seu interface público, o seu endereço e o tipo de comunicação a usar.
Os Adaptadores são definidos como **fornecedores de adaptação**.

3 Linguagem de adaptação

A linguagem de adaptação do SOM permite abstrair os dados de adaptação, passando estes a estar contidos exclusivamente no conteúdo das mensagens. Na descrição da *framework* definiu-se como objectivo de concepção que toda a informação trocada entre os vários componentes fosse explicitada nas mensagens. Por exemplo, evitou-se que alguma informação fosse codificada nas mensagens HTTP, nomeadamente em *cookies*, normalmente usadas pela maoiria das soluções existentes[1][4].
Devido ao uso do XML as mensagens são facilmente legíveis por humanos permitindo durante o processo de desenvolvimento inspeccionar as informações duma determinada comunicação, com vantagens evidentes durante o *debugging*.

3.1 Estrutura das Mensagens

Como já referimos, as mensagens que circulam na *framework* têm um tipo predefinido. Essa definição em XMLSchema é também usada pelo *Broker* para validar

as mensagens em circulação. Deste modo é garantida a validade das mensagens em circulação na *framework*. Isto serve para evitar, por exemplo, mensagens destinadas a corromper ou utilizar o sistema com outros fins que não o da adaptação. As figuras seguintes são exemplos de alguns elementos que compõem as mensagens. Existe um elemento de topo `message`, que contém como informação adici-

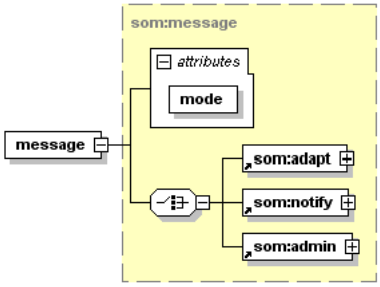

Figura 2. Elemento `message`.

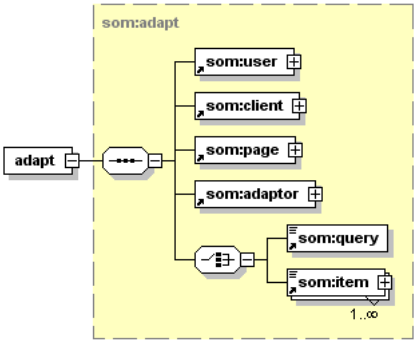

Figura 3. Elemento `adapt`.

onal de contexto um atributo `mode` indicando o tipo de mensagem (`request` ou `response`). Este elemento pode albergar três tipos de elementos e que correspondem no fundo aos três tipos de mensagens possíveis:

`adapt` Mensagens de adaptação que circulam entre os componentes com os respectivos pedidos e respostas de adptação.

`notify` Mensagens de notificação que são enviadas pelo Cliente para efeitos de recolha de dados.

`admin` Mensagens de administração que são usadas para diversas tarefas de manutenção e configuração do sistema.

Em todos os casos há elementos comuns contidos nas três definições, como por exemplo informação sobre o utilizador ou a página. Apenas diferem em alguns elementos que são significativos para o seu tipo. É o caso das mensagens de adaptação que contêm informação sobre o adaptador que deve ser usado, ou o caso das de notificação que contêm conteúdos específicos, resultado de eventos processados junto do utilizador.

4 Comunicação e Processamento

Para utilizar mensagens XML na adaptação foi necessário encontrar meios para a sua construção e comunicação. Para o Cliente a solução passou pela utilização de **Ajax**[9] e para a comunicação entre o *Broker* e os Adaptadores a escolha recaiu sobre os ***Web Services***[5]. Em particular esta escolha deveu-se às necessidades de cada módulo consoante a sua localização.

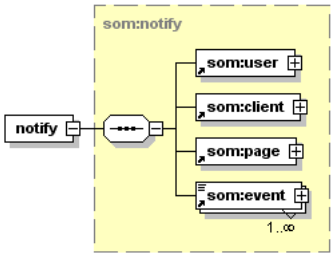

Figura 4. Elemento `notify`.

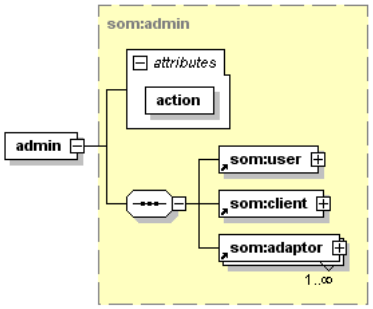

Figura 5. Elemento `admin`.

Como um dos objectivos era introduzir adaptabilidade em sítios já existentes, no caso do cliente a solução encontrada passou pela introdução de uma biblioteca JavaScript no código HTML da página. A biblioteca define uma função que é accionada em cada carregamento de página e despoleta todo o processo de adaptação. Desta forma é possível ligar e desligar facilmente a adaptação do lado do cliente e de modo selectivo.
No caso dos Adaptadores a solução dos *web services* foi escolhida por permitir independência de tecnologia. Usando *web services* não se impõem nenhuma obrigatoriedade sobre a a construção dos Adaptadores. Estes podem ser construídos em qualquer linguagem com capacidade de utilização de um motor SOAP, algo que muitas das linguagens hoje existentes já suportam. Ambas as soluções permitem manter nas mensagens a definição e especificação da adaptação, tornando a camada de implementação completamente independente.

4.1 Ajax

O Ajax - Assynchronous JavaScript and XML, é uma designação comum para um conjunto de tecnologias que permitem a realização de pedidos HTTP e alterações na estrutura e conteúdos de páginas HTML sem necessidade de as redesenhar na totalidade. Para realizar os pedidos HTTP o Ajax usa um objecto JavaScript, o XMLHttpRequest, que permite realizar pedidos de forma síncrona ou assíncrona. Os pedidos assíncronos tem a particularidade de permitir que o processamento não seja bloqueado pelo aguardar da resposta. Isto faz com que uma página possa ser utilizada e simultaneamente realize pedidos HTTP, sem que o utilizador se aperceba. Com este método, fica facilitada a troca de informações que sirvam de complemento a um sítio. A utilização do Ajax traz ainda algumas vantagens adicionais, em particular a diminuição da largura de faixa usada. Como os pedidos feitos com recurso ao Ajax transportam apenas a informação necessária contribuem para um descongestionamento das vias de comunicação e dos servidores.
O Ajax é usado na *framework* em três tarefas:

- Construção de mensagens de adaptação;

- Comunicação assíncrona de dados;
- Reformatação da página a adaptar.

A construção de mensagens de adaptação é feita de duas maneiras: compilando dados existentes na página corrente e através do processamento de eventos. Os eventos permitem recolher informações que apenas se encontram do lado do navegador, mais concretamente permitem obter dados sobre a utilização real das páginas por parte de um determinado utilizador.
A construção de mensagens de adaptação assim como a modelação da página, depois de obtida a resposta de adaptação, recorrem à DOM para aceder e navegar no seu conteúdo. Usando os objectos que a DOM disponibiliza, a biblioteca de JavaScript consegue ler e escrever directamente sobre a página visualizada. Isto permite acelerar o processo de actualização da página visto que se processa directamente sobre o objecto em memória, tornando as alterações instantâneas. Toda a comunicação de envio e recepção é feita assincronamente permitindo que o utilizador não seja afectado enquanto decorrem este processos. O utilizador tem sempre acesso à página pedida independentemente da demora do processamento de adaptação.
De salientar que a utilização de Ajax ficou a dever-se ao facto de o JavaScript ser uma linguagem largamente aceite pela maioria dos *browsers*. No entanto outras linguagem podem ser usadas desde que tenham capacidade de processar XML e comunicar assincronamente, como por exemplo o Flash.

4.2 *Web Services*

Os *web services* permitem o estabelecimento de ligações ponto a ponto de forma transparente e autónoma. Recorrendo ao protocolo SOAP os *web services* criam canais de comunicação por onde circulam mensagens XML. O facto de os *web services* apresentarem características que se encaixam no modelo SOA influenciou a nossa escolha. A existência de um interface público que permite dar a conhecer o serviço sem que seja conhecida a sua implementação permite que os diferentes componentes possam trocar mensagens facilmente. No caso da solução proposta, a utilização do motor SOAP Axis[10], permitiu usar um método de comunicação que facilita a integração de documentos XML nos envelopes SOAP. O Axis suporta quatro métodos de comunicação : *RPC*, *Document*, *Wrapped* e *Message*. Os serviços `RPC` destinam-se à invocação de métodos mapeando os elementos do XML com os argumentos do método. Os serviços `Document` e `Wrapped` fazem mapeamento de objectos entre o Java e o XML e são também orientados à invocação de métodos. Por fim, os serviços do tipo `Message` têm como característica deixarem para a implementação do próprio serviço o processamento do corpo das mensagens SOAP, permitindo que estes manipulem directamente o XML das mensagens geradas.

4.3 Interface Message

Apesar de implementados em linguagens diferentes e usarem protocolos diferentes de comunicação Cliente, *Broker* e Adaptador comunicam segundo a mesma

linguagem, a linguagem de adaptação. Esta linguagem é comum a todos os componentes e como tal as necessidades para o seu processamento e manipulação são também comuns. Isto permitiu definir um interface para as mensagens. O

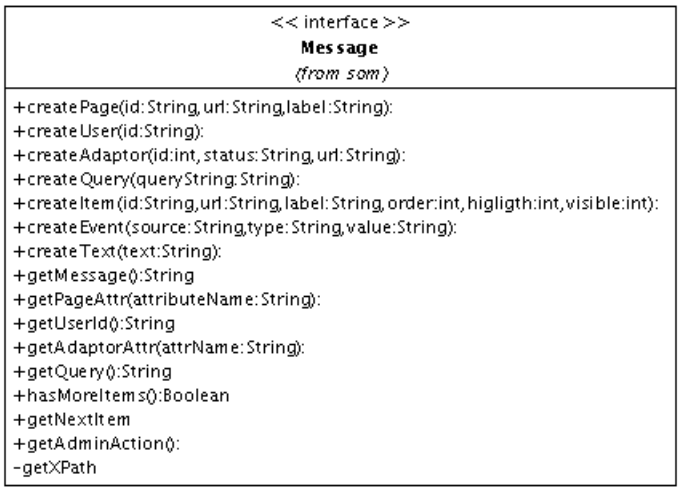

Figura 6. Diagrama de classes do interface `Message`.

interface `Message` apresentado na figura 6 é o esqueleto para qualquer implementação que necessite de lidar com as mensagens que circulam nos componentes. Esse interface foi criado com o intuito de homogeneizar processos e acelerar a implementação de funções que são comuns aos diferentes componentes. Há no entanto que salientar que este interface é implementado em diferentes linguagens, podendo, em algumas, não ser possível codificá-lo usando os recursos da linguagem. Este é o caso do módulo Cliente implementado em JavaScript. Apesar disso, a implementação em JavaScript e em Java seguiu os mesmos padrões, mantendo nomes de métodos e o modo como são usados, o que nos permite dizer no fim que ambos os módulos partilham um mesmo interface.
A definição deste interface, independente das linguagens de implementação, simplifica a construção e leitura de mensagens de adaptação nos fornecedores e consumidores de adaptação.

4.4 Adaptadores

Os adaptadores são um ponto de extensão da *framework* que através do WSDL ficam disponíveis para realizar adaptações. Este é caso dos adaptadores simples já incluídos na *framework* para efeitos de testes. No entanto, a *framework* inclui um **Meta-Adaptador** para permitir a inclusão rápida de programas externos

em linguagens de *scripting*. Esta capacidade tem como objectivo permitir aos investigadores, integrar rapidamente o seu código com a *framework*, escrito na sua linguagem de *scripting* favorita. Para conseguir isto definiu-se uma configuração em XML que indica ao meta-adaptador como interagir com o programa externo. A linguagem define um conjunto de regras que permitem através dos canais de `output` e `input` interagir com o programa externo.
As instruções mapeiam elementos do XML, como por exemplo `commandLine`, `queryCommand` ou `getItemProperty` respectivamente com a linha de comandos a usar para lançar o programa, o método que permite executar uma *query* ou o método que permite obter uma determinada propriedade. Foi este método que foi usado para construir os adaptadores de teste usados na aplicação desenvolvida para realizar a validação e avaliação da solução.
A utilização deste meta-adaptador é opcional podendo, se assim se desejar, desenvolver um adaptador que implemente um *web service* ficando directamente disponível para o *Broker*. Aliás, em ambientes de produção é de esperar que os Adaptadores tenham implementações mais robustas e de maior desempenho.

5 Avaliação

Como prova de conceito foi criada uma aplicação *web* que faz uso desta *framework*. Com esta aplicação foi possível observar o impacto da solução proposta e avaliar a aplicabilidade em ambientes de produção.
A aplicação criada foi um leitor *web* de manuais[1]. Ao aceder à aplicação um utilizador depara-se com uma página dividida em três zonas: uma zona de topo com um campo de pesquisa para indicar o tópico a consultar; uma zona central onde é visualizada a informação pedida; e uma zona lateral onde são apresentados conteúdos adaptados ao utilizador. Quando um utilizador acede ao sistema e realiza um pedido de uma página, um pedido de adaptação é gerado e é enviado assincronamente para o *Broker*. De notar, que neste ponto o utilizador pode já estar a ver a página pedida. Chegado ao *Broker*, para além de registar numa base de dados o pedido, a mensagem gerada no cliente é enviada ao Adaptador, por forma que este a possa processar. Quando o Adaptador termina as suas tarefas constrói uma mensagem de resposta com o conteúdo adaptado. No caso da aplicação criada trata-se apenas da compilação das últimas páginas vistas, ou seja um histórico. Essa mensagem é então encaminhada ao *Broker* e por sua vez ao Cliente que irá tratar de processar a mensagem e realizar as alterações necessárias à página. Neste caso estas alterações baseiam-se na construção de um menu lateral com a indicação do histórico.
Paralelamente a estas actividades a monitorização de eventos regista um conjunto de dados pré-definidos: *click*, *scroll*, ou *focus* que vão sendo registadas na base de dados. A aplicação é simples mas permite observar o funcionamento de todos os componentes e avaliá-los.

[1] O leitor de manuais foi testado com os manuais do Unix (*man pages*).

5.1 Resultados e observações

A avaliação da solução foi feita recorrendo a ferramentas de teste unitários e medições de tempos de resposta e processamento. Nas várias medições feitas registou-se um comportamento aceitável dos vários componentes mesmo em situações de utilização intensiva, isto é, com vários adaptadores e utilizadores. Observando o gráfico da figura 7 podemos ver que à medida que aumenta o

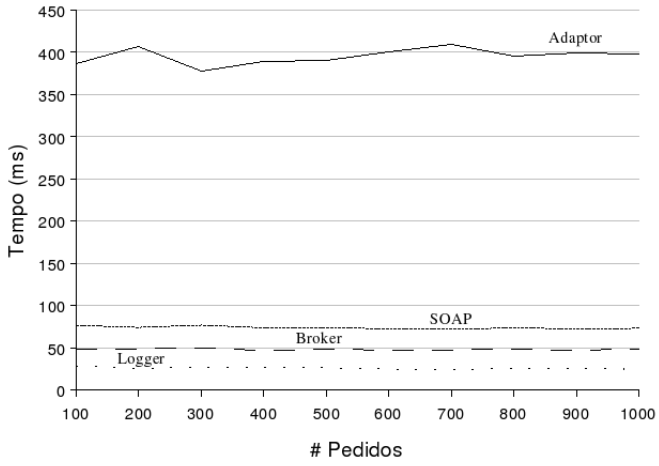

Figura 7. Tempos médios em ms, para 3 clientes e uma configuração de página com 2 adaptadores, com o número de pedidos a variar entre 100 e 1000.

número de pedidos o comportamento do *Broker*, *Logger* e da comunicação se mantêm quase constantes. Apenas o Adaptador apresenta valores elevados em virtude do mecanismo usado para interagir com o programa externo. A implementação na linguagem de *scripting* e o recurso aos canais de *input* e *output* limitam o seu desempenho. Para além deste teste foram também realizados testes de desempenho onde foram medidos os tempos para uma utilização com uma determinada sequência de pedidos. Os valores obtidos para esse teste assemelham-se aos do gráfico da figura 7.

6 Conclusões e trabalho futuro

Neste trabalho foi descrita uma metodologia para a introdução de adaptabilidade em sítios *web*. A solução apresentada cumpre os objectivos propostos através da definição desta *framework* e recorrendo a tecnologias como o Ajax, os *Web Services* e o XML. Esta solução permitiu-nos ter:

– Independência das tecnologias *web*;

– Abstracção das operações de adaptação;
– Expansibilidade dos módulos de adaptação.

Os resultados obtidos nos testes mostram que os benefícios da adaptabilidade não diminuem a qualidade da interacção para o utilizador final. Acresce ainda o facto de se ter conseguido introduzir um novo ponto de recolha de dados que vem trazer novas capacidades aos adaptadores de conteúdo.
Como se trata de um protótipo há ainda algum pontos a tratar no futuro entre os quais estão:

– Utilizar métodos e bibliotecas compatíveis em diversos *browsers*, para permitir uma fácil integração da biblioteca JavaScript fornecida com a *framework*;
– Permitir a utilização de modelos para especificar o modo como os dados de e para adaptação devem ser tratados no cliente, mantendo as características da página original;
– Definir interfaces de administração para tarefas básicas de configuração do *Broker* ou Adaptadores;
– Utilizar métodos para o registo de Adaptadores, como por exemplo o UDDI;
– Definir métodos de teste apropriados para aplicações que façam uso do Ajax como é o caso do protótipo criado, permitindo assim obter dados sobre o cliente duma maneira mais sistemática e menos empírica;
– Realizar testes com um grupo de utilizadores para fazer uma avaliação qualitativa, usando um sítio de produção com utilização regular.

Referências

1. Eirinaki, M., Vazirgiannis, M: Web Minig for Web Personalization. ACM Transactions on Internet Technology 3,1 1-27 Feb. 2003
2. Mikroyannidis, A., Theodoulis, B. A Theoretical Framework and an Implementation Architecture for Self Adaptive Web Sites. Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on 20-24 Sept. 2004 pages: 558 - 561
3. Mobasher, B., Cooley, R, Srivastava J. Automatic personalization based on Web usage mining. Comunications of the ACM August 2000 43,8
4. Jaideep Srivastava,Robert Cooley,Mukund Deshp,Pang-ning Tan Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. SIGKDD Explorations Jan. 2000 1,2 12-23
5. Thomas Erl: Service-Oriented Architecture - Concepts, Technology, and Design. Prentice Hall, 2005
6. David Flanagan: JavaScript The Definitive Guide. O'Reilly, 2002
7. Site-O-Matic: http://www.niaad.liacc.up.pt/Site-O-Matic
8. Núcleo de Inteligência Artificial e Análise de Dados: http://www.niaad.liacc.up.pt/
9. Mozilla Developer Center - Ajax: http://developer.mozilla.org/en/docs/AJAX
10. Apache Axis: http://ws.apache.org/axis