A Heterogeneous Multi-Agent System for Adaptive Web Applications

Andrea Bonomi, Giuseppe Vizzari
Department of Informatics, Systems and Communication
University of Milan–Bicocca
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
{andrea.bonomi, vizzari}@disco.unimib.it

Marcello Sarini
Department of Psychology
University of Milan–Bicocca
Piazza dell'Ateneo Nuovo 1, 20126 Milan - Italy
sarini@disco.unimib.it

***Abstract*— A web site presents an intrinsic graph–like spatial structure composed of pages connected by hyperlinks. This structure may represent an environment in which agents related to visitors of the web site are positioned and moved in order to track their navigation. To consider this structure and to keep track of these movements allows the monitoring of the site and of its visitors, in order to support the enhancement of the site itself through forms of adaptivity, carried out by specific interface agents. This paper presents a heterogeneous multi-agent system supporting the collection of information related to user's behaviour in a web site by specific situated reactive agents. The acquired information is then exploited by an application supporting the proposal of hyperlinks based on the history of user's movement in the web site environment.**

I. Introduction

A web site presents an intrinsic graph–like spatial structure composed of pages connected by hyperlinks. However, this structure is generally not considered by web servers, which essentially act as a sort of extended and specific File Transfer Protocol servers [1], receiving requests for specific contents and supplying the related data. Several web–based applications instead exploit the structure of the sites itself to support users in their navigation, generating awareness of their position. For instance, many e–commerce sites emphasize the hierarchical structure linking pages related to categories (and possibly subcategories), included products and their specific views, and remind users' relative position (i.e. links to higher level nodes in the tree structure). Some specific web–based applications, mainly bulletin boards and forums (see, e.g., phpBB[1]), are also able to inform users about the presence of other visitors of the web site or even, more precisely, of the specific area of the site that they are currently viewing. Web site structure and users' context represent thus pieces of information that can be exploited to supply visitors a more effective presentation of site contents.

Different visitors, however, may have very different goals and needs, especially with reference to large web sites made up of several categories and subcategories. This consideration is the main motivation for the research in the area of adaptive web sites [2]. The various forms of adaptation may provide a customization of site's presentation for an individual user or even an optimization of the site for all users. There are various approaches supporting these adaptation activities, but they are generally based on the analysis of log files which store low–level requests to the web server: this kind of file is generally made up of entries including the address of the machine that originated the request, the indication of the time and the resource associated to the request. In order to obtain meaningful information on users' activities these raw data must be processed (see, e.g., [3]), for instance in order to collapse requests related to various elements of a single web page (e.g. composing frames and images) into a single entry. Moreover, this kind of information must be further processed to detect groups of requests that indicate the path (web pages connected by hyperlinks) that a user followed in the navigation. Recent results [4] show that this kind of analysis, also referred to as web usage mining, could benefit from the consideration of site contents and structure.

This paper proposes to exploit the graph-like structure of a web site as a Multi–Agent System (MAS) environment [5] on which agents representing visitors of the web site (hereafter *user agents*) are positioned and moved according to their navigation. In particular, in this case, the environment is a virtual structure which allows the gathering of information on user's activities in a more structured way, simplifying subsequent phases of analysis and adaptation of site contents. Furthermore, part of the adaptivity could be carried out without the need of an off-line analysis, but could be the result of a more dynamic monitoring of users' activities. In particular, the paths that are followed by users are often related to recurrent patterns of navigation which may indicate that the user could benefit from the proposal of additional links providing shortcuts to the terminal web pages, as a sort of suggestion to the web site visitor. Index pages may thus be enhanced by the inclusion of links representing shortcuts to the typical destinations of the user in the navigation of the web site. Moreover, links between terminal content pages that are not provided by the static structure of the site can also be identified and exploited. Users without a relevant history (and also anonymous or unrecognized ones) may instead exploit the paths that are most commonly followed by site visitors. Moreover such an information could also be communicated to the webmaster suggesting possible modifications to the

[1] http://www.phpbb.com/

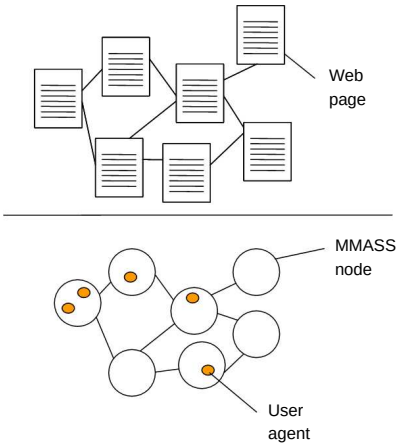

Fig. 1. The diagram shows a mapping between a web site structure and an agent environment.

static predefined structure of the site. This approach provides thus both a support for site optimization, but also for the customization to specific visitor's needs and preferences.

The metaphor of a web site as an environment on which users move in search for information is not new (see, e.g., [6] but also more recent approaches such as [7]), and its application to web site adaptation resembles the emergent, collective phenomenon of trail formation [8] which can be identified in several biological systems. However, this proposal provides more than just gather information on users' behaviours for sake of web pages adaptation or navigation support, but exploits the MAS environment to provide users a means for mutual perception and interaction. In fact information related to users' positions on the environment representing the web site can also be used to supply them awareness information on other visitors which are currently browsing the same page or area of the site. Moreover, to keep track of this information allows the conception of a form of interaction among users that is based on their positions on the site. Essentially, more than just showing a user the other registered visitors that are "nearby" (i.e. viewing the same page or adjacent ones), the system could also allow to communicate with them. This form of interaction, in addition to the web page adaptation function, requires the adoption of a supporting technology that goes beyond the request/response model.

The overall system architecture requires thus proper *interface agents*, able to interact with user agents situated in the previously introduced environment in order to exploit the acquired information on users' behaviours. This second type of agent is totally different from user agents, both from a modelling point of view and with reference to the supporting technology. In fact the web interface agent must be active as long as the related web page is being viewed by a visitor and it must be able, in collaboration with the rest of the system, to proactively modify the page to improve the user's browsing experience. The overall system architecture includes thus heterogeneous agents collaborating to achieve this goal.

The following section describes the general framework of this approach, the mapping between the web site structure and agents' environment, while Section III introduces the gathered information on agents' movement in their environment. Section IV describes an application providing the exploitation of this information for the adaptation of web pages, both for customization and optimization. The adopted technology supporting the design and development of the related interface agent is introduced, and discussed with reference to existing alternatives. A brief comparison of this approach and related work can be found in Section V, and finally concluding remarks and future developments will end the paper.

II. Site Structure and Reactive User Agents

A web site is made up of a set of HTML pages (generally including multimedia contents) connected by means of hyperlinks. It is possible to obtain a graph-like structure mapping pages to nodes and hyperlinks to edges interconnecting these nodes. This kind of spatial structure could be exploited as an *environment* on which user agents related to site visitors are placed and move according to the related users' activities. A diagram showing a sample mapping among a web site and this kind of structure is shown in Figure 1.

This structure can be either static or dynamic: for instance it could vary according to specific rules and information stored in a database (i.e. database driven web sites). However, this kind of structure (both for static and dynamic web sites) can generally be obtained by means of a crawler (see, e.g., Sphinx [9] and the related WebSphinx project[2]); then it could be maintained by having periodic updates.

Given this spatial structure, a multi-agent model allowing an explicit representation of this aspect of agents' environment is needed to represent and exploit this kind of information. Environments for Multi Agent Systems [10] and situated agents represent promising topics in the context of MAS research, aimed at providing first class abstractions for agents environment (which can be more than just a message transport system), towards a clearer and more concrete definition of concepts such as *locality* and *perception*. There are not many models for situated agents, which provide an explicit representation of agent's environment. Some of them are mainly focused on providing mechanisms for coordinating situated agent's actions [11], other provide the interaction among agents through a modification of the shared environment (see, e.g., [12], [13]). An interesting approach that we adopted for this work is represented by the Multilayered Multi Agent Situated Systems (MMASS) [14] model. MMASS allows the explicit representation of agents' environment through a set of interconnected layers whose structure is an undirected graph of nodes (also referred to as sites in the model terminology; from now on we will use the term node to avoid confusion with web sites). The model was adopted given the similarity among the defined spatial structure of the environment and the structure underlying a web site. Moreover, the model defines a set of allowed actions for agents' behavioural specification

[2]http://www-2.cs.cmu.edu/ rcm/websphinx/

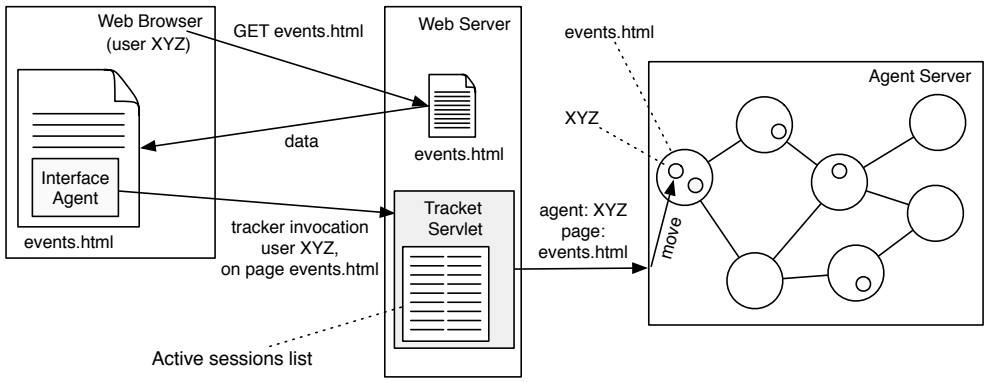

Fig. 2. A diagram showing how user actions influence the related reactive user agent through the capture of requests by the Tracker module.

(including a primitive for agents' movement); for this specific application, however, the constraint which limits the number of agents positioned in a node was relaxed. In fact there is no limit to the number of users that are viewing the same web page.

Moreover a platform for the specification and execution of simulations based on the MMASS model [15] was exploited to implement the part of the system devoted to the management of agents in their environments. The definition of spatial structure of the environment was supplied by the previously introduced crawler, while agents' movement is guided by external inputs generated by the requests issued by the related web site visitor. The general architecture of the system is shown in Figure 2: the *Agent server* module is implemented through the MMASS platform, while the *Web server* is a Tomcat servlet container hosting SnipSnap[3], a Java-based weblog and wiki software. The highlighted *Tracker* module is a implemented through a Java Servlet, which is invoked by every page of the site but does not produce a visible effect on the related web page. The Tracker is responsible for triggering the creation and the movement of agents related to visitors in the environment related to the web site structure. In particular, when a user makes his/her first page request the Tracker is invoked by the interface agent associated to the page. Then the Tracker tries to set a cookie on the client including the session information. If the cookie is accepted, it is possible to use the session information to identify the user; on the other hand, requests from clients not accepting cookies will not be monitored.

The management of agents creation and movement is not as simple as its intuitive description might indicate. In fact, the same user could be using different browser pages or tabs to simultaneously view distinct pages of the site. In other words, a user might be simultaneously following different trajectories in his/her web site navigation. In order to manage these situations, a user can be related to different agents, and his/her requests must be associated to the correct agent (possibly a new one). Finally, agents related to finished (or interrupted) user navigation should be eliminated by the system, storing the relevant part of their state in a persistent way, until the related user requires again a page of the site. In particular, remote users' requests may be divided into two main classes, according to their effects on the Tracker and Agent server:

- *creating a new agent*: whenever a new user requires a web page, the Tracker will invoke the Agent Server requiring the creation of an agent whose starting position is the node related to the required page; the same effect is generated by a request coming from an already registered user which was not present in the system, but in this case information related to previous user agents must be retrieved in order to determine the new agent's state; finally, when an already registered and active user requires a page that is not adjacent to its current one, a new agent related to the new browsing activity must also be created;
- *generating the movement of an agent*: when the viewer of a page follows one of the provided links, the related web browser will generate a request for a page that is adjacent to one of the related agents which must be moved to the node related to the required page; whenever there are two or more agents in positions that are adjacent to the required page, in order to solve the ambiguity and choose the agent to be moved, the Tracker will invoke the Session object in which it stores the current URL related to the viewed page.

The following section will describe how the raw information that can be gathered thanks to the above described framework can be processed in order to obtain higher level indications on users' behaviours. Since the interface agent collaborates to the user monitoring process, more details on this topic will instead be given in Section IV-B.

III. Gathered Information: Browsing Traces

This system allows to gather and exploit two kinds of information: first of all situated agents related to web site visitors have a perception of their local context, both in terms of relative position, adjacent nodes and presence of other visitors; second, agents may gather information related to the paths defined by the browsing activities or the related user in the site itself.

[3]http://snipsnap.org

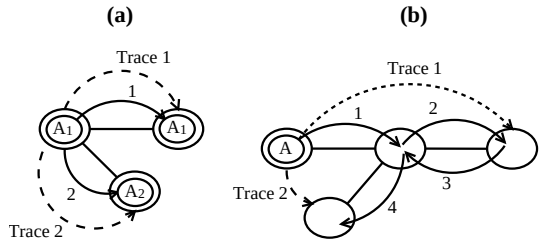

Fig. 3. A diagram describing two traces that are derived by a sequence of user requests.

There are inherent issues in determining in a precise way the actual users' activities on the web site, due to the underlying request/response model: the only available indications on these activities can be obtained by requests captured by the Tracker. In particular, we have an indication of the page that was required by a user and the time-stamp of the request. Starting from this raw information the system can try to detect *emerging links*, which are hyperlinks that are not provided by the structure of the site but can be derived by the behaviour of specific visitors. To this purpose, the concept of *trace* was introduced as a higher level information describing the behaviour of a user. A trace synthesizes a path followed by a user, from the web page representing his/her entry point, to a different point of the environment (i.e. another web page) which may represent an interesting destination. Every agent related to a visiting user is associated to a *temporary* trace, and it may generate several actual traces (also called *closed* traces) in the course of its movement in the environment.

Formally a trace is a three-tuple $\langle A_{Id}, Start, Dest \rangle$, where A_{Id} represents the identifier of the agent to which the trace is related, while $Start$ and $Dest$ indicate the starting and destination node related to the browsing sequence which generated the trace. A new trace is generated when a user enters the site, triggering the creation of a related agent. The starting trace has a null value for the destination node. Subsequent requests by the user generated following hyperlinks will bring the related agent to an adjacent node, and the the $Dest$ field of the corresponding trace will be modified in order to reflect user's current position. Non trivial traces provide $Start$ and $Dest$ nodes that are not directly connected by means of a hyperlink.

There are two relevant exceptions to the basic rule for trace update, that are related respectively to the *duplication* of a trace and to its *closing*. According to the previously introduced informal definition, a trace should be coherent in time and space. In fact, whenever the same user requires simultaneously two or more different pages he/she is probably following distinct search trajectories, possibly even related to different goals. In this case, as previously introduced, the Tracker will detect this situation and create additional agents that refer to the same user. Figure 3 shows two sample situations providing respectively trace duplication and closing: in (a) the user has chosen to open a hyperlink in a new browser page (request 1) and then has followed another link in the first browser page (request 2). According to the previously described Tracker behaviour, two agents are now associated to the user, and they are associated to different traces sharing the $Start$ field.

In (b), instead, the user has followed links 1 and 2 from the starting page, then he/she made a step back (request 3) and eventually moved to the last known position (request 4). The step back causes the closure of the temporary trace associated to the agent (Trace 1 in the Figure), and the creation of a new temporary one with the same $Start$ field (Trace 2). In this case the step back may have different interpretations: it could refer to a negative evaluation of the page contents but it could also indicate the fact that the user has found what he/she was searching for. An information that could be exploited to determine if the $Dest$ field of the trace was interesting for the user is the time interval between request 2 and 3: for instance, given Δt_d a threshold indicating the minimum time required to reasonably inspect the content of a specific web page, if $timestamp(3) - timestamp(2) < \Delta t_d$ then Trace 1 could be ignored. However, the mere interval between the two requests is not a safe indicator of the fact that the page was actually viewed and considered interesting.

In fact, the time spent on a web page is also important in order to determine when a temporary trace must be closed. In fact, whenever a user does not issue requests for a certain time we could consider that his/her browsing activity has stopped, possibly because he/she is reading the page related to the $Dest$ field of the trace associated to the related agent. In other words, every agent has a timer, set to the previously introduced threshold Δt_d, which is set when the agent is created and it is reset whenever it moves. The action associated to this timer specifies that its temporary trace becomes closed, and a new timer is set: the action associated to this second timer caused the disappearance of the agent from the system, and the storage of the related state.

It is important to note that even anonymous visitors (i.e. non authenticated ones) whose clients are accepting cookies, can be tracked and can thus generate traces, although anonymous ones. The latter can be exploited for sake of web optimization but are not relevant for sake of user specific site customization.

User agents provide thus a support to interface agents by monitoring users' behaviours and, in this specific case, selecting relevant traces. Figure 4 shows how the user agents interact with the interface agents to provide them with relevant information for page adaptation, but more details on this topic will be provided by the following section.

IV. The Web Interface Agent

The aim of the Interface Agent is to improve the browsing experience of a user by adapting the page he/she is currently viewing to his/her preferences, needs or habits. To do so, it must be active during the time–span in which the page is visualized by the browser, and it must be able to dynamically alter its appearance. To do so, it must also be able to interact with the previously introduced system to be informed about past user's behaviour. In other words the interface agent is

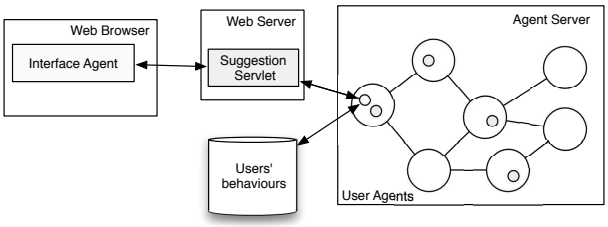

Fig. 4. A diagram showing the interaction among an interface agent, the user agent (in the MMASS environment) and the users' behaviors database.

a client–side component, "living" in the web browser and interacting with it in a proactive way, as shown in Figure 4.

In the following sections, we describe the technology adopted to implement the interface agent, comparing it with other currently available technologies that could have been selected to develop this kind of client-side web application. Then the behaviour of the interface agent is briefly introduced, focusing on its setting in the overall architecture and on the adopted strategy for page adaptation.

A. Technologies for Web Interface Agents: Java Applet, Flash and AJAX

Today there are several technologies suitable to develop rich client–side web applications, and in particular interface agents able to "live" in a common web browser. The most common are Java Applet, Macromedia (now Adobe) Flash and AJAX. We intentionally chose not to consider recent browser extensions and plug-ins for the visualization of 3D virtual environments, and to focus on more traditional forms of web browser interfaces.

Java Applet[4] is the oldest technology used to provide interactive features to web applications. An applet is a Java software component that runs in a Web browser using the Java Virtual Machine. Applets can be included in HTML (or XHTML) pages in the same way as an image or another multimedia content, and they are executed in a *sandbox*, an infrastructure preventing them from accessing client's local data (though there may be exceptions to this principle, and in particular *trusted applets*). This kind of approach is very powerful because applets can exploit all the Java API: they can, for example, generate complex user interfaces, with a rich multimedia support (e.g. 3D graphics, sound, movies), or they can interact with server–side application via Web Services, Java RMI (Remote Method Invocation) or CORBA. It is possible to develop very complex applications using common Open Source Java IDEs (like Eclipse or Netbeans) and run them in web browser as applets. Though Java Applet can be a suitable technology for many complex web application, it is difficult to implement an interface agent with an applet because of its lack of integration with the web browser. An applet is in fact confined in a sandbox and cannot manipulate the data of the page in which it is being executed. For example, an applet cannot be used to extract all the links of the current user page. Another disadvantage of Java Applet is represented by the requirements of the Java Runtime Environment: first of all it is not available by default on all web browsers, moreover it has a large memory occupation (around 20 Mb) and applets cannot start until the Java Virtual Machine is running.

Flash[5] is a multimedia technology commonly used to create animations, to build interactive web pages and to develop client-side web applications. The flash files (called *Flash Movies*) run in a virtual machine called *Flash Player*, that is available for a wide variety of different browsers, platforms and devices. The Flash Player is smaller than Java runtime (less than 1 MB) and it is installed on over 500 million devices and more than 97% of Internet-enabled desktops[6]. Moreover, a Flash Player is embedded in many consumer electronics devices, like Kodak EasyShare-One digital camera: the user interface, built using Flash, enables simple navigation during picture taking and sharing, and includes rich graphical scene modes. Flash Movies can be programmed with a scripting language called *ActionScript*, that is an ECMAScript[7]–based programming language, object oriented, loosely–typed and has a syntax quite similar to C. In contrast with JavaScript (which is also ECMAScript compliant), ActionScript is compiled into bytecode which is interpreted by a virtual machine. ActionScript has a rich API supporting the elaboration of numbers, strings, XML and graphical element (vectorial and raster); it allows to play sounds and movies and to interact with server side application with a fast proprietary protocol (Flash Remoting[8]) or the slower SOAP (Simple Object Access Protocol).

AJAX (shorthand for Asynchronous JavaScript and XML) is not a technology in itself, but a term that refers to the use of a group of technologies together [16]. In fact, AJAX is a combination of JavaScript, DHTML (Dynamic HTML)[9], XML and the Remote Scripting (also described in [16]). Remote Scripting is used to deliver content dynamically without the need to refresh the page and DHTML is a method for creating interactive web pages by using a combination of a markup language (HTML) and a client–side scripting language (JavaScript): one major use of JavaScript is to write functions that are embedded in or included from HTML pages and interact with the Document Object Model (DOM). Other typical examples of JavaScript usage are: validating web form input, opening popup window, playing sounds, changing images size and performing text conversion operation. The scripts can be embedded in HTML pages or contained in *.js* files linked to the web pages. The overall AJAX web application model, compared to traditional web applications, is shown in Figure 5. Since JavaScript is an interpreted language, errors are not detected until the faulty program line is executed. Another problem of AJAX (and JavaScript in general) are the

[4]http://www.sun.com/applets/

[5]http://www.adobe.com/products/flash/

[6]NPD Online survey, conducted in April 2006

[7]http://en.wikipedia.org/wiki/ECMAScript

[8]http://www.adobe.com/products/flashremoting/

[9]http://www.w3.org/DOM/faq.html#DHTML–DOM, http://www.w3schools.com/dhtml/

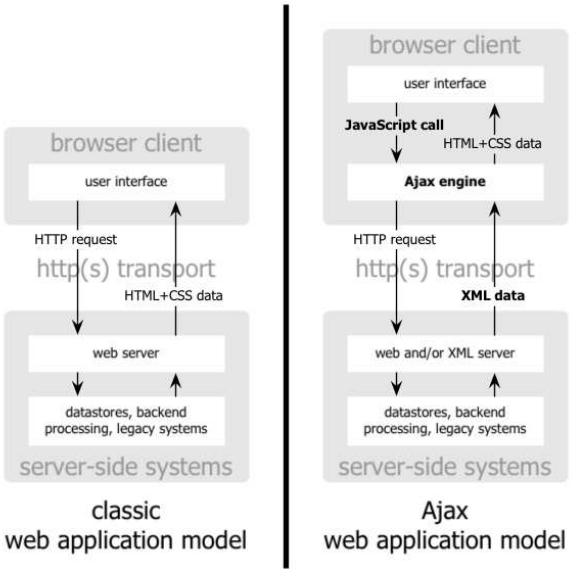

Fig. 5. The traditional web applications compared to the AJAX model. Figure by J. J. Garrett taken from [16].

differences between different JavaScript engine implementations, so applications must be tested systematically on the different target browsers and platforms. Nonetheless, AJAX is not only a scripting language that supports a rapid prototyping of web applications but it is also suitable for industry-strength systems (from WebGIS applications like Google Maps[10], to complex enterprise messaging and collaboration systems like Zimbra[11]).

To compare the different technologies, several sample applications that are available and freely accessible online can be evaluated. In particular several instant messengers have been implemented adopting Java Applet, Flash and AJAX technologies: for instance ICQ2Go![12] is is available both as a Java Applet and as a Flash application and Meebo[13] is developed with AJAX. Despite all are instant messenger applications, the user experience is very different: the Java version as ICQ2Go! has a very long startup time and it requires a huge amount of memory but it has most functions of the stand–alone ICQ client application and it is able to communicate with the server adopting the common ICQ protocol. The new Flash version of ICQ2Go! and Meebo are comparable in terms of user experience: both of them start much faster than the ICQ2Go! applet, but they still have a very good look and feel and an extensive set of functionalities. However, both the Flash and the AJAX version required a special server–side wrapper because they can communicate only with a XML protocol.

After the analysis of the various technologies, we have chosen to adopt AJAX in order to develop the Interface Agent. With AJAX, it is possible to create an agent hosted in the web browser that remains alive and active during the visualization of a web page. So it is possible to go beyond the classic web request/response model and develop proactive interface agents. We chose AJAX instead of Flash because it is possible to develop AJAX applications with Open Source tools (in fact, only a common text editor is needed). Today, a commercial IDE is required to build Flash web applications; although there is an Open Source ActionScript compiler[14], the lack of a proper full–featured Open Source IDE and mature tools for user interface drawing is a major drawback. Compared to Java Applet, instead, AJAX is lightweight and better integrated in the browsing environment: JavaScript functions have a complete control on the page content while applets are confined in a sandbox. This is a very important feature because the aim of an interface agent is to interact with the user, so an agent with more freedom of action over the interface can perform its task more effectively.

B. The Interface Agent in the Overall Architecture

The interface agent starts its activity when a web page of the site is loaded into client Web Browser. The first action performed by the agent is adding to every link of the page a parameter (called *linkfrom*) with the URL of the current page as value. This action permits to identify the source page of every subsequent request. For example, assume that current page address is `http://host/index.html`, the link `<a href="events.html">Events</a>` included in the page will be rewritten as

```
<a href="events.html?linkfrom=index.html">Events</a>
```

Similarly, `<a href="news.jsp?news=1">Events</a>` will be rewritten as

```
<a href="news.jsp?news=1&linkfrom=index.html">Events
</a>
```

The content of the page is dynamically changed at client-side by JavaScript DOM (Document Object Model), so the original page on the server remains intact. DOM will allow scripts to dynamically access and update the content, structure and style of current page. The document can be further processed and the results of that processing can be incorporated back into the presented page. The agent doesn't update every link of the page, but only the HTTP links to the current site. So links to other sites, or links to a FTP repository or mail address remain unchanged.

The next action performed by the interface agent is to call the tracker. If the current page is called with the linkfrom parameter, this parameter is passed to the tracker. The tracker uses this parameter to build the traces. For example, if the URL of the current page is `events.html?linkfrom=index.html` the user's last page was `index.html`. The tracker can add a trace for the current user from index.html to events.html (or update an existing one). The tracker doesn't perform this operation itself, instead it informs the user agent on the MMASS environment, which is responsible for adding the trace. Then the interface agent can

[10]http://maps.google.com/
[11]http://www.zimbra.com/
[12]http://go.icq.com/
[13]http://www.meebo.com/
[14]http://www.mtasc.org/

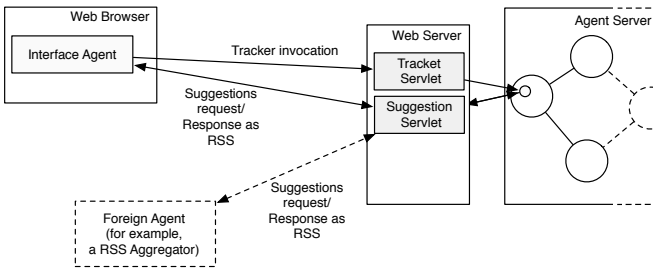

Fig. 6. Interface Agent and Foreign Agent interaction with the MMASS user agent are performed through the Suggestion Servlet.

query the server to obtain the emerging links to be suggested to the user.Suggestions are in fact generated on the server–side and are published as an RSS[15](Really Simple Syndication) feed. The agent suggestion request is managed by the user agent (analogously as for traces). We choose RSS instead of a proprietary format because this allows foreign interface agents (other then our interface agent) to interact with the system.

The interface agents loads the RSS by using the `XMLHttpRequest`[16] class, which allows to perform an asynchronous request to the web server hosting the current web page and to store the response in a local variable. The response could be a XML document or plain text. In the first case, `XMLHttpRequest` stores the retrieved data in a DOM-structured object, which can be navigated using the standard JavaScript DOM access methods and properties, such as `getElementsByTagName()` and `childNodes[]`. The following code is an example of using `XMLHttpRequest` to asynchronously request the server side page `suggestions.jsp`:

```
req = new XMLHttpRequest();
req.onreadystatechange = processReqChange;
req.open("GET", "suggestions.jsp", true);
req.send(null);
```

In order to find out when the method has finished retrieving data, a specific event listener must be defined: in this case the method is `processReqChange`, reported in the following code snippet:

```
function processReqChange() {
 if ((req.readyState == 4) && (req.status == 200)) {
  // Gets the items from the XML document
  var xml = req.responseXML;
  var items = xml.getElementsByTagName("item");
  // Builds new suggestions
  var html = "";
  for (item in items) {
   var title = getValue(item, "title");
   var link = getValue(item, "link");
   // Adds a link and a carriage return
   html += "<a href='" + link + "'>" + title + "</a>";
   html += "<br/>";
  }
  // Replaces the content of the suggestions box
  document.getElementById("sBox").innerHTML = html;
 }
}
```

[15]http://www.rssboard.org/rss-specification
[16]http://www.w3.org/TR/XMLHttpRequest/

This method of the interface agent parses the RSS document and displays the suggestions in a box in the web page. This operation is done by using DHTML: the agent searches for the suggestion box (`sBox`) in the DOM of the page (which is a tree representation of the page HTML source) and than it replaces the content of the suggestion box with the freshly generated one. The latter is based on RSS suggestions: for each suggested page (represented as an item in the RSS) the Interface Agent adds a link to the page and uses the title of the page as label for the link. The following RSS is a suggestion example:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0"
  xmlns:lintar="http://www.lintar.disco.unimib.it/">
 <channel>
  <title>Suggested contents for index.html</title>
  <link>http://example.com/index.html</link>
  <language>en</language>
  <pubDate>Wed, 28 Jun 2006 02:28:19 +0200</pubDate>
  <ttl>1</ttl>

  <item>
   <title>Events</title>
   <link>http://example.com/events.html</link>
   <guid>http://example.com/events.html</guid>
   <lintar:usersTraces>75</lintar:usersTraces>
   <lintar:onlineUsers>3</lintar:onlineUsers>
  </item>

  [ ... more items ... ]

 </channel>
</rss>
```

In this example, the first suggested element is the Events page, whose URL is `http://example.com/events.html`. The tags in the `lintar` namespace are our extension to the basic RSS: the `<lintar:onlineUsers>` tag identifies the number of users currently viewing the page and the `<lintar:usersTraces>` tag represent the intensity of footprints on the page, in the spirit of [6]. Footprints are signs that one or more users have recently viewed the page. This information is also displayed by the interface agent on the suggestion box: the number of online users is displayed as a picture of little red man and the presence of users traces is represented by corresponding icon. The number of online users and the intensity of footprints are displayed in a tip box that it is shown when the mouse arrow is over the picture. It must be noted that the interface agent does not just provide a “one shot” behaviour. In fact, when initialized, it sets a timeout for a cyclical invocation of its main execution cycle by the web browser. In this specific application, in particular, it is this able to update and refresh the indication on the presence of other visitors and footprints on suggested pages. The overall cycle of interaction between the interface agent and the back end of the system is illustrated in Figure 6 and a screenshot of the web page enriched by the interface agent is shown in Figure 7.

C. The Adaptation Strategy

Every MAS agent of the implemented system provide personalized suggestions about items that user will find interesting, according to the history of the user and to the other

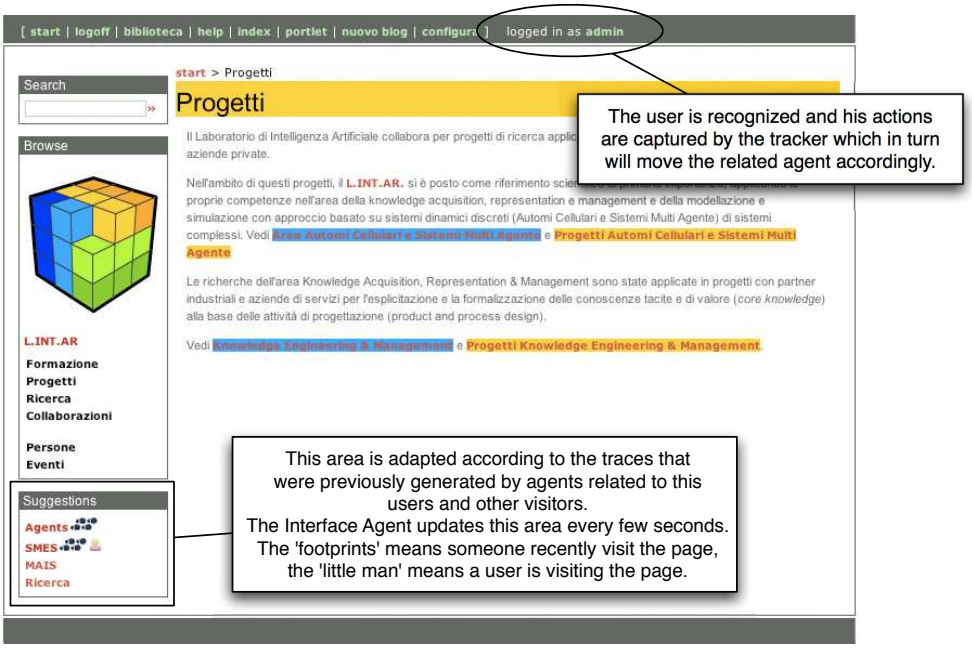

Fig. 7. A screenshot of a web page adapted according to gathered traces.

users path. These suggesting links have relationship with the previously introduced traces, which represent behaviors and movements of a user in a web site: the strategy which is adopted to select the most relevant traces to be presented to a given user considers the occurrence of trace generation and the success rate of the traces that were proposed.

A first element of this strategy is adopted when new users (or non authenticated ones) enter the site. In this case the user has no previous history (or it is not possible to correlate the user with his/her history), and the adopted strategy considers all stored traces, not considering the user which generated them. An additional information that is stored with traces is the number of times that the related trace was effectively selected and shown to a user and the number of times that the related link was effectively exploited by a user. This kind of information allows to obtain an indication of the success rate of the suggestions that were chosen by the agent, and can be exploited to select the traces to be shown in the adaptive block. When the agent has an indication of the user which issued the request, it may focus the selection activity to those traces that compose the history of user's activities in the web site, in a web customization framework. In fact traces include an indication of the agent which generated them, and in turn agents are related to registered users. Moreover, in order to focus on a specific user's history but do not waste the chance to exploit other users' experiences, just two of the three available slots for emergent links are devoted to traces that were generated by that user and one is selected according to the strategy adopted for anonymous or new users. Because the time spent on a page had a strong correlation with explicit interest [17], the adopted strategy uses this information to refine the proposed suggestions.

An example of page adaptation refers to the adoption of a recurrent trace leading from the index of the web site to a content page, that is not directly connected to the index but that is visited very frequently. This kind of "vertical"[17] emerging link is frequently observed in the prototypal implemetation of the system, which is installed in a web site presenting information about a research laboratory as well as information on courses held by members of the group[18]. Since the number of students of some of these courses is very high, they frequently generate traces connecting the index to the page related to those courses. These traces represent effective shortcuts allowing to bypass intermediate index pages related to education activities and university courses. However, emerging links can also connect pages deep in the site structure. For example, a page related to a project might not be explicitly connected to another page describing a particular modeling approach adopted in that project, but a user might browse the web site and effectively discover that page, causing the generation by the system of a correspondant trace connecting the project and the modeling approach. This trace might not be extremely relevant to all visitors of the web site, due to the fact that this navigation path will probably be not very frequent, but if the visitor is a registered user the trace could be stored and suggested anyway, since a number of slots in the adaptive area of the page is reserved to user–generated emerging links.

This strategy for the exploitation of the gathered and stored traces, based on users' behaviours and movement in the web site environment, represents a very simple way of exploiting this kind of information without requiring an off-line analysis

[17]Here vertical is intended as describing the typical navigation path starting from an index page and going deeper into the web site.

[18]http://www.lintar.disco.unimib.it

of the logs generated by the web server. The design, implementation and test of more complex strategies, for instance based on details of the outcomes of emerging link proposals (e.g. which user effectively followed the suggested adaptive hyperlink) are object of future works.

V. Related Work

There are several different approaches and relevant experiences in the area of web site adaptation, and some of them are also related to agent technologies. In particular, a relevant approach provides the adoption of information agents supporting users in their navigation [18]. These agents generally consider both the specific behaviour of the user and the actions of other visitors, and adopt multiple strategies for making recommendations (e.g. similarity, proximity, access frequency to specific documents).

The Footprints system [6] instead provides a site optimization through the metaphor of site visitors leaving traces in their navigation. These signals accumulate in the environment, generating awareness information on the most frequently visited areas of the web site. No user profile is needed, as visitors are essentially provided this information which could represent an indicator of the most interesting pages to visit. The metaphor of the structure of the web site as an environment on which visitors move in their search for information is very similar to the one on which the proposed framework is based, but we also propose the exploitation of the gathered information on users' paths for user specific customization. Another interesting recent work [19] represents an attempt to integrate interaction mechanisms similar to the one adopted by Footprints, often referred to as stigmergic interaction mechanisms [20], and cognitive agents. This line of research could represent an interesting way to integrate the proposed approach, which is able to generate and manage awareness contextual information, with higher level mechanisms and strategies of adaptation.

Other approaches provide instead the generation of index pages [3], that are pages containing links to other pages covering a specific topic. These pages, resulting from an analysis of access logs aimed at finding clusters grouping together pages related to a topic, are proposed to web masters in a computer-assisted site optimization scheme. A different approach provides the real-time generation of shortcut links [21], through a predictive model of web usage based on statistical techniques and the concept of expected saving of a shortcut, which considers both the probability that the generated link will be effectively used and the amount of effort saved (i.e. intermediate links to follow). In particular, this framework is very similar to the one proposed here with reference to the aims of the overall system, but it incorporates a complex algorithm for off-line analysis of logs, while the proposed approach provides a light and dynamic generation of most probable useful links and the storage of these proposals and high level information on site usage for a possible further off-line analysis.

A different approach to web site adaptation provides the adoption of a learning network to model the evolution of a distributed hypertext network, such as a web site [22]. Also in this case the adaptation provides a modification in the structure of a web site, and the concept of emergent link and the underlying mechanisms present a similarity with the learning rules adopted for that kind of learning network. However that approach also provides a deep modification in the architecture of the site and modifications in the web protocols, while this work aims at providing a solution that can be easily integrated with a traditional web architecture. Moreover, recent developments of that line of research were aimed at identifying analogies and relations among words by means of web mining [23], rather than realizing adaptive web systems.

The introduced system supporting web site adaptation seems more similar to a recommendation system. A relevant type of recommender exploiting users' behaviours to decide which contents could be interesting for a certain visitor is represented by the collaborative filter approach [24]. The latter has been adopted in different recommendation systems, filtering mail messages, newsgroup articles and web contents in general, but typically requires users to rate these items. Moreover, it generally provides a concept of explicit users descriptions through profiles which can be compared to determine similarity among them. The idea is that contents that received a high rating by a certain user could be considered interesting by a similar user. The introduced system instead does not require an explicit rating of contents, but it rather observes the frequency of specific navigation paths, and exploits emergent links for customization or optimization of site structure. However, the adaptive block of the page can include emerging links that are not related to the specific visitor who is currently browsing that page, but were generated by other users which frequently followed paths that the current one still did not follow. From this point of view, the system provides a very basic collaborative browsing scheme, but a more through analysis of a possible integration with this approach is object of current and future works.

VI. Conclusions and Future Developments

This paper introduced a general framework providing the adoption of a web site as an environment on which agents related to visitors move and possibly interact. This approach allows the gathering of a structured form of information on users' behaviours and activities in the web site. The concept of emerging links and traces have been introduced in order to support an application exploiting information on users' browsing history for sake of web pages adaptation. The introduced framework and the application to web site adaptation have been designed and implemented, exploiting a platform supporting systems based on the MMASS model.

A campaign of tests aimed at evaluating the effectiveness of the adaptation approach, and also for sake of tuning the involved parameters (e.g. timings, number of presented possible emerging links) is under way. This evaluation will

be based on user interviews and also on the exploitation of the gathered information of the success rate of proposed adaptive hyperlinks. Such an indicator might be obtained as a ratio between the number of times an emerging link has been actually selected by a user and the total number of times its has been shown. However, it must be noted that we currently do not have an indication of threshold to discriminate successful suggestions from unsatisfactory ones; a further analysis of methods adopted to evaluate related approaches is currently being carried out. The results of this evaluation might also lead to consider the modelling, design and implementation of more complex trace selection strategies, and thus a more complex behaviour for the interface agent.

Future works will be focused on the introduction and exploitation of higher level semantic information related to the site structure and contents, and thus agents' environment, aimed at providing additional forms of adaptation, including images and multimedia contents. While in [25] an analysis on how a conceptual view on the topics may be used as an additional level of description of the environment, another aspect that will be considered is the possibility to improve the effectiveness of web–based applications supporting processes with adaptive functionalities. Finally, a further development provides also the design and implementation of a prototype supporting the context-aware interaction among web site visitors. In this framework, the environment related to the web site also supports the mutual perception of the agents situated in it and it also supports a form of interaction among them depending on their relative positions. The latter can be thus considered as a form of context–dependant interaction. A more thorough analysis of the possible applications of this approach can be found in [25], and a prototypal implemetation of these interaction mechanisms is currently under way.

REFERENCES

[1] A. S. Tanenbaum, *Computer Networks - third edition*. Prentice Hall, 1996.

[2] M. Perkowitz and O. Etzioni, "Adaptive Web Sites: an AI Challenge." in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 1997)*, 1997, pp. 16–23.

[3] ——, "Adaptive Web Sites," *Communications of the ACM*, vol. 43, no. 8, pp. 152–158, 2000.

[4] R. Cooley, "The Use of Web Structure and Content to Identify Subjectively Interesting Web Usage Patterns," *ACM Transactions on Internet Technology*, vol. 3, no. 2, pp. 93–116, 2003.

[5] D. Weyns, H. V. D. Parunak, F. Michel, T. Holvoet, and J. Ferber, "Environments for Multiagent Systems State-of-the-art and Research Challenges." in *Environments for Multi-Agent Systems, First International Workshop (E4MAS 2004)*, ser. Lecture Notes in Computer Science, vol. 3374. Springer–Verlag, 2005, pp. 1–47.

[6] A. Wexelblat and P. Maes, "Footprints: History-Rich Tools for Information Foraging," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press, 1999, pp. 270–277.

[7] J. Liu, S. Zhang, and J. Yang, "Characterizing Web Usage Regularities with Information Foraging Agents," *IEEE Transactions Knowledge and Data Engineering*, vol. 16, no. 5, pp. 566–584, 2004.

[8] D. Helbing, F. Schweitzer, J. Keltsch, and P. Molnár, "Active Walker Model for the Formation of Human and Animal Trail Systems," *Physical Review E*, vol. 56, no. 3, pp. 2527–2539, January 1997.

[9] R. C. Miller and K. Bharat, "Sphinx: a Framework for Creating Personal, Site-specific Web Crawlers," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 119–130, 1998.

[10] D. Weyns, F. Michel, and H. V. D. Parunak, Eds., *Environments for Multi-Agent Systems, First International Workshop (E4MAS 2004)*, ser. Lecture Notes in Artificial Intelligence, vol. 3374. Springer–Verlag, 2005.

[11] D. Weyns and T. Holvoet, "Model for Simultaneous Actions in Situated Multi-Agent Systems," in *First International German Conference on Multi-Agent System Technologies, MATES*, ser. Lecture Notes in Computer Science, vol. 2831. Springer–Verlag, 2003, pp. 105–119.

[12] M. Mamei, F. Zambonelli, and L. Leonardi, "Co-fields: Towards a Unifying Approach to the Engineering of Swarm Intelligent Systems," in *Engineering Societies in the Agents World III: Third International Workshop (ESAW2002)*, ser. Lecture Notes in Artificial Intelligence, vol. 2577. Springer–Verlag, 2002, pp. 68–81.

[13] K. Hadeli, P. Valckenaers, C. Zamfirescu, H. V. Brussel, B. S. Germain, T. Hoelvoet, and E. Steegmans, "Self-organising in Multi-Agent Coordination and Control Using Stigmergy," in *Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering*, ser. Lecture Notes in Computer Science, vol. 2977. Springer–Verlag, 2004, pp. 105–123.

[14] S. Bandini, S. Manzoni, and C. Simone, "Dealing with Space in Multi–Agent Systems: a Model for Situated MAS," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. ACM Press, 2002, pp. 1183–1190.

[15] S. Bandini, S. Manzoni, and G. Vizzari, "Towards a Platform for Multilayered Multi Agent Situated System Based Simulations: Focusing on Field Diffusion," *Applied Artificial Intelligence*, vol. 20, no. 4–5, pp. 327–351, 2006..

[16] J. J. Garrett, "AJAX: a New Approach to Web Applications," Adaptive Path Essay, Tech. Rep., 2005. [Online]. Available: http://www.adaptivepath.com/publications/essays/archives/000385.php

[17] M. Claypool, P. Le, M. Waseda, and D. Brown, "Implicit Interest Indicators." in *Intelligent User Interfaces*, 2001, pp. 33–40.

[18] M. J. Pazzani and D. Billsus, "Adaptive Web Site Agents," *Autonomous Agents and Multi-Agent Systems*, vol. 5, no. 2, pp. 205–218, 2002.

[19] A. Ricci, Omicini, M. Viroli, L. Gardelli, and E. Oliva, "Cognitive Stigmergy: a Framework Based on Agents and Artifacts," in *3rd International Workshop "Environments for Multi-Agent Systems" (E4MAS 2006)*, D. Weyns, H. V. D. Parunak, and F. Michel, Eds., 2006, pp. 44–60.

[20] G. Theraulaz and E. Bonabeau, "A Brief History of Stimergy," *Artificial Life*, vol. 5, no. 2, pp. 97–116, 1999.

[21] C. R. Anderson, P. Domingos, and D. S. Weld, "Adaptive Web Navigation for Wireless Devices," in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, 2001, pp. 879–884.

[22] J. Bollen and F. Heylighen, "Algorithms for the Self-Organisation of Distributed, Multi-User Networks. Possible Application to the Future World Wide Web," in *Proceedings of the 13th European Meeting on Cybernetics and Systems Research*, R. Trappl, Ed. Austrian Society for Cybernetic Studies, 1996, pp. 911–916.

[23] F. Heylighen, "Mining Associative Meanings from the Web: from Word Disambiguation to the Global Brain," in *Proceedings of the International Colloquium: Trends in Special Language & Language Technology*, R. Temmerman and M. Lutjeharms, Eds. Standaard Editions, Antwerpen, 2001, pp. 15–44.

[24] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an Open Architecture for Collaborative Filtering of Netnews," in *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM Press, 1994, pp. 175–186.

[25] S. Bandini, M. Sarini, C. Simone, and G. Vizzari, "WWW in the Small: Towards Sustainable Adaptivity," *World Wide Web Journal*, 2006 (to appear).