Sensor-driven adaptation of Web document presentation

Spyros Nathanail, Vassileios Tsetsos, Stathes Hadjiefthymiades

Pervasive Computing Research Group, Dept of Informatics and Telecommunications
University of Athens, Panepistimiopolis, Ilissia, 15784, Greece
{s.nathanahl, b.tsetsos, shadj}@di.uoa.gr
Tel: +30 210 7275127, Fax: +30 210 7275601

Abstract. The increasing variety of user device technologies has raised the necessity for ubiquitous content provision, which is characterized by "intelligent" content delivery to end users, irrespectively of their device. Moreover, changes in the user environment create the necessity of appropriately adapting the content and its presentation so that it can be more easily perceived. In this paper, we present a multimodal and adaptive system for Web user interfaces that deals with these challenges. Using as input a generic and abstract form of Web content, our system transforms it to a specific data format suitable for the client device. In addition, it uses contextual information, gathered from the user environment through a wireless sensor network, in order to present the served content in an optimal way under the current environmental conditions.

Keywords: adaptive user interfaces, abstract user interfaces, context-awareness, wireless sensor networks

1 Introduction

According to the Ubiquitous Computing paradigm, which is increasingly penetrating our everyday lives, information processing and communication tasks should be available to roaming users with limited effort from their part. Central concepts in this paradigm are, among others, seamless support for device heterogeneity, advanced user interfaces and context-awareness. Hence, the ability of adapting the information to many different forms, according to the user's general context, is deemed crucial for smart environments and services. Similarly to the need for information adaptation there also arises the need for *context-aware presentation* of the delivered information to the users. The term "context" refers to everything that affects the interaction between a user and her computing environment, such as device capabilities, user status/profile and environmental conditions [8].

The vision of context-aware content presentation is what has mainly motivated our work. In this paper, we describe the design, implementation and performance evaluation of a system, named Chameleon, that performs such adaptation. An indicative use case for the proposed system is the following:

An emergency response team using various hand-held devices (e.g. Tablet PCs, PDAs) needs information that can be updated and adapted rapidly and accurately, according to their surrounding environment. Since that team usually operates under stress and extreme environmental/weather conditions, changes in factors such as light and noise levels must be treated by the application, without extra effort and attention from the team members. Hence, the proposed system a) transforms the content to a format appropriate for each member's device, and, b) adapts the content presentation elements (colour, size, etc.) to the environmental conditions, provided that the adaptation rules and the sensing infrastructure have been set.

Obviously, Chameleon can also facilitate the Web content provisioning process for people facing various disabilities (e.g., elderly people or people with visual impairments).

In general, an adaptive and multimodal user interface system that adheres to the Ubiquitous Computing paradigm should address three requirements:

a) access to the same content from devices using different platforms and having different capabilities,
b) adaptive formatting of the content in order to address changing contextual conditions,
c) seamless provision of the above functionality (i.e., transparently to the end user)

The proposed system fulfills these requirements by extending the functionality of a web server so as to deliver content in a format tailored to the user device capabilities and the contextual status of the mobile or stationary users. Specifically, Chameleon senses the environmental conditions near users through wireless sensor networks, and properly adjusts the formatting/presentation of the requested content, according to predefined presentation rules. Such rules may be the same for all users or adjusted appropriately on a per-user basis.

The implementation of such system would be of little value and utility without the recent advances and wide acceptance of Wireless Sensor Networks (WSN). Such networks enable remote sensing and can be attached to moving objects, such as handheld devices or wearable computers. Hence, they provide a ubiquitous sensing infrastructure that can be transparently incorporated in pervasive and mobile services.

The organization of the paper is as follows. In Section 2, we discuss the overall architecture and functionality of the system as well as describe its end-to-end workflow. In Section 3, we further discuss some selected implementation issues. In Section 4, the setup and the results of a performance evaluation are presented. Finally, in Section 5, we present some related work and the paper concludes with some directions for further research.

2 Chameleon Architecture and Functionality

Chameleon consists of three modules. The *M-module* provides the multimodality functionality. The content is initially described in an *abstract user interface language*, which only describes its structure and semantics. The M-module transforms such content according to a user interface language that can be supported by the user device (hereinafter referred to as *target language*). The transformed content contains also presentation-specific annotations. The *A-module* provides the adaptation functionality of Chameleon. It is responsible for formatting the content resulting from M-module according to i) certain presentation rules, and, ii) the data received from the sensing infrastructure (i.e., WSN). The S-module is responsible for collecting and handling the sensor readings. These modules, along with the basic data flows, are depicted in Figures 1 and 2 and described in the following paragraphs.

A session begins when the user accesses the server from a web client (i.e., browser) and issues a certain request for content. For the rest of this paper, we assume that such content is a Web page. After the server has received the request, it retrieves the page in its original, abstract form and forwards it to the M-module.

2.1 System Modules

The M-module is responsible for the translation of the original, abstract web page into a final web page that suits the browser and device capabilities of the requesting user. The page transformation is performed by applying a set of XSL Templates [16] on the original page. The outcome is a page formatted in a target language (e.g., HTML) that will eventually be sent to the user device. The whole process is transparent to the user.

It should be noted that Chameleon does not send an abstract XForms document to the user and, thus, it does not burden the user's browser with the task of converting it to an appropriate target language. Such an approach would not permit the context-driven adaptation of the final page, since most end devices cannot retrieve and process contextual information locally. In the current implementation we support only HTML as a target language. Hence, we will not deal with multimodality issues in the rest of the paper, but focus on the sensor-driven presentation adaptation instead.

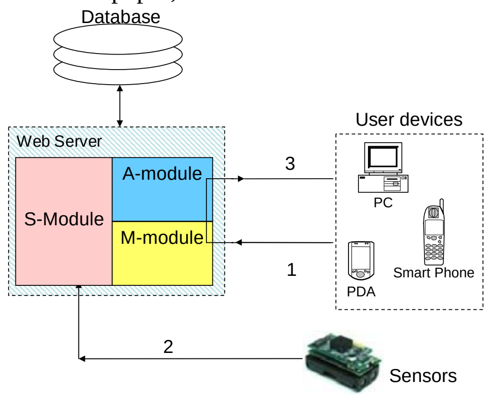

Fig. 1. System architecture and main data flows (1: User request, 2: Sensor readings, 3: System response)

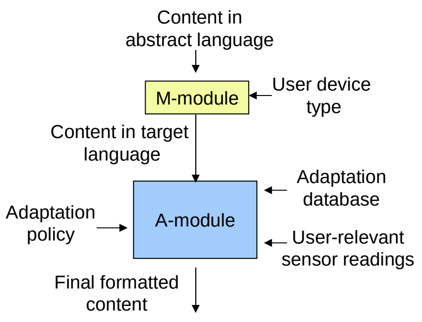

Fig. 2. Basic system functionality

After the M-module has performed the page transformation, the system forwards the processed page to the A-module. The latter is responsible for the adaptation and formatting of the page content based on the contextual information sensed from the user environment. Such information is retrieved through a wireless sensor network, the sensors of which may be physically located on the user's device (hereinafter called *device sensors*), or can be deployed at the broader area of the user (hereinafter called *area sensors*). The sensors constantly communicate with the system through a middleware in order to report their readings (see also Section 3.1).Prior to deciding how the adaptation of the content will be performed, the system must select which sensors will affect the presentation adaptation. This is the responsibility of the S-module.

Upon system deployment, a permanent ID is assigned to each user device (e.g., PC, PDA, mobile handset). This ID relates the device to a) its device sensors and b) the location in which it resides. For the location estimation of a device, several positioning methods can be used, either for indoor or outdoor environments [7]. By knowing where the device is located, the system can determine which area sensors are relevant to each user. Hence, only that particular subset of device and area sensors (hereinafter called *user-relevant sensors*) are monitored, and their readings are stored to a database table accessible by the A-module.

2.2 Adaptation through Abstract Policies

The rules that govern the reaction of A-module to sensor value changes are specified as Adaptation Policies (AP) expressed in an Adaptation Policy Language (APL), which we have defined. In particular, the system, during the adaptation of the requested page, checks the values of all user-relevant sensors against the rules specified through the AP. An AP is an XML document that assigns *groups of actions* to sensor ranges, with each action group containing one or more generic formatting commands (i.e., *actions*). Such generic commands are not bound to any target language, but are rather "rough instructions" for the final page layout. The AP is specified with abstract formatting commands instead of “target language” tags in order to support multimodality.

Sensors of the same type are grouped together by Chameleon, and a single value (e.g., the median of the individual sensor readings) is computed for each sensor group. We have decided to give priority to device sensors over area sensors, since the former are, in general, closer to the user and its device, and, thus, their readings are expected to be closer to the actual conditions of the user-device interaction. In Table 1 some indicative rules are described, where generic actions (e.g., LowContrast) are assigned to the various sensor value ranges. The actions fall into *attribute* categories which group together actions that affect similar aspects of formatting (i.e., “Size”, “Contrast”, “Volume”). This classification of abstract actions in Table 1 primarily serves the rule author of the system in organizing the rules.

The actual adaptation is performed through the mapping of abstract actions to CSS tags (see also Section 3.3). First, we create a table containing all CSS tags for each abstract action along with their default values (see Table 2).

Subsequently, we create the mappings between the abstract actions and the CSS tags (see Table 3). These mappings also specify the target language to which each transformation is referred, and a *relative modifier* that will be applied to the CSS tag value. For example, in the case of the action *LargeSize* the modifier applied to the

Table 1. Sample Adaptation Policy rules

Attribute / Sensor	Sensor Value	Contrast	Volume	Size
Luminance Sensor	0-5	LowContrast	-	MidSize
	5-7	-	-	-
	8-10	HighContrast	Loud	-
Velocity Sensor	0-3	-	Soft	SmallSize
	4-7	-	-	MidSize
	7-10	HighContrast	Loud	LargeSize

Table 2. Table of default formatting values

End Language	CSS Tag	Default Value
HTML	font-size	9
HTML	font-weight	400
VoiceXML	voice-volume	5
HTML	background-color	0xFFFFFF

Table 3. Mappings between AP actions and CSS tags

End Language	AP Action	CSS Tag	Modifier
HTML	LargeSize	font-size	+4
HTML	LargeSize	font-weight	+100
HTML	SmallSize	font-size	-2
VoiceXML	Soft	voice-volume	-2
VoiceXML	Loud	voice-volume	+3

CSS tag “font-size” is “+4”. We already know from Table 2 that the default value for the “font-size” tag is “9”. Therefore, if the system executes the *LargeSize* action, it will set a new value of “13” to “font-size”.

The modifiers, of course, have to be numbers so that they can be added/subtracted. However, there is a solution for non-numerical modifiers as well. The values of such modifiers can be represented as *enumerators*, assuming the relevant CSS tags have *ordered* values (e.g. small, medium, large). Hence, instead of manipulating tag values, we add/subtract enumerator values. The final value of the tag is the value of the enumerator that corresponds to the number resulting from the addition/subtraction of the relevant modifiers. For example, consider the CSS tag “border-width” which may take the values “thin”, ”medium” and “thick”. Since the values can be ordered, we represent them with the numerical values “1”, “2” and “3”, respectively. If the current tag value is “thin” and the following hypothetical rule applies, then the “border-width” will be set to “medium”:

If velocity > 0.3m/sec then “border-width” = “border-width” + 1

When transformation is complete, the system has produced a set of CSS tags that will be applied to the end page. These tags are integrated in the page as an internal CSS style (see Section 3.3). This process is repeated every time the value of at least one of the user-relevant sensors changes, thus, causing the system to “push” the new version of the page to the client. This automated update is performed through AJAX technology [18].

It should be noted a this point that the default values in Table 2 and the modifier values in Table 3 can be further personalized based on the user profile. This can be achieved by adding an extra element in the rule antecedents: the user class. The implementation of such personalization has not incorporated in the Chameleon prototype yet, but it is rather straightforward. All that is needed is a way to capture/extract the user profile (e.g., possible limitations regarding the perception of user interfaces) and to rewrite the adaptation rules by taking this profile into account.

Finally, similarly to every rule-/policy-based system, there may occur the situation that conflicting actions should be executed. In such case, two categories of conflicts can be identified:

1) *Rules that affect different presentation elements in conflicting ways*. For example, a rule that increases the text font size in a web page and another one that decreases the size of buttons. This type of conflict should be addressed by the AP author, who should create rules that affect all elements on a page and not just a subset of them.
2) *Rules that affect the same presentation element in conflicting ways*. In order to resolve such conflicts, the APs should be designed so that the rule actions do not result in contradictory formatting instructions. This last type of conflict is addressed by the system through the use of relative modifiers for the CCS tags, since the final value of the CSS tags is computed by adding/subtracting the values of all relevant modifiers.

3 Selected Implementation Issues

3.1 Wireless Sensor Network and Middleware

For the development of a system prototype, a limited WSN was used. The sensor nodes were the Crossbow Mica2 motes [9], carrying light, noise and temperature sensors (with only the first being exploited). Each mote was executing the TinyDB application (part of the TinyOS suite [20]), which enables the programmer to issue SQL-like requests against the sensor network. The readings of the sensors were periodically forwarded to the S-module, where the database table with the sensor values was updated accordingly.

3.2 Abstract User Interface Languages

Another key decision was to use XForms [15] as the abstract user interface language. The main reasons for adopting XForms are:

1. XForms is a W3C standard. Hence, it is well defined, widely supported and not likely to be abandoned in the near future.
2. XForms separates data from presentation. This, practically, means that the content author does not have to get involved with page presentation details.

A number of other languages were considered prior to adopting XForms. These are the User Interface Markup Language [4], the Extensible Interface Markup Language [19] and the Alternative Interface Access Protocol [17]. A number of reasons discouraged us from using the three above languages. For example, UIML is well-developed but does not separate data from presentation as strictly as XForms does. Moreover, both XIML and UIML require an abstract user interface vocabulary (a set of mappings between abstract and concrete elements) to be included in the page in order to render it. AIAP, on the other hand, does not impose the aforementioned restrictions, but is still under development from INCITS (InterNational Committee for Information Technology Standards) and is not yet a standard, thus limiting its potential support and usability.

3.3 Adaptation through Cascading Style Sheets

As already mentioned, the pages are formatted using Cascading Style Sheets (CSS). Apart from being a well-established standard, the recent developments in its specification offer versatility and interoperability with mobile devices (CSS Mobile Profile) [10] and voice-enabled devices (CSS3) [11]. CSS Mobile Profile offers a variety of formatting commands that are specific to mobile devices (e.g., cell phones). CSS3 includes a speech module, which enables auditory formatting of a rendered page. The CSS flavor used by our system is *internal CSS*. In internal CSS the formatting instructions are embedded in a HTML tag (<STYLE>) in the page header.

Regarding the CSS update procedure, changes in page formatting are performed on the client-side through the use of the AJAX [18] programming model. More specifically, the client, using a XMLHttpRequest object, periodically queries the system regarding sensor value changes. If such changes have occurred since the last check, the system responds with the new CSS stylesheet elements, which replace the old ones. This way, formatting is performed seamlessly (i.e., without discarding any user data).

4 Performance Evaluation

4.1 Evaluation Setup

In order to evaluate the performance of the system, and to assess its scalability, we have performed a series of tests. The metrics adopted for these tests are: a) the end-to-end (E2E) response time: the time between the issue of the request by the user and the end of reception of the response by her browser and b) the throughput of the web server (expressed in requests per minute, rpm).

The variable parameters are: a) the period of sensor checks. We test the system for 500ms, 1s, 2s and 5s periods of sensor checks and b) the number of concurrent users. We test the system for 1, 2, 5, 10, 20, and 50 concurrent users.

For each test, a number of threads equal to the number of concurrent users are created, and, subsequently, each thread requests a page from the system.

The evaluation was performed on a desktop PC with an AMD Athlon XP 2600+ CPU, 512MB of RAM, running Windows XP Professional Edition, and the Tomcat servlet container v.5.5 [14]. The client browser is the JMeter tool v2.0.3 [12], which adds virtually no overhead when compared to a typical HTML browser (e.g., Mozilla). The client runs on a laptop with an AMD Mobile Athlon XP 1.6 CPU and 256 MB of RAM, running Windows XP Home Edition.

The source XForms document has a size of 1687 bytes. The resulting HTML file has a size of about 2100 bytes (depending on the CSS tags used each time).

The test is further divided into test groups. Each group is characterized by a different *sensor check period*. For each group, the test measures the system response time and the web server throughput for different numbers of users and for a network connection speed of 11 Mbps (WLAN). This is can be considered a *stress test*, which simulates page updates for each sensor check. In other words, this is a worst-case scenario where all sensor checks lead to page updates.

In order to reduce the statistical error, a fairly large (around 500) number of samples (i.e. total number of requests) are collected in each measurement.

Table 4. Test setup matrix

Sensor check period	0.5, 1, 2, 5 seconds
Bandwidth	11 Mbps
Requested page size	~2 Kbytes
Page refresh	At every sensor check
# of users	1-50
# of samples	~500

4.2 Evaluation Results

From the performed tests, it can be readily deduced that the longer the sensor check period is, the more scalable the system is. This is due to the (expected) fact that if the sensors are checked less frequently, fewer requests per minute are issued to the server, and, thus, the latter is capable of handling more users concurrently.

The **maximum throughput** (see Fig. 3) the system has achieved is slightly over 400 requests per minute (rpm). With a 5-second sensor check period, the system serves a single user at a rate of 12 rpm and scales well up to 40 users with 400 rpm. With lower periods, the system is saturated earlier, (20 users for 2s, 10-20 users for 1s, 5-10 users for 500ms); this is reflected, among other things, in the increased response times.

The **E2E response times** follow the same trend for the whole range of sensor check periods (see Fig. 4). While increasing with the number of users in a roughly linear fashion, that increase becomes sharp after the 20-user mark (less sharp for a 500ms sensor update period). It can be easily seen that, in general, a longer sensor check period reduces the response time.

The main conclusion is that, since the system needs roughly 130ms to process and reply to each individual request, the rate at which the requests arrive to the system should be less than that, i.e. roughly 460 rpm. This is close to the observed maximum throughput of 410 rpm. Thus, the more users the system has to support, the longer the sensor check periods must be so as to avoid server saturation and delays. For 1-5 users, the sensor check period can be as low as 500 ms, but for 50 concurrent users, the period must be at least 5s.

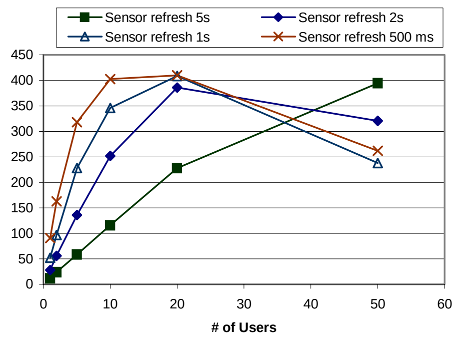

Fig. 3. Server Throughput (requests/min)

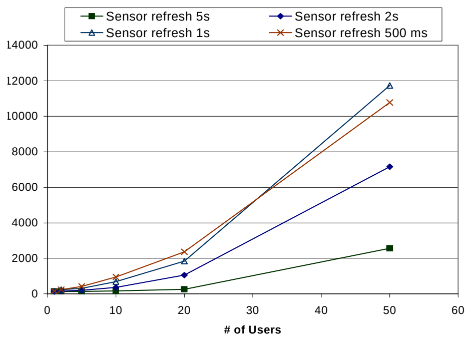

Fig. 4. Average E2E Response Time (msec)

5 Related Work

The presented system, although supporting multimodality of user interfaces, focuses on *presentation adaptation*. To our knowledge, this is a new concept in adaptive systems. In fact, we are not aware of any work sharing our objectives. The most relevant research fields are content adaptation, based either on device capabilities or user context, and multimodal interfaces. In the following paragraphs we briefly present such efforts.

The field of adaptive and context-aware systems is currently undergoing intensive research. However, most of these research efforts are focusing on the adaptation of the served content itself, rather than of its presentation. Examples of content adaptation can be found in [1], [2], and [5].

Additionally, there is active research in multimodal systems, which transform an interface written in a generalized, abstract language to a concrete user interface language, appropriate for the client device. Indicative tools in this field are [3] and [13]. Another proposal, presented in [6], describes a system that selects the categories of the content that will be presented to the client (text, pictures or sound), based on the capabilities of her device.

6 Conclusions

In this paper we have presented Chameleon, an innovative system for dynamically adapting Web user interfaces upon changes in the user's environmental conditions. Key enabler for the system is the wireless sensor network that allows the system functionality to be provided to mobile users. The implementation of the system with established technologies and the performed evaluation showed that the concept of "context-aware presentation" is quite promising and could be incorporated in next generation products and services. In fact, as the sensing technologies advance and are embedded in end-user devices, we expect more research efforts towards this direction. Such efforts should also address user evaluation issues as well as develop a theoretical basis for context-driven presentation adaptation. For instance, the required expressiveness of the adaptation rules is still an open issue. Research in uncertainty reasoning, fuzzy logic and defeasible reasoning may provide some useful modeling and reasoning tools for designing more advanced system behaviors.

Among our planned research tasks, are the integration of multimodal support (i.e., target languages such as WML and VoiceXML), apart from the currently supported HTML, and the evaluation of the system with real users.

Acknowledgement. This work was partially supported by the Greek General Secretariat for Research and Technology (GSRT) under the framework PENED 2003 (research grant 03ED173).

References

[1] Billsus, D., Brunk, C. A.,Evans, C., Gladish, B., and Pazzani, M.: Adaptive Interfaces for Ubiquitous Web Access. Communications of the ACM, 45(5) (2002) 34-38

[2] Cheverst, K., Mitchell, K., Davies, N.: The role of Adaptive Hypermedia in a Context-Aware Tourist GUIDE. Communications of the ACM, 45(5) (2002) 47-51

[3] Mayora-Ibarra, O., Cambranes-Martínez, E., de la Paz-Arroyo, O., Fuentes-Penna, A.: A Visual Programming Environment for Device Independent Generation of User Interfaces. Proc. 1st Latin American Conf. on Human-Computer interaction (2003), Brazil

[4] Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S., and Shuster, J., UIML: An Appliance-Independent XML User Interface Language. WWW'8 (1999), Toronto

[5] Samulowitz, M., Michahelles, F., Linnhoff-Popien, C.: Adaptive Interaction for Enabling Pervasive Services. Proc. 2nd ACM Int'l workshop Data Eng. for wireless and mobile access (MobiDE) (2001), Santa Barbara, CA., U.S.A.

[6] Yew, A., Liotta, A., Yang, K.: Adaptive Content for the Mobile User: A Policy-Based Approach. 2nd Int'l Workshop on Mobility Aware Technologies and Applications (2005), Montreal, Canada.

[7] Schiller, J., Voisard, A.: Location-Based Services. Morgan Kauffman (2004)

[8] Anagnostopoulos, C., Tsounis, A., Hadjiefthymiades, S.: Context Awareness in Mobile Computing: A Survey. Mobile and Ubiquitous Information Access Workshop (2004), UK

[9] Crossbow. Wireless Sensor Networks. http://www.xbow.com

[10] Wugofski, T., et al.: CSS Mobile Profile 1.0. W3C Candidate Recommendation (2002)

[11] Meyer, E.A., Bos, B: Introduction to CSS3, W3C Working Draft, (2001)

[12] Apache Software Foundation: Apache JMeter. http://jakarta.apache.org/jmeter/

[13] Harmonia Inc: LiquidUI Tool. http://www.harmonia.com/index.html

[14] Apache Software Foundation: Apache Tomcat. http://tomcat.apache.org/

[15] Dubinko, M., Klotz, L., Merrick, T., Raman, T. V.: XForms 1.0, W3C Recommendation, (2003)

[16] Clark, J.: XSL Transformations (XSLT), Version 1.0. W3C Recommendation,(1999)

[17] NCITS V2 - Standards Development Committee on Information Technology Access Interfaces. www.ncits.org/tc_home/v2htm/V2docs/v201011.htm.

[18] Zakas, N. C., McPeak, J., Fawcett, J.: Professional Ajax. Wrox Press (2006)

[19] Puerta, A., Eisenstein, J.: XIML: a common representation for interaction data. 7th international conference on Intelligent User Interfaces. ACM Press (2002)

[20] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for network sensors. ASPLOS (2000)