4

Content Adaptation in Ubiquitous Computing[1]

Wanderley Lopes de Souza[1], Antonio Francisco do Prado[1],
Marcos Forte[2] and Carlos Eduardo Cirilo[1]
[1]*Federal University of São Carlos*
[2]*Federal University of São Paulo*
Brazil

1. Introduction

According to the predominant computing environments, the history of Computing can be classified into the initial period of mainframes, the current one of personal computers, and the future one of Ubiquitous Computing whose goal is to provide the user with easy access to and processing of information at any time and from anywhere (Hansmann et al., 2003).
Mobile communication has contributed to drive the leap of Computing into this new era, since it has given users unprecedented choice and freedom, enabling them to search for new and rewarding ways to conduct their personal and professional affairs. In just one decade, mobile networks have allowed for a growth rate that took fixed networks almost a century to achieve, and the advances in mobile technologies have led to the transition from voice-exclusive services to web-based content services.
This globalized mobility requires new architectures and protocols that allow mobile networks to connect easily to several types of services and content providers spread over the Internet. The futuristic view of the mobile Internet presupposes users with different profiles using different access networks and mobile devices, requiring personalized services that meet their needs, availability and locations. In this context, it is necessary to describe information about people, places, devices and other objects that are considered relevant for the interaction between users and services, including the users and services themselves.
The fields of Ubiquitous Computing include content adaptation, which involves converting an original content into a large number of formats compatible with the user preferences, the access device capabilities, the access network characteristics, and the delivery context. Due to the infinity of possible adaptations, the greater the quantity of available adaptation services, the higher the chances of meeting the user's needs.
The content adaptation can occurs at several points along the data path, including the origin server, the user device, and the edge device. An essential requirement for carrying out this process is the establishment of an adaptation policy, which defines what adaptation is to be done on a given content, when, and who should do it. To be effective, this policy must take into account information on users, devices, access network, content, and service agreement.
The purpose of this Ubiquitous Computing book chapter is to do a survey on our main contributions in the field of content adaptation. The sequence of this chapter is organized as

[1] Supported by INCT-MACC/ CNPq

follows: section 2 deals with content adaptation, the Internet Content Adaptation Protocol (ICAP), and an adaptation service that uses this protocol; section 3 deals with frameworks for content adaptation, in particular the Internet Content Adaptation Framework (ICAF); section 4 deals with ontologies and Web services for content adaptation, and an ICAF extension for the use of these technologies; and section 5 presents some concluding remarks.

2. Content adaptation

Content adaptation involves modifying the representation of Internet content in order to come up with versions that meet diverse user requirements and the distinct characteristics of devices and access networks (Buchholz & Schill, 2003). Among the Internet content adaptation services the following stand out (Beck et al., 2000):

a. *Virus scan*, searches for viruses before delivering the content to the user;
b. *Ad Insertion*, inserts advertisements into a content based on user interests and/ or location;
c. *Markup Language Translation*, allows devices that do not support Hypertext Markup Language (HTML) pages but support other markup languages (e.g., Wireless Markup Language) to receive the content of such pages;
d. *Data compression*, allows the origin server to send its content in compressed form so that the edge device can extract it, thereby reducing the bandwidth used in this communication;
e. *Content Filtering*, redirects an unauthorized content request or blocks a response containing unsuitable content;
f. *Image Transcodification*, processes image files in order, for example, to transform its format, reduce its size and/ or resolution, or modify its color range; and
g. *Language Translate*, translates a Web page from one language to another.

These adaptation services require from an adaptation server special processing functions, such as video and voice trans-coding, intelligent text processing and filtering, and many others. Performing the adaptation services at the origin server has the advantage that the content author has a full control on what and how to present the content to the user. On the other hand, since these functions are quite different from the basic functions needed for building Web servers, the authoring process becomes more complex and time consuming. Another major drawback is the cost and the performance scalability issue of the origin server. Performing the adaptation services at the end user device limits the adaptation to the available functionality and capability of the device. It is therefore recommended to locate these functions in a separate computer that could be shared by many different applications.

2.1 Internet Content Adaptation Protocol (ICAP)

ICAP (Elson & Cerpa, 2003) was first introduced in 1999 by Peter Danzig and John Schuster from Network Appliance and further developed by the ICAP Forum, a coalition of Internet businesses. ICAP is a client/ server application protocol running on the top of TCP/ IP, similar in semantics and usage to HTTP/ 1.1, and designed to off-load specific Internet-based content adaptation to dedicated servers. Each server may focus on a specific value added service, thereby freeing up resources in the Web servers and standardizing the way in which these services are implemented. At the core of this process there is an ICAP client that intercepts HTTP messages and transmits them to the ICAP server for processing. The ICAP server executes its adaptation service on these messages and sends them back to the ICAP client. ICAP can be used in two modes as shown in Figure 1: request modification mode (*reqmode*) and response modification mode (*respmode*).

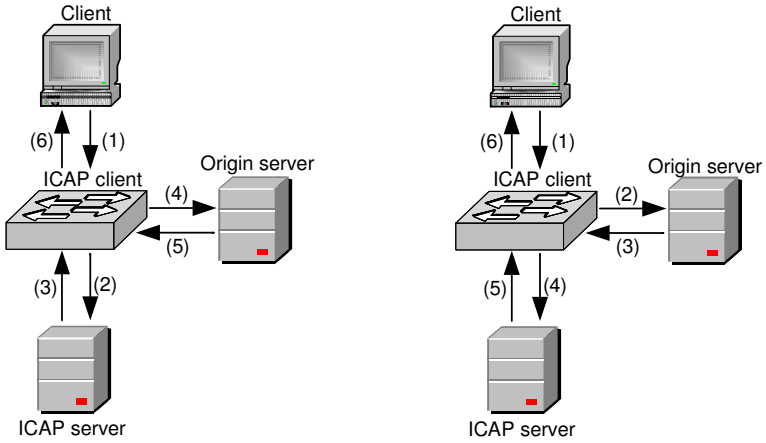


Fig. 1. (a) ICAP *reqmode* (b) ICAP *respmode*

In the *reqmode*, a user Client sends a request to an Origin server. This request is intercepted by an ICAP client, which redirects it to an ICAP server. The ICAP server may then: (a) send back a modified version of the request containing the original URI, and the ICAP client may then send the modified request to the Origin server, or may first pipeline it to another ICAP server for further modification; (b) modify the request so that it points to a page containing an error message instead of the original URI; (c) return an encapsulated HTTP response indicating an HTTP error. Figure 1(a) shows a data flow, where the message sequences in cases (a) and (b) are 1, 2, 3, 4, 5, and 6, while the message sequence in case (c) is 1, 2, 3, and 6. The response modification mode (*respmode*) is intended for post-processing performed on a HTTP response before it is delivered to the user client. The ICAP client forwards the request directly to the Origin server. The Origin server's response is intercepted by the ICAP client, which redirects it to an ICAP server. The ICAP server may then: (a) send back a modified version of the response; or (b) return an error. Figure 1(b) shows a data flow for this case.
Although it is an essential part, the transaction semantics defined by ICAP is of limited use without a control algorithm, that determines what adaptation or processing function should be requested for what HTTP request or response passing through the ICAP client.

2.2 Adaptation policy

One fundamental aspect in content adaptation is the definition of an adaptation policy, i.e., what adaptation services are to be offered, which local or remote adaptors will execute these adaptations, and when the latter should be requested. The following information is necessary regarding the adaptation environment: characteristics and capacities of the access device; personal user information and preferences; conditions of the communication network; characteristics of the requested content; and the terms of the service agreement between the service provider and the end user. As proposed in (Forte et al., 2006) and illustrated in Figure 2, this information can be described and stored in *device*, *user*, *network*, *content* and *Service Level Agreement (SLA)* profiles.

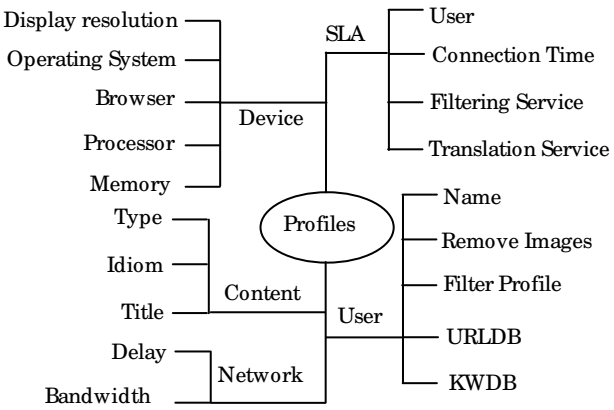


Fig. 2. Profiles and attributes

The user profile contains the user's personal information and his content adaptation preferences. Different users may wish to have different adaptations applied to a requested content (e.g., one user prefers having images removed while another prefers the sound). Adaptations not based on user preferences may be inconvenient or even undesirable.
The network profile can be dynamically obtained through agents that monitor parameters of the communication network between provider and user. Parameters such as latency and bandwidth guide some adaptation processes (e.g., images, video and audio on demand) so that the adapted content is optimized for the conditions of the network of a given context.
The content profile, also generated dynamically, is based on characteristics of the requested content. The applicable content modifications are determined based on information extracted from the HTTP header and, if it is available, the set of content metadata.
The SLA profile contains the terms of the service agreement between the user and the access provider, which can offer different plans, including bandwidth, connection time and added value services, allowing users to choose the plan that best fits their needs.
The adaptation policy must also consider adaptation rules, which comprise a set of related conditions and actions. These conditions refer to the profiles, reflecting the characteristics and needs of the entities involved in the adaptation process, and determine the action to be taken and the adaptation servers to be used. Figure 3 shows execution points for adaptation rules: points 1 and 2 during the request stage, the former before the content search in the cache and the latter after this search; points 3 and 4 during the response stage, the former before the content storage in the cache and the latter after this storage. The execution point definition for each rule depends on the adaptation service (e.g., an antivirus service should be executed at point 3 to prevent a contaminated content from being stored in the cache).

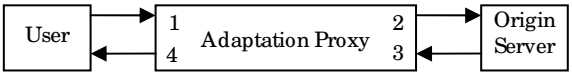


Fig. 3. Execution points for processing adaptation rules

2.3 Content Classification and Filtering Service (CCFS)

CCFS allows for controlling the access to undesirable content. Three interrelated terms are defined for this kind of service: (a) *labeling* is the process that describes a content associated

to a label without requiring the user to open the file to examine its content; (b) *rating* is the process that confers values on a content based on certain suppositions/ criteria, and if the content has a label, it already possesses a prequalification which may or may not be accepted by the filter; (c) *filtering* is the process aimed at blocking access to a content by comparing its classification against the system's definition of undesirable content. It should be noted that CCFS is not restricted solely to illegal (e.g., racism) or inappropriate contents (e.g., pornography), but also to undesirable contents in a corporation (e.g., shopping, chats).
The oldest and most commonly employed classification method is based on proprietary *Uniform Resource Locator (URL)* collections, in which each URL is associated to a specific content category. When a page is requested, the classifier checks its address in the database to find its category. With the category definition the filter can block or release the access to the site, according to the configured Internet policy. Keeping these collections updated is a challenge for CCFS suppliers, since the rate at which new Internet pages are created far exceeds their capacity to classify them (ICOGNITO, 2002).
A second generation of classifiers executes the analysis and classification of all Web traffic requested by the user on demand (e.g. keywords, textual analysis, labels, image analysis).When a page is received, it is classified according to its content, and the system blocks or releases that page in line with the pre-established filtering policy. The classification process is subject to the following problems: (a) *under-blocking*, when the filter fails to block undesired content, usually due to an outdated *URL* database or, in the dynamic approaches, an incorrect content classification; (b) *over-blocking*, when the filter blocks a content unduly, usually due to the use of keywords without context analysis. Pages on sexual education and medicine are the most commonly affected by this last problem (Rideout et al., 2002).
The CCFS we developed (Forte et al., 2006), illustrated in Figure 4, is part of a general architecture encompassing a set of dedicated adaptation servers and a content adaptation proxy. The purpose is to allow access to the available Internet content, independently of the device the user is employing, and to adapt the content according to the user's preferences.

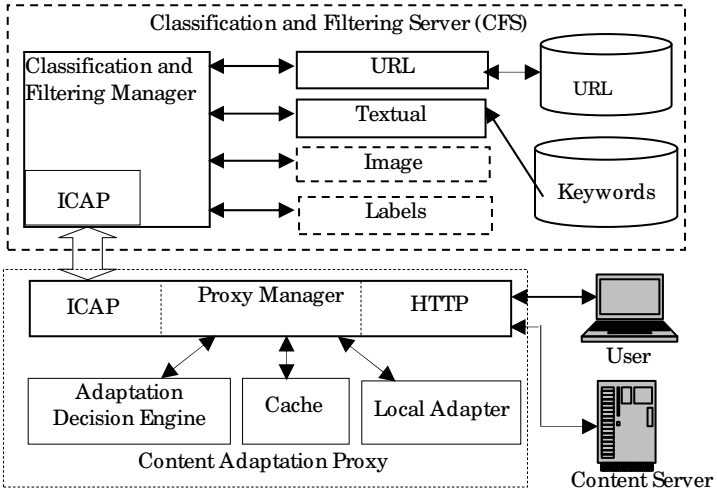


Fig. 4. CCFS general architecture

This architecture is based on the client-server model, in which the proxy captures the user's requests and the content server's responses. The adaptation decision mechanism implements the adaptation policy. If the adaptation policy defines the need for the CCFS, the proxy will send an ICAP request to the Classification and Filtering Server (CFS).
The CFS was designed to allow for the easy integration of new Classification modules. The Classification and Filtering Management module manages the Classification modules and, using ICAP and including the ICAP and HTTP headers, manages the communications with the Content Adaptation Proxy. It also filters the content based on the information sent by the Classification modules. The profiles were implemented using the *Composite Capability /Preference Profile (CC/PP)* (W3C, 2004a). Figure 5 shows a fragment of the user profile.

```
<rdf:Description rdf:ID="UserProfile">
<ccpp:component>
    <rdf:Description rdf:ID="Identification">
        <usr:UserName>mlobato</usr:UserName>
        <usr:Gender>Male</usr:Gender>
        <usr:Age>21</usr:Age>
    </rdf:Description>
</ccpp:component>
<ccpp:component>
    <rdf:Description rdf:ID="Preferences">
        <usr:ImageGrayScale>0</usr:ImageGrayScale>
        <usr:Filter_Profile>001</usr:Filter_Profile>
        <usr:UrlDB>001</usr:UrlDB>
        <usr:KWDB>005</usr:KWDB>
    </rdf:Description>  ….
```

Fig. 5. Fragment of the user profile

The adaptation rules were implemented as clauses stored in a database, and they use the Prolog inference mechanism to deduce the actions to be taken as a function of the conditions to be met. Each adaptation executing point is represented at the rules base by a different functor, allowing the rules of a given executing point to be processed. Figure 6 illustrates an example of adaptation rule implementation to be invoked at the executing point 3.

```
Point_three(Ret,UserID.DeviceID,SLAID,Content):-
        contentIsText(Content),
        userPayforFilter(ContractID),
        =(Ret,' content-filter.com filter').
```

Fig. 6. Example of adaptation rule

In this adaptation rule the user identification (*UserID*), device identification (*DeviceID*), service level agreement (*SLAID*), and content type (*Content*) are provided. If they met, the action is stored in the variable *Ret*, which receives the values of the *contentfilter.com* and *filter*. Figure 7 illustrates the sequence of a content adaptation. Starting from an HTTP request from the user (1), the access provider sends the HTTP request to the adaptation proxy together with the user identification (2). The adaptation decision mechanism pulls the user profile and SLA from its database and verifies that the user chose a filtering service, which

needs the requested content and the content profile. Failing to locate this content in its cache, the proxy sends a request to the origin content server (3) and, upon receiving a response (4), dynamically creates the content profile. Since the requested content is of text type, all the requisites of CFS's rule are met. Then, the decision mechanism creates an ICAP request, attaching the user's preferences (e.g., icap:/ / adaptation.com/ filter?*filter_profile=001 &urldb=001&kwdb=005&append*) and the content received from the Web server, and sends them through the proxy to the adaptation server (5). The latter executes the requested adaptation and returns the result to the proxy (6), which in turn sends it to the user (7).

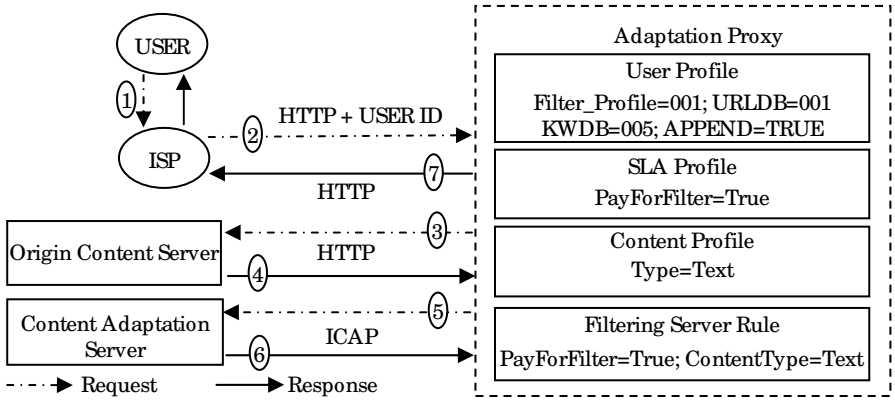


Fig. 7. Sequence of a content adaptation

An ICAP request encapsulates the ICAP header, the HTTP request header, the HTTP response header and the requested page body, the last two only when operating in *respmod*. When this request reaches the CFS, the following information is extracted from the ICAP header: the categories of content to be blocked (e.g., *filter_profile=001*); the URL databases and categorized domains that will be used (e.g., *urldb=001*); and, if the adaptation is in *respmod*, the database of keywords (e.g., *kwdb=005*). The domain and URL page requested by the user are extracted from the HTTP request header. In *respmod* the requested content will be extracted (*body*), allowing for classification by keywords. Figure 8 shows the states model of this ICAP request.

2.4 CCFS evaluation

For the CCFS performance evaluation three computers were employed: the first executing Linux Fedora Core 2 (2Ghz – 256MB) and the others Windows 2000 (700Mhz – 256MB). Because the variations in the response times of the Origin Servers, including those due to the Internet throughput, could interfere on this evaluation, an Apache 2 server was installed. This enabled the tested pages to be cloned, restricting the data flow to the computers involved in the case study. The software described in (Forte et al., 2007) was employed for the content adaptation proxy implementation, with the addition of specific CCFS profiles and rules, and was installed on the Linux platform. The CFS and its database, containing 651,620 categorized sites and 29 keywords, were installed on a Windows 2000 platform. Five predefined links have been accessed: two links were not blocked by the filter, one was blocked because of its address domain and another because of its URL address, and the last

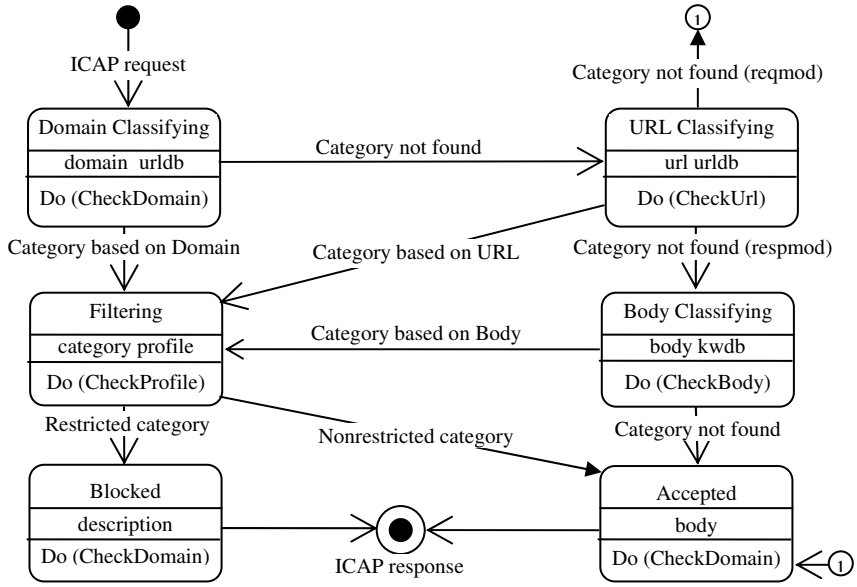


Fig. 8. States model of an ICAP request for CCFS

was blocked due to a restricted keyword. The load was progressively increased, adding one user every 1.5s up to the limit of 200 users, each user accessing a link every 5s. Figure 9 shows the results of these scenarios. The difference between the response times in *reqmod* and *respmod* is due to the addition of the content classification routine based on keywords.

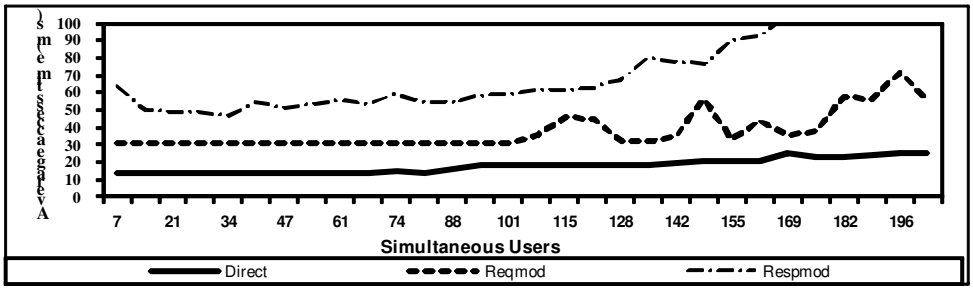
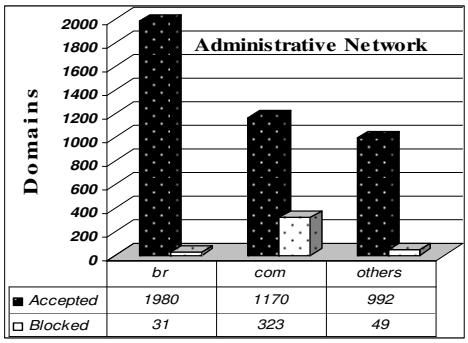


Fig. 9. Average response time *versus* number of users

For the CCFS efficiency evaluation the CFS and the Squid proxy (SourceForge, 2002) were used in a Brazilian university administrative network, and in an informatics laboratory of this university. During 48 hours 1,215,854 requests (6.8 GB) from the administrative network and 409,428 requests (2.8 GB) from the laboratories' network were checked. To minimize the interference on the response time of the employees' and students' accesses, only the *reqmod* was used by the adaptation server. Figure 10 depicts the results of this evaluation.



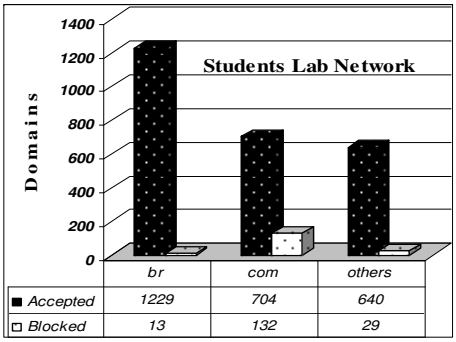


Fig. 10. CCFS efficiency evaluation

Considering all the domains classified in the two networks, 3 were incorrectly classified in restricted categories (*over-blocking*), and 22 were not classified passing incorrectly through the filter (*under-blocking*). Among those that passed incorrectly through the filter, 19 belonged to .br domains, which demonstrates the lower efficiency of *blacklists* produced in other countries. These 22 domains were tested again, with CCFS operating in *respmod* and using keywords for the content analysis, and 16 were correctly categorized.

3. Frameworks for content adaptation

In the mid-90s most of the content adaptations were done in the proxy (Bharadvaj et al., 1998; Smith et al., 1998). Since this approach tends to overload the proxy, services networks were developed for intercepting the content delivery and adapting this content, and these networks were mainly based on the Open Pluggable Edge Services (OPES) model (Tomlinson et al., 2001). Since OPES distributes adaptations among dedicated servers, it became feasible to build a single architecture offering several types of adaptation.

The OPES WG also developed the languages Intermediary Rule Markup Language (IRML) and P for the specification of content adaptation rules. IRML is based on eXtensible Markup Language (XML), and was designed to express service execution policies and to reflect the interests of the origin server and user on a content transaction (Beck & Hofmann, 2003). P is based on Smalltalk and C++, it is interpreted and has the following qualitative aspects: exactness, flexibility, efficiency, simplicity, security and hardiness (Beck & Rousskov, 2003).

Several requirements must be considered when offering adaptation services. For instance, to avoid overloading the proxy in a service network, it is important to distribute the adaptation services among dedicated servers. Based on such requirements, the ideas of service networks, and using the OPES model, some important works have been done.

(Beck & Hofmann, 2001) presents an architecture for executing content adaptations that contains a decision-making mechanism based on a condition set. These conditions and related actions constitute the adaptation rules, which are specified in IRML. However, these conditions do not employ information related to the adaptation environment. (Ravindran et al., 2002) proposes a framework to manage personalization of services, whose adaptation policy is based on a combination of user preferences, device constraints, and content characteristics. (Marques & Loureiro, 2004) presents an adaptation architecture specifically designed for mobile devices, which offers image, audio, and text compression adaptations, and uses access network information to decide the best adaptation to be carried out.

Unlike the aforementioned works, we developed the Internet Content Adaptation Framework (ICAF) (Forte et al., 2007), an extremely flexible framework thanks to the reuse of its components. Moreover, this framework contains an adaptation policy based on the user, device and SLA profiles, and based on the access network and content information.

3.1 Internet Content Adaptation Framework (ICAF)

ICAF has two main packages: Adaptation Proxy and Adaptation Server. The User and Origin Server actors interact with the Adaptation Proxy through content requests and responses. The Adaptation Proxy analyzes these interactions based on an adaptation policy and uses the available services at the Adaptation Server, which in turn adapts the transmitted content. Figure 11 illustrates the components of the Adaptation Proxy package. *Content Transfer Protocol* is responsible for the communications between User and Origin Server. It is generic enough to support most of the used protocols for content transmission on the Internet, including: Hypertext Transfer Protocol (HTTP), Real-time Transport Protocol (RTP), Simple Object Access Protocol (SOAP), and File Transfer Protocol (FTP).

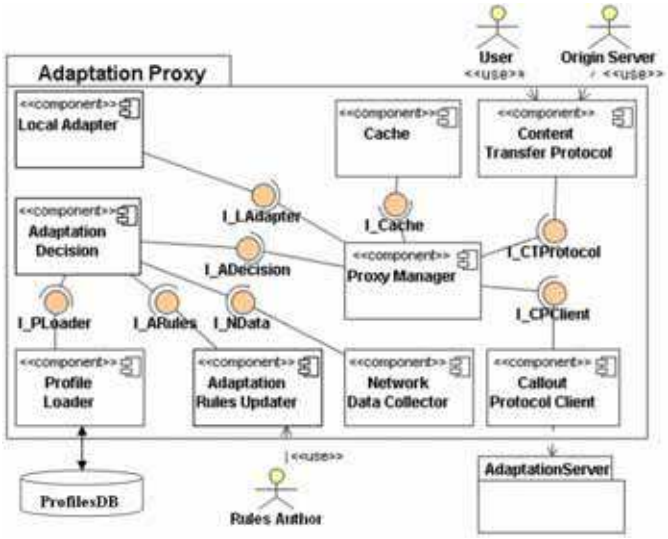


Fig. 11. Components of the Adaptation Proxy package

Since the Adaptation Proxy requests remote adaptations to an Adaptation Server using communication protocols, the *Callout Protocol Client* was defined for doing the remote adaptation calls. This component supports different protocols, including those especially created for this communication: ICAP, and OPES Callout Protocol (OCP) (Rousskov, 2005).
Cache was built to improve the ICAF's performance. It temporarily stores requested contents on the Internet, as well as Web pages, videos and images. Before requesting content to the Origin Server, the Adaptation Proxy checks whether it is already in the *Cache*. If so, a request to the Origin Server is avoided, speeding up the content delivery to the User.
Although it uses adaptation service modules, ICAF can do content adaptations locally via the *Local Adapter*. As this component employs the proxy resources, its wide-scale use may affect the Adaptation Proxy performance, and delay the content delivery to the User.

The Adaptation Proxy data flow is controlled by the *Proxy Manager*, which receives, through the *Content Transfer Protocol*, the User requests and the Origin Sever responses. Using the information carried in these communication primitives, the *Proxy Manager* asks for an adaptation analysis to the *Adaptation Decision*. If adaptation is needed the *Proxy Manager* invokes locally or remotely this service. To avoid an unnecessary content request, the *Proxy Manager* checks if this content is already in the *Cache*.

The ICAF's adaptation policy takes into account the user's interests and preferences, device capabilities and constraints, access network conditions, content characteristics, and SLA for building the adaptation rules. It is implemented through the *Adaptation Decision*, *Profile Loader*, *Adaptation Rules Updater* and *Network Data Collector* components.

The *Profile Loader* gets the user, device and SLA profiles in the ProfilesDB database. For inserting these profiles in this database an Internet interface must be available, which could be a Web page with a form where the user fills out the data related to these profiles. The SLA profile could be inserted in the same way by the system administrator.

The ICAF policy is controlled by the *Adaptation Decision* that receives: the access network conditions from the *Network Data Collector*, which monitors the network parameters; the profiles stored in the ProfilesDB from the *Profile Loader*; and the content to be analyzed from the *Proxy Manager*. Based on this information set, the *Adaptation Decision* uses the adaptations rules to decide what adaptations will be done, which servers (local and/ or remote) will execute these adaptations, and the order in which they will be executed.

To prevent processing of outdated rules, the *Adaptation Rules Updater* inserts, removes and updates the adaptation rules implemented in the *Adaptation Decision*. The Rules Author carries out these updates using an interface provided by the *Adaptation Rules Updater*.

The Content Adaptation Server is responsible for doing the content adaptations requested by the Adaptation Proxy. Since different types of adaptation services can be offered, we adopted the Component-Based Development approach to define a generic structure for the Adaptation Servers, which includes the components illustrated in Figure 12. The *Callout Protocol Server* handles the communications with the Adaptation Proxy, supporting several protocols (e.g., ICAP, HTTP), analyzes the adaptation requests received from the *Callout Protocol Client*, and defines the action to be taken and its parameters. Based on this information, which is sent in the header of the appropriated protocol message, the *Callout Protocol Server* asks for the requested adaptation to the *Remote Adapter*.

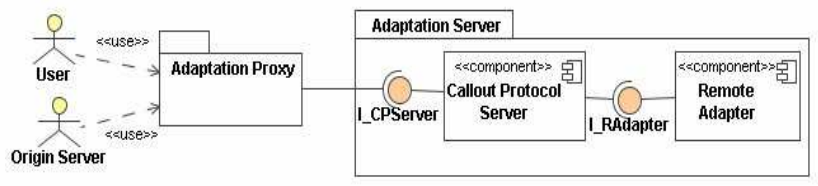


Fig. 12. Components of the Adaptation Server package

The *Remote Adapter* executes the content adaptations, its internal structure can vary according to the type of adaptation offered by the Adaptation Server, and it carries out the same functions of the *Local Adapter* but does not use the processing resources of the Adaptation Proxy. The decision-making process of adapting the content locally or remotely must consider the ratio between the execution times of the Callout Protocol and the content

adaptation. The lower the ratio the more the decision will be in favor of the remote adaptation. A design example of the *Remote Adapter* was presented in section 2.3.
ICAF was built to support the same ICAP operating modes and the same execution points defined in section 2.2. It was also developed to offer a basic structure for creating Internet content adaptation applications through the reuse of its components.

3.2 ICAF reuse and evaluation

Figure 13 shows the components used in a case study, where the following components were reused by ICAF direct instantiation: *Local Adapter*, *Remote Adapter*, *Proxy Manager*, *Cache*, *Adaptation Rules Updater*, *Network Data Collector*, and *Profile Loader*.
ICAF's adaptation policy takes into account adaptation rules and information related to the adaptation environment. Since there are several ways of implementing this policy, including by a procedural language algorithm, several ICAP components were customized and new components were added to this framework.

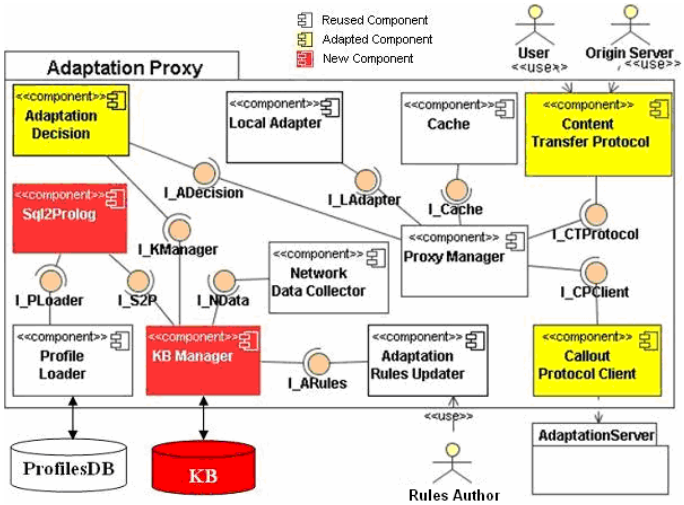


Fig. 13. ICAF reuse example

The *Adaptation Decision* is responsible for the decision-making process of the ICAF's adaptation policy. For this case study, this component was customized to perform with an inference mechanism, based on adaptation rules and environment's information, for defining the adaptation services to be executed. The *Adaptation Decision* decides the adapter (local or/ and remote) that will perform an adaptation and, if multiple adaptations are required, decides the sequence of them. The inference mechanism was introduced through a Prolog Knowledge Base (KB) for giving more flexibility to the ICAF´s adaptation policy, and for giving some "intelligence" to the decision-making process.
To handle KB, the *KB Manager* was added to ICAF. It receives the updated adaptation rules from *Adaptation Rules Updater*, translates these rules to Prolog, stores this translation in KB and, when requested by the *Adaptation Decision*, carries out a query for retrieving information on users, devices and access network. KB answers this query based on the user, device and SLA profiles, and based on the access network and content information. Since

KB is implemented in Prolog and ProfilesDB in SQL, the *SQL2Prolog* component was added to ICAF for enabling KB to receive and analyze profiles stored in ProfilesDB.
Three components were customized with specific interfaces, characterizing the reuse through specialization. *Content Transfer Protocol* was specialized for transmitting content requests and responses through HTTP, which consists of two fields: the header and the payload (i.e., content). The content is modified by an HTTP parser, which identifies the semantic actions of this protocol. *Callout Protocol Client* and *Callout Protocol Server* were specialized to allow for ICAP communication between the Adaptation Proxy and the Adaptation Server. Upon receiving a service request, *Callout Protocol Client* encapsulates the actions, parameters and content in an ICAP request and sends it to *Callout Protocol Server*. This latter component retrieves the information for doing a semantic analysis of this request, and asks the requested service to the *Remote Adapter*. After receiving the adapted content from the *Remote Adapter*, *Callout Protocol Server* encapsulates this content in an ICAP response for returning it to the Adaptation Proxy.
ICAF was developed having in mind the definition of an adaptation policy with a short processing time, and the offer of adaptation services without degradation of the Internet's infrastructure performance. In this case study, these requirements were evaluated measuring the adaptation policy and content adaptation execution times on a network with five computers: one Adaptation Proxy, three Adaptation Servers and one User. The Adaptation Proxy implementation was based on (SourceForge, 2003), the implementations of the Image Adapter (IA) and Virus Scan (VS) servers were based on (Network, 2001), and the implementation of the Content Filter (CF) server was based on the CFS presented in section 2.3. The Origin Servers were accessed directly from Internet content servers.
For the performance evaluation of this case study, the model described in (Mastoli et al, 2003) was employed, and the temporal collecting points T_0 to T_7 were defined for measuring: the origin server response time T(Origin Server); the processing time of the adaptation policy T(Analysis); the adaptation time consumed by the ICAP protocol and by the content adaptations T(Adaptation); and the delivery time of the content to the user after executing all adaptations T(Delivery). Figure 14 depicts these points and measuring times.

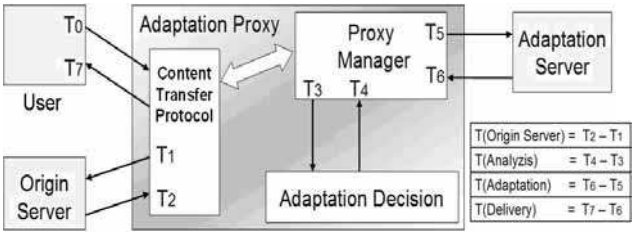


Fig. 14. Temporal collecting points and measuring times

For this evaluation 1,000 requests for www.folha.com.br were executed with five different kinds of adaptation on this Web page: no adaptation (NA), using each one of the Adaptation Servers (VS, IA, CF) independently, and combining these servers (VS+IA+CF). Figure 15 shows the T(Origin Server), T(Analysis), T(Adaptation) and T(Delivery) average times.
The origin server response times are by far the longest ones, and the adaptation policy processing times are similar for all adaptations. Without adaptation (NA) it was consumed 121 ms, corresponding to 103 ms of origin server response time, 17 ms of adaptation policy

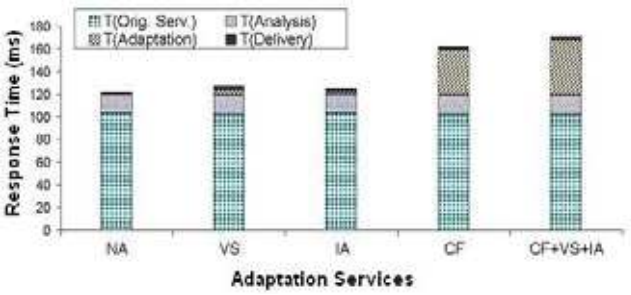


Fig. 15. Measuring times *versus* adaptation services

processing time, and only 1 ms of delivery time. Therefore, the relatively short delay (17ms) introduced by the inference mechanism can be considered satisfactory. With the VS and IA servers the adaptation times were 5.3 ms and 2.5 ms respectively, and the delays introduced by these servers can be also considered satisfactory. However, with the CF server the adaptation time was 40 ms, and the delay introduced by this server is relevant.

4. Ontologies and Web services for content adaptation

To achieve interoperability among heterogeneous systems executing applications of a given domain, it is essential to be able to share information, with a common and unambiguous understanding of the terms and concepts used by these applications. In this context, ontologies are important artifacts for making feasible the treatment of this heterogeneity.

Berners-Lee proposed the Semantic Web (Berners-Lee et al., 2001) as an evolution of the traditional Web to allow for the manipulation of content by applications with the capacity to interpret the semantics of information. The Web content can thus be interpreted by machines through the use of ontologies, rendering the retrieval of information from the Web less ambiguous and providing more precise responses to user requests.

The World Wide Web Consortium (W3C) guides the development, organization and standardization of languages to promote interoperability among Web applications. These languages include the Resource Description Framework (RDF) (W3C, 2004b), and the Ontology Web Language (OWL) (W3C, 2004c).

OWL is a markup language used for publication and sharing of ontologies in the Web. In this language, ontology is a set of definitions of classes, properties and restrictions relating to the modes in which these classes and properties can be used.

Web services have been for years the basis of service-oriented architectures, but begun to show deficiencies for service description, discovery, and composition due to the lack of semantic support in Web Services Description Language (WSDL) (W3C, 2007), and in the mechanism of storage and discovery services of Universal Description Discovery and Integration (UDDI) (UDDI Spec TC, 2004). To integrate semantic Web to Web services the Ontology Web Language for Services (OWL-S) (Martin et al., 2006) was developed.

OWL-S allows for the discovery, composition, invocation and monitoring of services, it has a larger number of Application Programming Interfaces (APIs), and inherits tools from OWL and from the Semantic Web Rule Language (SWRL) (Horrocks et al., 2003). OWL-S combines elements of WSDL, OWL's semantic markup and a language for rules description (e.g., SWRL). The OWL-S model is composed of three parts: Service Model for describing

how a Web service operates; Service Grounding for describing the access to a Web service, and Service Profile for describing what the Web service does.
Service Model specifies the communication protocol, telling what information the requester must send to or receive from the service provider at a given moment of the transaction. This module distinguishes two types of processes: atomic and composite. The first one supply abstract specifications of the information exchanged with the requester, corresponding to operations the supplier can directly execute. The latter is employed to describe collections of processes (atomic or composite) organized through some type of flow control structure.
Service Grounding describes how atomic processes are transformed into concrete messages, which are exchanged via a network or through a procedure call. A "one-on-one" mapping of atomic processes for WSDL specifications is defined.
Service Profile is a high level specification of the service provider and service functionalities, including: contact information of a provider/ service (e.g., *serviceName*, *textDescription*, *contactInformation*); categorization attributes of the offered service (e.g., *serviceParameter*, *serviceCategory*); and service functional representations in the form of *Inputs*, *Outputs*, *Preconditions*, and *Effects* (IOPEs). IOPEs are described by the properties *hasParameter*, *hasInput*, *hasOutput*, *hasPrecondition*, and *hasEffect*.
Since descriptions of OWL-S services are based on OWL, the OWL domain model can be employed to structure the service descriptions, facilitating the reuse of OWL ontologies already developed. In this sense, we extended ICAF for allowing the use of ontologies and Web services in the development of content adaptation applications (Forte et al., 2008).

4.1 Adaptation policy specification with ontologies and Web services

One major challenge in Ubiquitous Computing is the description of the delivery context, which is as a set of attributes that characterizes aspects related to the delivery of Web content. For content adaptation the delivery context must contain even more information that can be described in a set of profiles. For the CCFS development, presented in section 2, these profiles were implemented using CC/ PP, and the adaptation rules were implemented as clauses stored in a database and the Prolog inference mechanism was employed. In this section we propose to specify the same profiles in OWL, and to employ semantics in the adaptation rules description for facilitating their extension and the addition of new rules.
To make available an infrastructure of adaptation servers over the Internet, we propose to use Web service technology, since it offers a large number of tools, and well-defined standards. Moreover, the use of ontologies and the inclusion of semantics in these standards help the migration from the proprietary solutions, for the discovery and composition of services, to an open distributed architecture based on the semantic Web.
The following information about the adaptation servers is essential: characteristics; communication needs (e.g., protocols, addressing); and the conditions, for the execution of their services, which are described by the adaptation rules. We propose to make available this information via the adaptation server profile and to specify it in OWL, and the services information via the service profile and to specify it in OWL-S.
All ontology models for the OWL profiles are based on the EMF Ontology Definition Metamodel (EODM) (IBM, 2004), which is derived from the OMG's Ontology Definition Metamodel (ODM) and implemented in the Eclipse Modeling Framework (EMF). These models use the following OWL components: *Classes* that are the basic building blocks of an OWL ontology; *Individuals* that are instances of classes; *Object properties* to relate individuals

to other individuals; and *Datatype properties* to relate individuals to data type, such as integers, floats, and strings. OWL supports six main types of classes: *named*, *intersection*, *union*, *complement*, *restrictions,* and *enumerated*.

The *UserAgent* field of the HTTP header is employed to identify the user's access device and to look up the device profile stored in the database. Figure 16 depicts an ontology model for a device profile, describing the following characteristics: supported image formats (*Supported_image* class); display information, including resolution (*Display_Resolution* class) and colors (*Color* class); supported audio and video streaming (*Streaming* class); markup languages supported by the device's browser (*Supported_Markup* class), including their properties (*WML_UI*, *XHTML_UI* and *CHTML_UI* classes); model and manufacturer (*Product_Info* class); and security (*Security* class). One OWL characteristic present in this model is the restriction insertion that helps the consistency checking and the validation of this profile. For instance, *Supported_ImageRestriction* defines that the class *Supported_Image* can only be instantiated with the individuals declared in the enumerated class *Image_Format*.

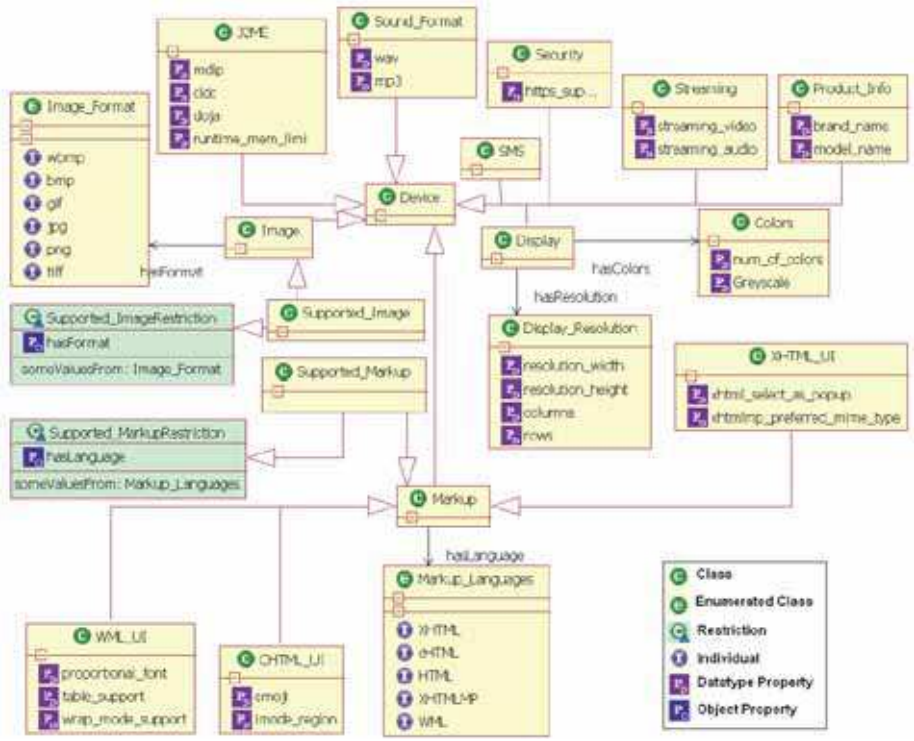


Fig. 16. Ontology model for a device profile

Figure 17 depicts an ontology model for a user profile. This model represents the *User* class that encloses the *Info* and *Service_Preferences* subclasses. *Info* holds the following properties: *ID* for retrieving the user's information from the database, and *FirstName*/*LastName* for identifying the user's name. *Service_Preferences* contains the specifications of the adaptation service properties, which can be configured by the user according to his/ her preferences.

ID_Required defines that the class *Person* has exactly one value for the property *ID* inside *Info*. Three adaptation services are represented in this model: *Antivirus*, where the user can choose to check or not viruses using a script language on a Web page; *Image_Adapter*, where the user can define the action to be taken if a low throughput is detected (e.g., color and/ or resolution reductions, conversion to black and white); *Classification_and_Filtering*, where the user can define the content types to be blocked (e.g., sex, shopping, games) and which URLs (*URLDB*), databases, and keywords (*KWDB*) will be used for the classification.

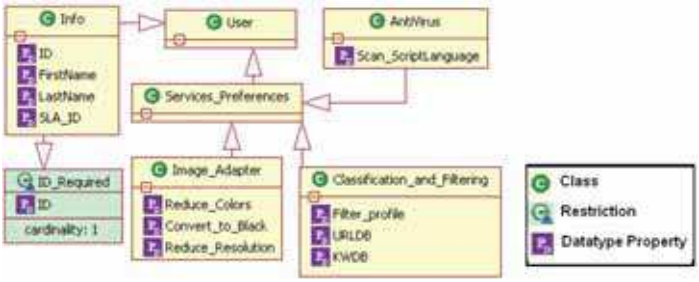


Fig. 17. Ontology model for a user profile

The network profile is built dynamically using the agents that monitor network parameters. The information in this profile is used to guide some adaptation processes (e.g., images, video and audio on demand), for the optimization of the content adaptation as a function of the network current conditions. Figure 18 depicts an ontology model for a network profile.

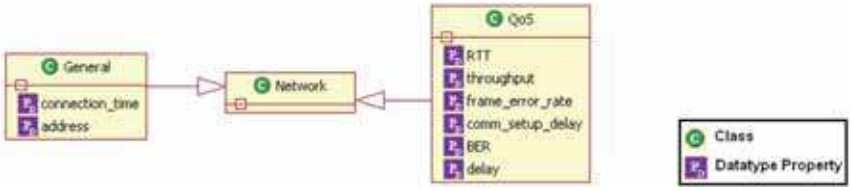


Fig. 18. Ontology model for a network profile

The content profile is built dynamically using the characteristics of the requested content. The needed and applicable adaptations to the content depend on information extracted from the HTTP header (e.g., the content has text and/ or image, language) and depend on content metadata, if available. Figure 19 depicts an ontology model for a content profile.

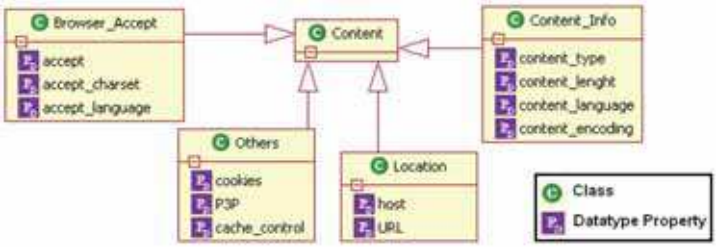


Fig. 19. Ontology model for a content profile

The terms of the agreement between the access provider and the user are described in the SLA profile. These providers offer to their users a variety of plans, including bandwidth, connection time and several added value services, enabling the user to choose the plan that best meets his/ her needs. Figure 20 depicts an ontology model for a SLA profile.

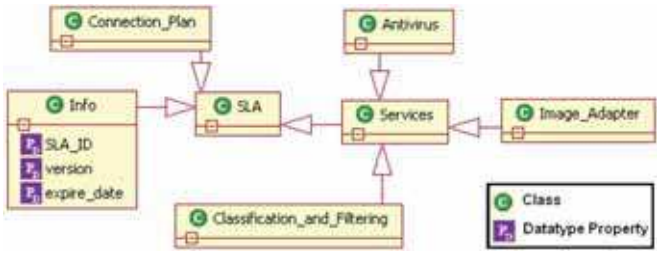


Fig. 20. Ontology model for a SLA profile

Some characteristics described in the adaptation server profile are related to the Quality of Service (*QoS*), including the server's *Availability* and *Reliability*, the last characteristic for evaluating the successful execution rate. This information, allied to *MaxProcessTime*, *RequiredBandwidth* and *Cost*, helps the decision-making process when the service discovery finds more than one service provider satisfying the delivery context needs. This profile also defines the communication protocol between the proxy and the adaptation server. Figure 21 shows an ontology model for an adaptation server profile, where *Supported_Execution_Points* specifies four execution points for processing adaptation rules, with exactly the same meaning found in section 2.2. To help in consistency checking and validation of this profile, it is inserted *Supported_ProtocolsRestriction*, restricting *Supported_Protocols* to be instantiated with individuals declared in *Protocols*, and *Supported_Execution_PointsRestriction*, restricting *Supported_ExecutionPoints* to be instantiated with individuals declared in *Exec_Points*.

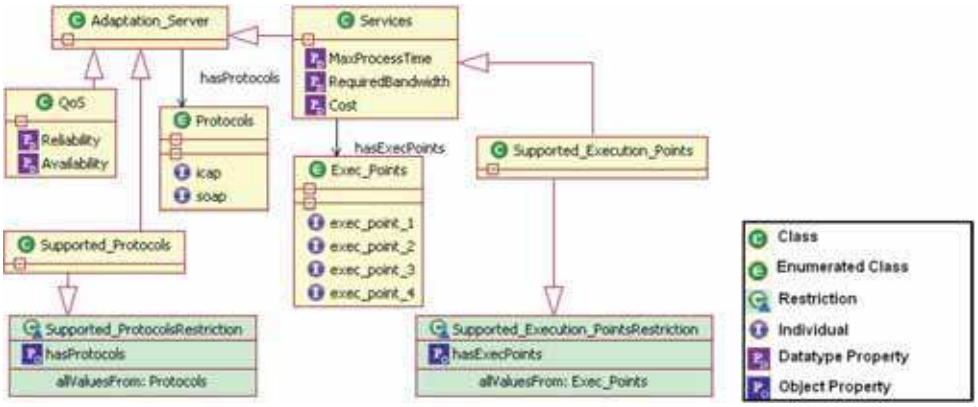


Fig. 21. Ontology model for an adaptation server profile

Service characteristics, including the necessary conditions for its execution, can be associated to the adaptation process, represented by the *Inputs* and *Outputs*, or associated to the service, represented by the *Preconditions* and *Effects*, which help the adaptation policy deciding whether or not to execute a given service. Figure 22 depicts these associations.

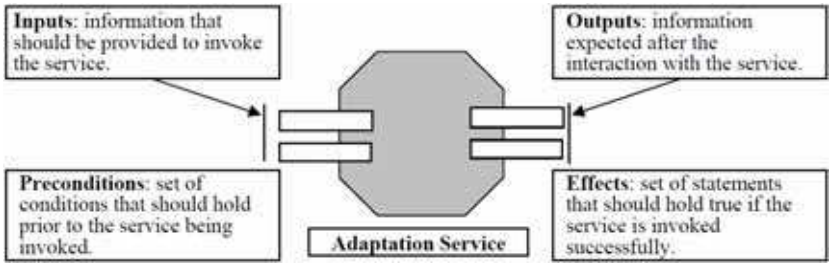


Fig. 22. Associations of service characteristics

The service profile has the service characteristics and some exclusive functions that must be specified in OWL-S. The main one is the mapping of OWL semantic specifications of other profiles into WSDL syntactic specifications, allowing for the integration of ontologies and Web services. The service profile also imports semantic specifications contained in other profiles or other OWL ontologies to facilitate their reuse and to avoid ambiguities.

Figure 23 shows the OWL-S specification skeleton of a Markup Language Translation Service (MLTS) profile. The Ontologies, to be imported to include the semantic information needed for this service, are defined, the public ontology *semwebglossary.owl*, which specifies terms of the semantic Web for avoiding ambiguous definitions, the *content.owl* profile, and the *device.owl* profile are reused (1). *SupportedMarkupLanguage* class (2) and *canBeConvertedTo* property (3), which will be used for defining the possible conversions to be carried out by the service (4), are defined. The *InputMarkupLanguage* parameter indicates the markup language to be converted (5), which is obtained from the *Content_Type* class of the content profile, previously instantiated with information about the requested content. The *OutputMarkupLanguage* parameter indicates the desired markup language that is defined in the *Supported_Markup* class of the device profile, which in turn informs the supported languages (6). The inputs, outputs, preconditions and effects are defined in the process *MarkupConverterProcess* (7). The precondition *SupportedTranslation* is based on a SWRL rule (8), which returns *true* when the needed translation belongs to the list defined in (4).

If it is not found a Web service that meets all the needs of a given adaptation, those needs may be met by an appropriate composition of Web services, which should be identified. Let $S = \{S_1, S_2, ..., S_n\}$ be a Web service set, where each S_i is defined by a quadruple

$$\left(I_s^i, O_s^i, P_s^i, E_s^i\right) \tag{1}$$

with its elements representing the sets of *Inputs*, *Outputs*, *Preconditions* and *Effects* of the service S_i. Let $G = \{G_1, G_2, ..., G_m\}$ be a goals set, where each goal G_j is defined by a quadruple

$$\left(I_g^j, O_g^j, P_g^j, E_g^j\right) \tag{2}$$

with its elements representing the sets of *Inputs*, *Outputs*, *Preconditions* and *Effects* required by the goal G_j. If there is at least one S_i that satisfies

$$\bigcup_{i=1}^{n} I_s^i \subseteq \bigcup_{j=1}^{m} I_g^j \text{ and } \bigcup_{i=1}^{n} P_s^i \subseteq \bigcup_{j=1}^{m} P_g^j \text{ and} \tag{3}$$

$$\bigcup_{i=1}^{n} O_s^i \supseteq \bigcup_{j=1}^{m} O_g^j \text{ and } \bigcup_{i=1}^{n} E_s^i \supseteq \bigcup_{j=1}^{m} E_g^j \quad (4)$$

```
...
<!ENTITY gloss "http://www.personal-reader.de/rdf/semwebglossary.owl">
<!ENTITY device "http://www.adaptationsrv.org/device.owl">
<!ENTITY content "http://www.adaptationsrv.org/content.owl">
...
<owl:Class rdf:ID="SupportedMarkupLanguage">
  <owl:oneOf rdf:parseType="Collection">
     <factbook:Language rdf:about="&gloss;#HTML"/>
     <factbook:Language rdf:about="&gloss;#XHTML"/>
     <factbook:Language rdf:about="&gloss;#XML"/>
     <factbook:Language rdf:about="&gloss;#CHTML"/>
     <factbook:Language rdf:about="&gloss;#WML"/>
  </owl:oneOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="canBeConvertedTo">
  <rdfs:domain rdf:resource="#SupportedMarkupLanguage"/>
  <rdfs:range rdf:resource="#SupportedMarkupLanguage"/>
</owl:ObjectProperty>
<rdf:Description rdf:about="&gloss;#HTML"><canBeConvertedTo rdf:resource="&gloss;#WML"/></rdf:Description>
<rdf:Description rdf:about="&gloss;#HTML"><canBeConvertedTo rdf:resource="&gloss;#XHTML"/></rdf:Description>
<rdf:Description rdf:about="&gloss;#HTML"><canBeConvertedTo rdf:resource="&gloss;#XML"/></rdf:Description>
<rdf:Description rdf:about="&gloss;#HTML"><canBeConvertedTo rdf:resource="&gloss;#cHTML"/></rdf:Description>
<rdf:Description rdf:about="&gloss;#XML"><canBeConvertedTo rdf:resource="&gloss;#XHTML"/></rdf:Description>
...
<process:Input rdf:ID="InputMarkupLanguage">
 <process:parameterType rdf:datatype="&xsd;#anyURI">&content;#Content_Type
 </process:parameterType>
</process:Input>
<process:Input rdf:ID="OutputMarkupLanguage">
  <process:parameterType rdf:datatype="&xsd;#anyURI">&device;#Supported_Markup
  </process:parameterType>
</process:Input>
...
<process:AtomicProcess rdf:ID="MarkupConverterProcess">
 <process:hasInput rdf:resource="#InputMarkupLanguage"/>
 <process:hasInput rdf:resource="#OutputMarkupLanguage"/>
 <process:hasInput rdf:resource="#InputString"/>
 <process:hasOutput rdf:resource="#OutputString"/>
 <process:hasPrecondition rdf:resource="#SupportedTranslation"/>
 <process:hasEffect rdf:resource= ="#MarkupLanguageConverted"/>
</process:AtomicProcess>
...
<expr:SWRL-Condition rdf:ID="SupportedTranslation">
  <rdfs:label>canBeConvertedTo(InputMarkupLanguage, OutputMarkupLanguage)</rdfs:label>
  <expr:expressionLanguage rdf:resource="&expr;#SWRL"/>
  <expr:expressionObject><swrl:AtomList><rdf:first>
        <swrl:IndividualPropertyAtom>
          <swrl:propertyPredicate rdf:resource="#canBeConvertedTo"/>
          <swrl:argument1 rdf:resource="#InputMarkupLanguage"/>
          <swrl:argument2 rdf:resource="#OutputMarkupLanguage"/>
        </swrl:IndividualPropertyAtom>
    </rdf:first></swrl:AtomList></expr:expressionObject>
</expr:SWRL-Condition>
...
```

Fig. 23. OWL-S specification skeleton of a MLTS profile

then S_i is put in the services list that provides the required content adaptation. Otherwise, if there is at least one S_i that satisfies (3), there is no service to carry out alone the adaptation, the composition algorithm is activated. Otherwise, the adaptation cannot be carried out.
One of the techniques used for the services composition is *forward chaining* (Lara et al., 2005). Starting from a goal G_j, where an S_i is found that satisfies (3), a new goal is defined

$$G_k = \left(I_g^k, O_g^k, P_g^k, E_g^k\right) \tag{5}$$

where

$$I_g^k = O_s^i \text{ and } P_g^k = E_s^i \tag{6}$$

This process is repeated until (4) is satisfied, otherwise the adaptation cannot be carried out.

4.2 Extended Internet Content Adaptation Framework (EICAF)

ICAF was extended to deal with ontologies and Web services. Figure 24 shows the EICAF component model, where the following ICAF components were reused: *Local Adapter, Cache, Content Transfer Protocol, Network Data Collector* and *Remote Adapter*. The components *Callout Protocol (Client* and *Server)*, *Proxy Manager* and *Service Profile Updater* were modified to include new interfaces and functionalities for using semantic profiles and new protocols. *Matching and Composition, Ontologies Manager* and *Reasoner* are new components, introduced for managing the ontologies and for processing the adaptation policy.
Callout Protocol Client and *Callout Protocol Server* support ICAP and SOAP. The first, already supported by ICAF, was kept to allow for communication with the servers previously developed. The second was introduced to provide compatibility with Web services.

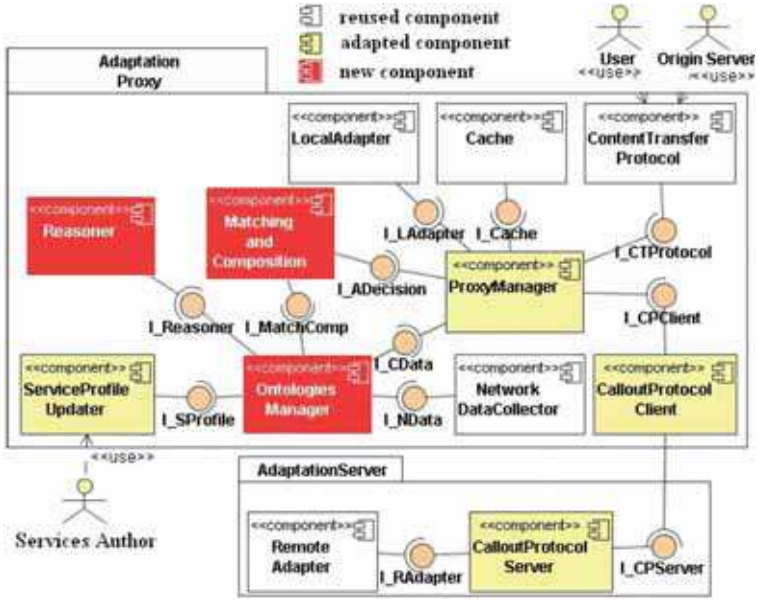


Fig. 24. EICAF component model

Ontologies Manager handles the storage, retrieval and processing of the static ontologies (device, user, SLA, adaptation server and service profiles) that are stored in a relational database. This component also treats the service profiles that are inserted through *Service Profile Updater*. The stored dynamic ontologies are generated on demand, from the information obtained by the *I_NData* interface, for the network profile generation, and by the *I_Cdata* interface, for the content profile generation.
To help the addition of new service profiles, *Service Profile Updater* provides a Web interface, where the service author can insert information related to the adaptation services to be converted to an OWL-S specification. Figure 25 shows some pages of this interface.

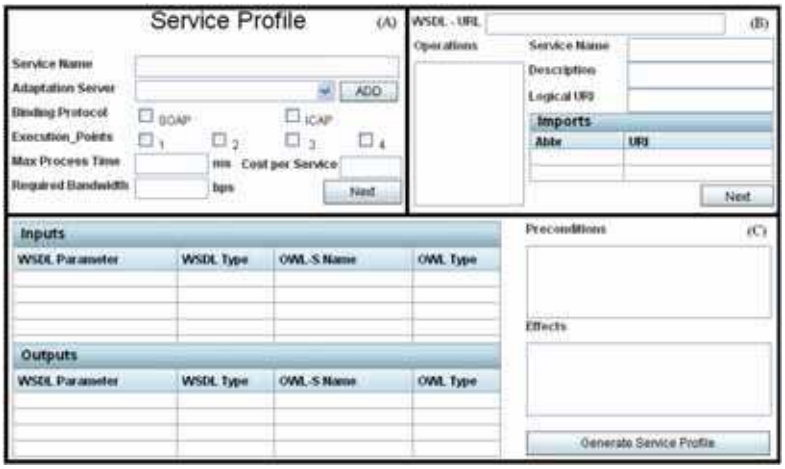


Fig. 25. Web interface for the insertion of service profiles

Initial page (A) is for insertion of service general information, mainly on the used protocol and the execution points for invoking the service: if ICAP is used, then this will be the only page available; if SOAP is used, then page (B) will be loaded. The operations are listed on this page, starting with the URI insertion of service WSDL specification, whose information will be converted in an OWL-S Service Grounding. Information about service profile URI is also inserted on page (B), and the ontologies to be imported are defined. On page (C) the mapping table of OWL semantic descriptions for WSDL parameters is filled out with the parameters inserted from information kept in WSDL file. The SWRL rules are specified in the *Preconditions* and *Effects* text boxes, and *Generate Service Profile* generates this profile.
EICAF was implemented in Java using several public APIs. The API OWL-S and the relational database MySQL were used in *Ontologies Manager*. The API Pellet was used in *Reasoner*, and the SPARQL module of the API OWL-S was used in *Matching and Composition*. This component makes requests based on SPARQL, using information contained in profiles and information inferred by the *Reasoner*, to locate the services that have inputs and outputs compatible with the delivery context. The results of these requests are filtered to check their conformity to preconditions and effects and, if a services composition is needed, *Matching and Composition* will manage the execution order of them. Next, this component sends to *Proxy Manager*, via the *I_ADecision* interface, the information regarding the services to be invoked. *Proxy Manager* forwards this information, via the *I_CPClient* interface, to *Callout*

Protocol Client that asks the services to the appropriated *Adaptation Servers*, which in turn execute these services. Last, *Proxy Manager* sends the adapted content to the *User*.
Figure 26 illustrates the tasks of the Web services composition mechanism by means of a delivery context example in which the user accesses the Web via a mobile phone. The notation *profile.class – context information* indicates the profile name, the class name, and the context information to be stored in this class. Since the user hired CCFS, the *Pay_for_Filtering* property of the SLA profile was configured as *true*. Once this profile information has been collected, the initial goal G_1 = *(HTML, XHTML, Pay_ for_Filtering, _)* is established, and the search for services to carry out this service is started. *Matching and Composition* discovers that the CCFS S_1 = *(HTML, HTML, Pay_for_Filtering, _)* meets the *Inputs* and *Preconditions*, but does not support the *Output XHTML*. Since the goal G_1 was only partially achieved, a new goal G_2 = *(HTML, XHTML, _ , _)* is established. A new search discovers that the MLTS S_2 = *(HTML, XHTML, _ , _)* meets completely this new goal, ending this composition process.

User.Filter_Profile – 001	**Network**.WAN – GPRS
SLA.Services – Pay_for_Filtering	**Network**.RTT – 10
Device.DisplayResolution – 320x200	**Content**.Type – HTML
Device.Supported_Markup – XHTML	**Content**.URL – www.test.com

Fig. 26. Example of a delivery context

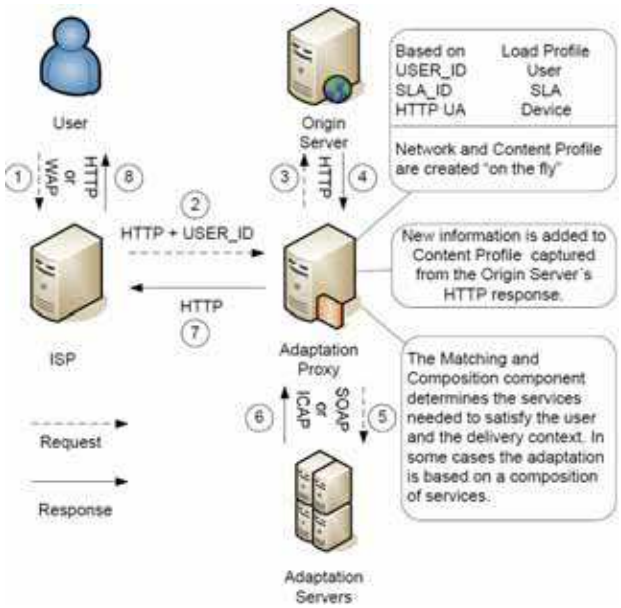


Fig. 27. Execution sequence of a content adaptation

Figure 27 illustrates the use of profiles and rules by means of an execution sequence of a content adaptation. Based on a user HTTP or WAP request (1), Internet Service Provider (ISP) sends the HTTP request to the Adaptation Proxy with the User identification (2). The *Ontology Manager* of the Adaptation Proxy employs the *USER_ID* for retrieving the user's

profile, the *SLA_ID* of the user's profile for retrieving the SLA profile, and the *HTTP User_Agent* for retrieving the device profile. The need for executing a content adaptation on the user's request is verified (*exec_point_1* and *exec_point_2* of the adaptation server profile). If so, the required services for this adaptation are executed (5)(6). If the requested content is not in the *Cache* of the Adaptation Proxy, it sends a request to the Origin Server (3), which responds with the content (4). *Ontology Manager* extracts information from this content to complete the content profile. It is verified if the content of the Origin Server response needs to be adapted (*exec_point_3* and *exec_point_4* of the adaptation server profile). If so, the required services for this adaptation are executed (5)(6). Lastly, the duly adapted content is sent to the ISP (7), which forwards it to the User (8).

4.3 EICAF evaluation

A case study was conducted involving the configuration illustrated in Figure 28: computers (1) and (2) were configured with Windows XP (700Mhz/ 256MB, 3Ghz/ 1GB) and computer (3) with the Linux Fedora Core 2 (2Ghz/ 256MB). The Apache JMeter was installed in computer (1) to simulate user access, and the Adaptation Proxy with the CCFS and MTS profiles was installed in computer (2). Because the variations in the response times of the Origin Servers, including those due to the Internet throughput, could interfere on this evaluation, an Apache 2 server with the virtual hosts resource was installed in computer (3). This enabled the tested pages to be cloned, restricting the data flow to the computers involved in the case study. A Tomcat/ Axis server with the respective *Remote Adapters*, which played the role of Adaptation Server, was also installed in computer (3).

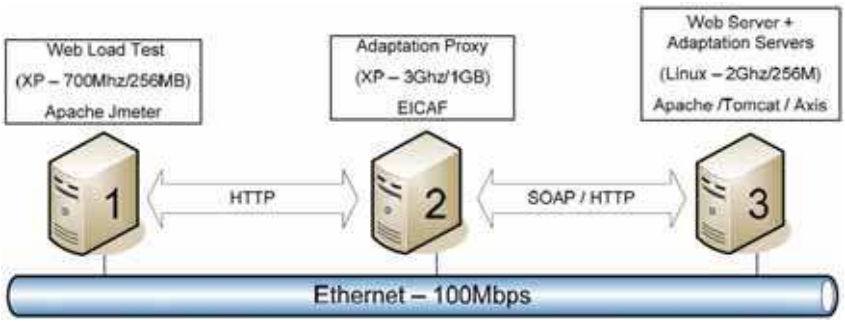


Fig. 28. Configuration in the case study

Five predefined links have been accessed for 5 minutes with three of these links blocked by the domain addresses. They were accessed using desktops and mobile phones, distributed randomly among the users, so that in some cases the MLTS would have to be used. The load was progressively increased, adding one user every 1.5s up to the limit of 20 users, each user accessing a link every 5s. To simulate different devices, 20 samplers were configured with the JMeter, each containing a different User-Agent.

Two test scenarios were defined: using ontologies, and carrying out the discovery and composition services on demand, an average response time of 425 ms was obtained; without using ontologies, and defining the discovery and composition services manually, an average response time of 38 ms was achieved. The difference of 387 ms between these two average response times can be attributed: 38.4% to the OWL-S API initialization, including all its

dependencies (e.g., Jena); 25.6% to the ontologies load and validation, corresponding to the dynamic and static profiles; and 36% to the matching and composition function. These values were obtained with the IDE Netbeans profiler tool. Figure 29 indicates an increase in response time and in processing demand caused by the use of ontologies in EICAF.

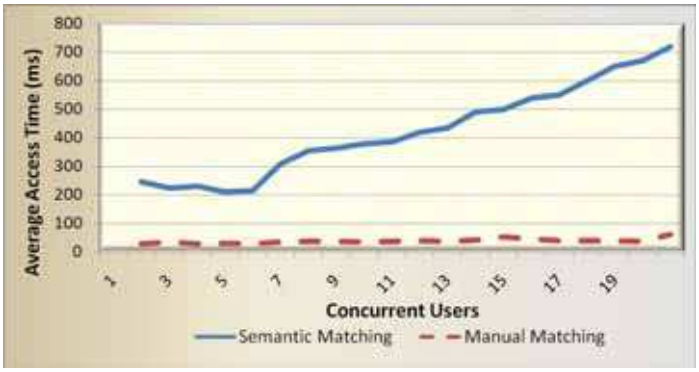


Fig. 29. Average Response Time *versus* Number of Users

From a qualitative standpoint, should be considered the enhanced flexibility and precision resulting from the use of semantics in the discovery and composition mechanism that aids the service authorship, and gives the user a high degree of satisfaction. It should also be considered that this response time tends to decrease with the semantic APIs improvement.

5. Conclusion

Applications involving multimedia usually require a certain level of QoS and, if this quality cannot be guaranteed over the Internet, these applications should be able to adapt to the available quality. ICAP defines the syntax and semantic for exchanging the content between the ICAP client and the ICAP server, but it does not define any adaptation policy. In this sense, we proposed a general architecture that considers a set of elements described in a set of profiles, which affect the quality of the content delivered to the user, for controlling the adaptation functions provided by the ICAP server. This architecture was employed in the development of a CCFS. We also proposed ICAF, a component-based framework that provides a flexible infrastructure for the Internet content adaptation domain, and we described an experiment reusing this framework. Finally, we proposed the use of ontologies and Web services to facilitate the development of content adaptation applications. To this end, ICAF was extended to allow for the use of ontologies and Web services, existing components were reused and adapted, and new components were introduced, thus broadening the range of applications that can be developed from EICAF.

6. References

Beck, A.; Hofmann, M. & Condry, M. (2000). Example Services for Network Edge Proxies, *Internet-Draft*, The Internet Society,

http:/ / standards.nortelnetworks.com/ opes/ non-wg-doc/ draft-beck-opes-esfnep-01.txt

Beck, A.; Hofmann, M (2001). Enabling the Internet to Deliver Content-Oriented Services, In: *Web Caching and Content Deliver*, A. Bestavros & M. Rabinovich (Eds), pp. 109-124, Elsevier, ISBN 0-444-50950-X, The Netherlands

Beck, A. & Hofmann, M. (2003). IRML: A Rule Specification Language for Intermediary Services, *Internet-Draft*, The Internet Society, http:/ / tools.ietf.org/ html/ draft-beck-opes-irml-03

Beck, A. & Rousskov, A. (2003). P: Message Processing Language, *OPES Internet-Draft*, The Internet Society, http:/ / tools.ietf.org/ html/ draft-ietf-opes-rules-p-00

Berners-Lee, T.; Hendler, J. & Lassila, O. (2001). The Semantic Web. *Scientific American*, May 2001, http:/ / citeseerx.ist.psu.edu/ viewdoc/ download?doi=10.1.1.115.9584&rep=rep1&type=pdf

Bharadvaj, H.; Joshi, A. & Auephanwiriyakul, S. (1998). An Active Transcoding Proxy to Support Mobile Web Access, *Proceedings of the 17th Symposium on Reliable Distributed Systems*, pp. 118-123, ISBN 0-8186-9218-9, West Lafayette-IN (USA), October 1998, IEEE Computer Society, USA

Buchholz, S. & Schill, A. (2003). Adaptation-Aware Web Caching: Caching in the Future Pervasive Web, *Proceedings of the 13th GI/ITG Conference Kommunikation in verteilten Systemen (KiVS)*, pp. 55-66, ISBN 3-540-00365-7, Leipzig (Germany), Februray 2003, Springer-Verlag, Germany

Forte, M.; Souza, W. L. & Prado, A. F. (2006). A content classification and filtering server for the Internet, *Proceedings of the 21st Annual ACM Symposium on Applied Computing*, Vol. 2, pp. 1166 – 1171, ISBN 1-59593-108-2, Dijon (France), April 2006, ACM, USA

Forte, M.; Claudino, R. A. T.; Souza, W. L.; Prado, A. F. & Santana, L. H. Z. (2007). A Component-Based Framework for the Internet Content Adaptation Domain, *Proceedings of the 22nd Annual ACM Symposium on Applied Computing*, Vol. 2, pp. 1450-1455, ISBN 1-59593-480-4, Seoul (Korea), March 2007, ACM, USA

Forte, M; Souza, W. L. & Prado, A. F. (2008). Using ontologies and Web services for content adaptation in Ubiquitous Computing, *Journal of Systems and Software,* Vol. 81 , No 3, March 2008, pp. 368-381, ISSN 0164-1212

Elson, J. & Cerpa, A. (2003). Internet Content Adaptation Protocol (ICAP), *Request for Comments 3507*, The Internet Society, http:/ / www.icap-forum.org/ documents/ specification/ rfc3507.txt

Hansmann, U.; Merk, L.; Nicklous, M. S. & Stober, T. (2003). *Pervasive Computing*. Springer-Verlag, ISBN 3-540-00218-9, Germany

Horrocks, I. et al (2003). SWRL: A Semantic Web Rule Language Combining OWL and RuleML, DAML, http:/ / www.daml.org/ 2003/ 11/ swrl/

IBM (2004). IBM Integrated Ontology Development Toolkit, http:/ / www.alphaworks.ibm. com/ tech/ semanticstk

ICOGNITO Technologies Ltd (2002). Dynamic Filtering of Internet Content: An Overview of Next Generation Filtering Technology, http:/ / www.icognito.com

Lara, R. et al. (2005). D2.4.2 Semantics for Web Service Discovery and Composition, *KnowledgeWeb*, http:/ / knowledgeweb.semanticweb.org/ semanticportal/ deliverables/ D2.4.2.pdf

Marques, M. C. & Loureiro, A. F. (2004). Adaptation in Mobile Computing, *Proceedings of the 22nd Brazilian Symposium on Computer Networks*, pp. 439-452, Gramado-RS (Brazil), Mai 2004, Brazilian Computer Society, Brazil

Martin, D. et al. (2006). OWL-S: Semantic Markup for Web Services. DAML, http:/ / www.ai.sri.com/ daml/ services/ owl-s/ 1.2/ overview/

Mastoli, V.; Desai, V.; Shi, W (2003). SEE: a service execution environment for edge services, *Proceedings of the Third IEEE Workshop on Internet Applications*, pp. 61-65, ISBN 0-7695-1972-5, San Jose-CA (USA), June 2003, IEEE Computer Society, USA

Network Apliance Inc (2001). Demo ICAP-Server by Network Appliance, http:/ / www.icap-forum.org/ icap?do=resources&subdo=specification

Ravindran, G.; Jaseemudin, M. & Rayhan, A (2002). A Management Framework for Service Personalization, In: *Management of Multimedia on the Internet*, Lecture Notes in Computer Science (LNCS) 2496, Kelvin C. Almeroth & Masum Hasan (Eds), pp. 276-288, Springer-Verlag, ISBN 3-540-44271-5, Germany

Rideout, V.; Richardson, C. & Resnick, P. (2002). See No Evil: How Internet Filters Affect the Search for Online Health Information, *The Henry J Kaiser Family Foundation*, http:/ / www.kff.org/ entmedia/ 20021210a-index.cfm

Rousskov, A (2005). Open Pluggable Edge Services (OPES) Callout Protocol (OCP) Core, *Request for Comments 4037*, The Internet Society, https:/ / tools.ietf.org/ html/ rfc4037

Smith, J; Mohan, R. & Li, C. (1998). Content-based Transcoding of Images in the Internet, *Proceedings of the 1998 IEEE International Conference on Image Processing*, Vol. 3, pp. 7-11, ISBN 0-8186-8821-1, Chicago-IL (USA), October 1998, IEEE Computer Society, USA

SourceForge (2002). Squid Web Proxy as ICAP Client, http:/ / icap-server.sourceforge.net/ squid.html

SourceForge (2003). Shweby Proxy Server, http:/ / shweby.sourceforge.net/

Tomlinson, G.; Chen, R. & Hofmann, M. (2001). A Model for Open Pluggable Edge Services, *Internet- Draft*, The Internet Society, http:/ / tools.ietf.org/ html/ draft-tomlinson-opes-model-00

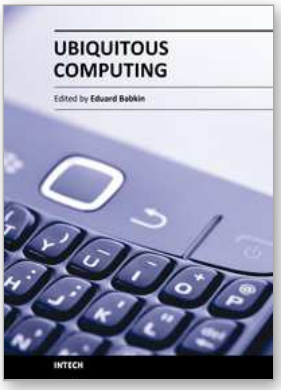
UDDI Spec TC (2004). UDDI Version 3.0.2, OASIS, http:/ / www.uddi.org/ pubs/ uddi_v3.htm

W3C (2004a). Composite Capability/ Preference Profiles (CC/ PP): Structure and Vocabularies 1.0, http:/ / www.w3.org/ TR/ CCPP-struct-vocab/

W3C (2004b). RDF/ XML Syntax Specification (Revised), http:/ / www.w3.org/ TR/ REC-rdf-syntax/

W3C (2004c). OWL Web Ontology Language Overview, http:/ / www.w3.org/ TR/ owl-features/

W3C (2007). Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, http:/ / www.w3.org/ TR/ wsdl20/



Ubiquitous Computing

Edited by Prof. Eduard Babkin

ISBN 978-953-307-409-2
Hard cover, 248 pages
Publisher InTech
Published online 10, February, 2011
Published in print edition February, 2011

The aim of this book is to give a treatment of the actively developed domain of Ubiquitous computing. Originally proposed by Mark D. Weiser, the concept of Ubiquitous computing enables a real-time global sensing, context-aware informational retrieval, multi-modal interaction with the user and enhanced visualization capabilities. In effect, Ubiquitous computing environments give extremely new and futuristic abilities to look at and interact with our habitat at any time and from anywhere. In that domain, researchers are confronted with many foundational, technological and engineering issues which were not known before. Detailed cross-disciplinary coverage of these issues is really needed today for further progress and widening of application range. This book collects twelve original works of researchers from eleven countries, which are clustered into four sections: Foundations, Security and Privacy, Integration and Middleware, Practical Applications.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Wanderley Lopes de Souza, Antonio Francisco do Prado, Marcos Forte and Carlos Eduardo Cirilo (2011). Content Adaptation in Ubiquitous Computing, Ubiquitous Computing, Prof. Eduard Babkin (Ed.), ISBN: 978-953-307-409-2, InTech, Available from: http://www.intechopen.com/books/ubiquitous-computing/content-adaptation-in-ubiquitous-computing

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821