Journal of Web Engineering, Vol. 6, No. 4 (2007) 360–388

A CLUSTER-BASED APPROACH TO WEB ADAPTATION IN CONTEXT-AWARE APPLICATIONS

ROBERTO DE VIRGILIO, RICCARDO TORLONE,
DAVIDE VALERIANO, DANIELE DI FEDERICO
Dipartimento di Informatica e Automazione, Università Roma Tre
Rome, Italy
{*devirgilio,torlone,valeriano,difederico*}*@dia.uniroma3.it*

Received January 1, 2007
Revised December 1, 2007

For mobile Web applications one needs to take into account context characteristics (such as the device capabilities, the network QoS, the user preferences, and the location) to meet the constraints of the client and guarantee a satisfying interaction with the user. A major issue in this framework is that, in real world scenarios, the number of adaptation requirements can change and increase very rapidly. Therefore, a relevant problem is the definition of effective methods for choosing efficiently the most suitable adaptation for a given context. To this aim, we propose in this paper a new cluster-based approach that automatically classifies the contexts on the basis of their characteristics: at a logical level, each class corresponds to contexts that require similar adaptations. We show that this classification strongly alleviates the adaptation process. The approach relies on a metric distance that is used to compare contexts and on a threshold that provides a reference to group them. Each context in a cluster is associated with the adaptation that best matches with the context requirements. We also illustrate an implementation of our approach and a number of experimental results that support its validity. Semantic Web design principles and enabling technologies are important ingredients of the overall framework.

Keywords: Web based Information System, Adaptive application, Clusterization, Semantic Web, RDF

1 Introduction

The World Wide Web has become the universal interface to information and, as result of this trend, the majority of information systems have been made available through the Web. An information system based on Web technology is usually called *Web Information System* (WIS) [1]. Some typical examples of Web Information Systems are e-commerce applications, online newspapers, educational sites, order tracking systems, etc. WISs are characterized by the fact that they provide the user with a hypermedia presentation of contents that is generated automatically from data sources, which can be distributed over the Web.

Recently, given the large spread of mobile devices, we have witnessed a shift toward an *ubiquitous* access to Web Information System. In this scenario, a very desirable feature is the ability of the system to adapt and personalize content delivery according to the *context* of the client, which involves the device capabilities, the network QoS, the user preferences, the location, and so on. Web-based systems that take into consideration personal, technical and, more in general, environmental characteristics in order to improve the interaction are usually referred as *adaptive* or *context-aware* Web systems. An important point that needs to be taken into account is that this scenario is very dynamic: many users with different preferences can be enabled to access the application, several and unpredicted types of access devices can be used, alternative network connections with varying quality rates can be made available, and so on. Moreover, the technology used to implement the adaptation can be changed and this should have a limited impact on the overall application. It follows that the adaptation process should guarantee a high level of flexibility in terms of fast responsiveness to rapidly changing requirements of adaptation and suitable actions to be undertaken to meet these requirements. In particular, the fundamental problem is how to select efficiently an effective adaptation that best fits the context requirements. Many approaches often demands this problem to the lower level, by proposing specific optimization techniques that aim at generating rapidly the adaptation on the basis of certain aspects of the context [2, 3, 4, 5]. The optimization is often based on caching frequently generated adaptations at the end of their elaboration.

In [6], we have proposed a comprehensive methodology for content adaptation that does not depend on a particular class of adaptation requirements. The methodology is based on a general context model that can be used to represent a variety of context components, called profiles, at different level of details [7]. The analysis of a profile (e.g., the profile of the access device) allows the generation of an abstract description of the various layers of a Web interface (content, navigation and presentation) that meets the adaptation requirements of the profile.

In this paper we extend this approach by introducing a clustering technique aimed at making efficient the adaptation process. Clustering is an unsupervised classification method widely used in Web usage mining, with the main objective to group a given collection of unlabeled objects into meaningful clusters [8, 9, 10]. In the Web domain the objects can be Web documents, references to Web documents, or user visits. Previous work includes text-based [11] and link-based methods [12]. In our approach, clustering is used to automatically classify the contexts of the clients on the basis of common characteristics. At a logical level, each class corresponds to contexts that require similar adaptations and it turns out that this classification strongly alleviates the adaptation process. The clusterization relies on a computationally efficient threshold-based distance metric between profiles that allows the identification of an effective set of clusters. This set is managed efficiently by means of a predefined set of operations (insertion, deletion and merge) that try to keep balanced the clusters in a set. The approach presented in this paper have been implemented in a practical tool for the adaptation of content delivery in Web Information Systems, which we have developed at Roma Tre University. The validity of the approach has been confirmed by a number of experimental results performed with our tool.

A fundamental aspect of our approach is the adoption of conceptual and practical principles of the Semantic Web. At the conceptual level, ontologies are used to model context data: this simplifies strongly the management of shared knowledge represented in heterogeneous models and languages. At the practical level, the implementation makes use of Semantic Web enabling technologies. In particular, data and metadata are conveniently represented and managed using the RDF language, a major component of the Semantic Web, and Jena, a Java based technology for building Semantic Web applications. It should be said that, because of its relative novelty, there are very still few model-driven WIS design methodologies and tools that make use of technologies for the Semantic Web.

The remainder of this paper is structured as follows. Section 2 provides an overview of the related work by investigating existing methodologies and tools for the adaptation of WIS and presents a scenario of reference. Section 3 briefly introduces our modeling approach. Section 4 illustrates the notion of adaptation cluster and define an effective distance metric between contexts. Section 5 shows efficient algorithms to initialize and maintain an effective set of clusters by means of a threshold level. Section 6 presents a practical implementation of our approach. Section 7 illustrates a case study and some experimental results testing the efficiency and effectiveness of the clustering. Finally, Section 8 concludes the paper and sketches future works.

2 Related Work

2.1 Adaptation of Web applications

Various methodologies have been proposed for the development of hypermedia applications [13], including HDM [14], RMM [15] and OOHDM [16]. Most of them tackle also the problem of modeling *adaptive* Web applications by means of rules or filters that specify the adaptation. The Hera methodology [17] provides a notation for the representation of slices and links, which are the basic model elements of the application model. This model is used to represent the navigation structure and presentation aspects of the Web application. Adaptation is either modeled explicitly, by specifying different choices with links and alternative sub-slices, or implicitly, through a rule-based adaptation model. Rules can be invoked recursively but determinism and termination are guaranteed. The OO-H approach [18] proposes the use of personalization rules. These rules are associated to the links of the navigation model and this implies that the adaptation is performed at runtime, during the navigation. Similarly, an adaptation specification language is used in WSDM [19] to specify the adaptations of the navigation structure to be performed at runtime. Adaptation rules can be defined independently of the navigation and both on single elements (node, link) and on groups of elements. Another approach is OOHDM [20], which separates the adaptation from the navigation by adding a special class for each navigation node that requires an adaptation. Ceri et al. [21] propose a method for the adaptation of Web applications that focus on user interaction: event-condition-action (ECA) rules are used to specify adaptivity actions, mainly at the navigational level, to be undertaken when events, consisting in a page request, occur. Another approach uses an aspect-oriented modeling approach, a novel means to improve the modularity of software models [22]. Here the designer can dynamically personalize various visible aspects of an hypermedia application according to individual user needs, preferences or knowledge [23]. In UWE [24], the customization of a Web application is also specified by means of ECA rules. These rules are activated by events, such as the change of the bandwidth, but a precise definition of rule activation is not given. Koch et al. [25] use a conservative extension of the UML metamodel for Web engineering. Finally Dolog et al. [26] model adaptive navigation using UML state diagrams.

In [6] we have also proposed a rule-based technique for the adaptation of data delivery in Web application. Differently from the approaches cited above, our adaptation rules are activated by a context rather than by an event (e.g., a user action). These rules can be used to specify complex adaptations. Conversely, most of the previous works on adaptive Web applications have focused on fairly simple adaptations (e.g., automatically creating shortcuts in the site) and on personalizing a Web site to suit the needs of each individual user. Moreover, the majority of the above mentioned approaches provide specific solutions that are suited for a particular class of predefined adaptation requirements: most of them consider device characteristics and user preferences and allow the designer to describe their effect on navigation and presentation. This makes not exactly trivial the addition of a new adaptation functionality. In other words, in these approaches it is hard to take into account new aspects of the context and modify the adaptation accordingly. Conversely, we have addressed the problem from a generic perspective aimed at providing a uniform and general solution to diverse requirements of adaptation, possibly not known in advance.

2.2 Applications

Two of the most popular applications of Web based systems nowadays are e-learning and e-commerce and a wide range of recent research efforts on adaptability of Web based systems focus on them. In order to provide an adaptive behavior to these applications, the systems must be able to accommodate widely varying service demands. Many techniques focus on the adaptation according to the user navigation and rely on Web mining approaches [27]. An explicit and dynamic representation of the user is usually maintained in these approaches [28, 29]. However, the adaptation can be influenced not only by user preferences, but also by many other aspects such as the capabilities of the access mechanism, the network quality rate, the localization and so on [30]. In this scenario, the adaptation process can be rather involved given the large number of context information that can be taken into account and the fact that the requirements of adaptation can be very different and change rapidly. For this reason we have focused on an optimization strategy that is able to choose efficiently the adaptation that best fits context requirements, even in an involved situation.

Our approach relies on clustering, a technique often used in many areas for a wide range of problems including graph theory, business area analysis, information architecture, information retrieval. Clustering can be meant as the grouping of large amounts of objects in clusters, in such a way that the objects in one cluster are closely related compared to the relationships between objects in different clusters. There are various approaches to clustering: Wiggerts [31] surveys different methods and classifies them in four categories: graph theoretical algorithms, construction algorithms, optimization algorithms, and hierarchical algorithms. In the area of Web information systems, the objects to classify are Web documents, references to Web documents, or user information (i.e. accounting or visits). Conversely, in our approach a cluster represents a grouping of context information that require similar adaptations. To our knowledge, clustering has never been used for the optimization of content delivery adaptation in Web information systems. Indeed, it turns out that the method largely improve the efficiency and effectiveness of the entire adaptation process.

3 The adaptation approach

In this Section we give an overview of our approach to content adaptation [6, 7]. We first introduce two logical concepts on which it is based: (i) the profile, a component of the context that influences the adaptation, and (ii) the configuration, a possible outcome of the adaptation. We then illustrate the adaptation technique, which basically consists of generating a suitable configuration for a given set of profiles.

3.1 Profiles

A *profile* is a description of an autonomous aspect of the context in which the Web site is accessed and that should influence the structuring and presentation of its contents. Examples of profiles are the user preferences, the device capabilities, the location, and so on. A *dimension* is a property that characterizes a profile. Each dimension is described by means of a set of *attributes*. For instance, a profile for a client device can be represented by means of the hardware, software, and browser dimensions. The hardware dimension can be described by means of attributes like CPU, memory, and display. Attributes can be *simple* or *composite*. A simple attribute has a domain of values associated with it, whereas a composite attribute has a set of (simple or composite) attributes associated with it. For example, the display attribute can be composed by the simple attributes width and height.

In order to guarantee an homogeneous representation of different contexts, we assume that dimension and attribute names are always taken from a fixed top ontology [32, 33, 34]. This ontology describes the concepts that can be involved in a context, but does not constraint the structure of the context itself. The ontology is represented using the OWL language [35] and, for simplicity, make use of part-of relationships only. This makes the approach easily extensible, scalable, and compliant to the Semantic Web.

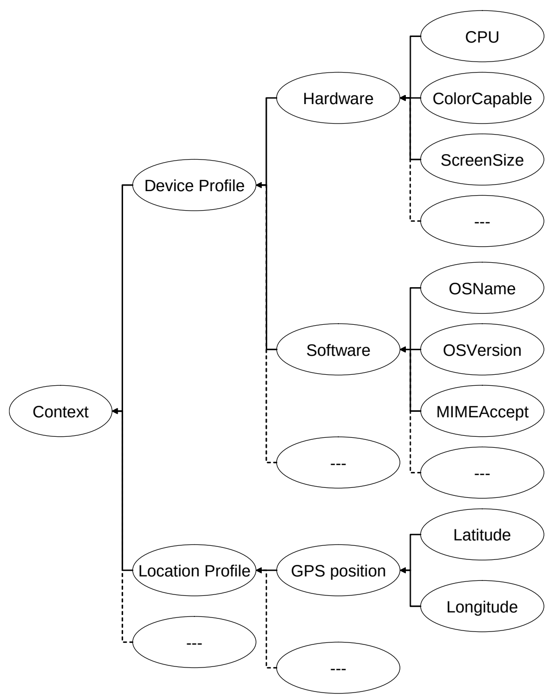


Fig. 1. The ontology of reference for the context model.

Therefore, we fix a vocabulary $\mathbf{V} = \{\mathbf{D}, \mathbf{A}\}$ where $\mathbf{D}$ is a set of *dimension names* and $\mathbf{A}$

is a set of (simple and composite) *attributes names*, all of them taken from a shared top ontology, partially shown in Figure 1. We denote by $D(A_1, \ldots, A_n)$ a *dimension definition*, where $D \in \mathbf{D}$ and $A_i \in \mathbf{A}$, for $1 \leq i \leq n$. If A_i is a simple attribute, then it is associated with a set of values called *domain*, otherwise, it has associated with a set $A_{i_1}, \ldots, A_{i_k}$ of (sub-)attributes.

Definition 1 (Profile and context) *A (generic)* profile *P over a set of dimension definitions $D_1(X_1), \ldots, D_n(X_n)$ is function that associates with each simple attribute of every dimension a value taken from its domain. A* context *is a collection of profiles.*

Examples of profiles are represented graphically in Figure 2. Profile P_1 has an hardware, a software, and a Category dimension. The hardware dimension has three attributes: CPU, memory, and display. The composite attribute display is composed by the simple attributes width and height.

An important aspect of our model is that different profiles can be compared making use of a subsumption relationship $\lhd$. Intuitively, given two profiles P_1 and P_2, if $P_1 \lhd P_2$ then P_2 is more detailed than P_1 since it includes the attributes of P_1 at the same or at greater level of detail.

More precisely, we first say that an attribute A of a profile P *is covered* by an attribute A of a profile P' if either they are simple and $P(A) = P'(A)$, or they are composite and for each sub-attribute A_i of A in P there is a sub-attribute A_j of A in P' such that A_i is covered by A_j. The subsumption relationship is then defined as follows.

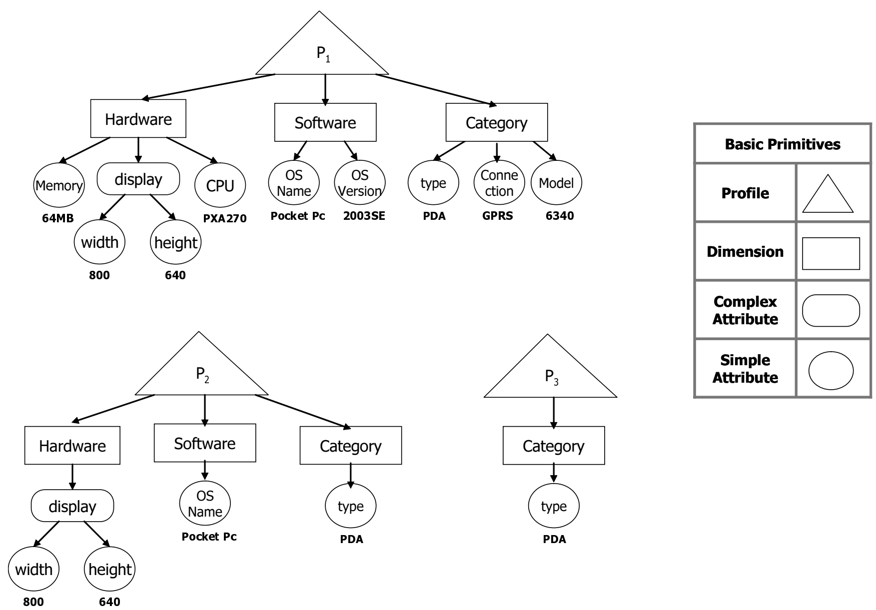


Fig. 2. Subsumption between profiles.

Definition 2 (Subsumption) *Given two profiles P_1 and P_2, we say that P_1 is subsumed by P_2, $P_1 \lhd P_2$, if for each dimension d of P_1 there is a dimension d' of P_2 such that for each attribute A of d there is an attribute A' of d' that covers A.*

As an example, given the profiles reported in Figure 2, we have that $P_3 \lhd P_2 \lhd P_1$. It is easy to show that $\lhd$ is a partial order relationship over profiles, as it is reflexive, antisymmetric and transitive. Therefore, we can define two binary operations on profiles: the *meet* ($\sqcap$) and *join* ($\sqcup$). Intuitively, given two profiles P_1 and P_2, the meet represents the greatest profile including the information in common between P_1 and P_2, whereas the join represents the least profile including all the information of both P_1 and P_2.

Definition 3 (Meet) *The* meet *of two profiles P_1 and P_2, denoted by $P_1 \sqcap P_2$, is a profile P such that, $P \lhd P_1$, $P \lhd P_2$ and, for each profile $P' \neq P$ such that $P' \lhd P_1$, $P' \lhd P_2$, it is the case that $P' \lhd P$.*

Definition 4 (Join) *The* join *of two profiles P_1 and P_2, denoted by $P_1 \sqcup P_2$, is a profile P such that, $P_1 \lhd P$, $P_2 \lhd P$ and, for each profile $P' \neq P$ such that $P_1 \lhd P'$, $P_2 \lhd P'$, it is the case that $P \lhd P'$.*

As an example, given the profiles reported in Figure 3, we have that $P_6 = P_4 \sqcap P_5$ and $P_7 = P_4 \sqcup P_5$.

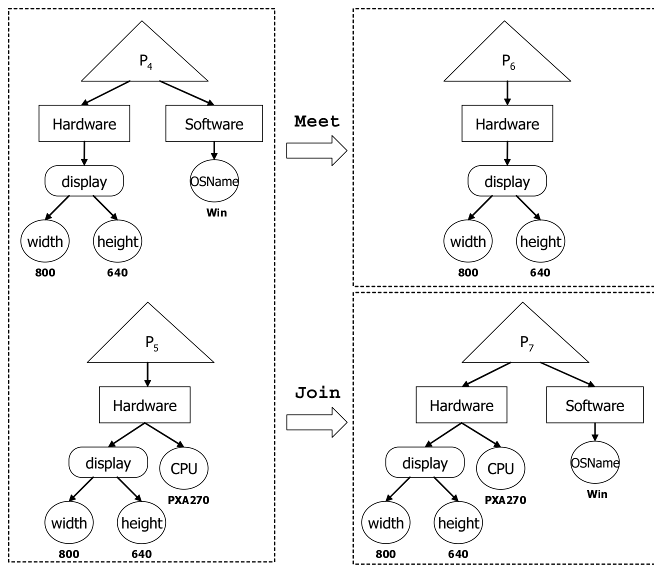


Fig. 3. An example of Meet and Join between profiles.

3.2 Configurations

The notion of configuration is used in our approach to describe, in abstract terms, a suitable adaptation [36, 6]. According to the classical organization of a Web resource (content, navigation, and presentation) [37], a configuration has three components: a *content view* (for adapting the content), an *hypertext definition* (for adapting the navigation), and a *logical style sheet* (for adapting the presentation).

A content view corresponds to a selection of contents extracted (dynamically) from the underlying database and is represented by a first order formula expression. An hypertext definition describes a way to organize the elements of the content view into a hypertext and, basically, is organized in a set of pages composed in turn of a set of units. A unit represents an atomic portion of a linked page and includes a direct reference to the elements of a content view from which it takes data. Finally, a logical style sheet describes the layout of the pages mentioned in a hypertext definition. It is based on a predefined set of *Web object types* (such as text, image, video), each of which is associated with a set of presentation attributes that identify possible styles (e.g. font, color, spacing, position) that can be specified for them.

As an example, a graphical representation of a configuration $C = (v, h, s)$ is given in Figure 4. In this Figure: (i) the content view v has been represented by the source (a relational databases), the target (a view over the source), and by an arrow labeled with the selection condition, (ii) the hypertext definition h by a site view (according to the hypertext model proposed by [37]), and (iii) the logical style sheet s by a table. A textual representation of C is reported below.

$q =$ $\{T : x_1, S : x_2, P : x_3, C : x_4, D : x_5, JN : x_6 \mid$
$\mathit{NewsItems}(N : x_0, T : x_1, S : x_2, D : x_5, G : x_7, J : x_8),$
$\mathit{Details}(I : x_0, P : x_3, C : x_4),\ \mathit{Journal}(J : x_8, JN : x_6, C : x_9) \mid x_7 = \mathit{Sport}\}$
$h =$ {**Page** HomePage [**IndexUnit** NewsIndex **hierarchical** (**source** Journal;
attributes JName; **NEST** News item **attributes** Title, Summary; **orderby** Title)],
Page NewsPage [**DataUnit** NewsData (**source** Selected news;
attributes Title, Date, Picture, Content; **orderby** Title)],
Link NewsToDetails **from** NewsIndex **to** NewsData **parameters** (Title = NT),
Link ToHome **from** NewsData **to** NewsIndex}
$s =$ {**Text**(**Font**: Arial, ..., **Border**: 0pt), **Link**(**Note**: False, ..., **Color**: Blue),
Image(**Format**: jpeg, ..., **Alignment**: left)}

An important point is that a configuration is a logical notion that can be represented and implemented in several ways. This property guarantees the generality of the approach with respect to actual languages and tools used to implement the adaptive application.

3.3 Adaptation

Several solutions to the problem of content delivery adaptation have been proposed in the literature [38]. In [6] we have proposed a novel approach to this problem in which special adaptation rules are used to represent, declaratively, how to build a response to a user request that meets the requirements of a given context. More specifically, an adaptation rule has the form $P_r : C_d \Rightarrow C_f$ where P_r is a parametric profile, C_d is a logical condition on the parameters occurring in P_r, and C_f is a parametric configuration in which parameters occurring in P_r can appear in place of values. The intuitive semantics of an adaptation rule is the following: if the client has a profile P_r and the condition C_d is verified then generate the configuration C_f.

For instance, let us consider a profile P_1 describing the display's size of a device such as $P_1(\text{Hardware}(\text{display}(\text{width} : 800, \text{height} : 640)))$. Since P_1 represents a device with limited size of the screen, we need to decompose large Web pages and to resize text and images. This operation can be represented by the rule shown in Figure 5: it describe the configuration to generate under certain conditions of width and height of the display.

However, other approach can be used to generate the adaptation. In order to show the generality of the clustering technique presented in the next Section, we just assume to have

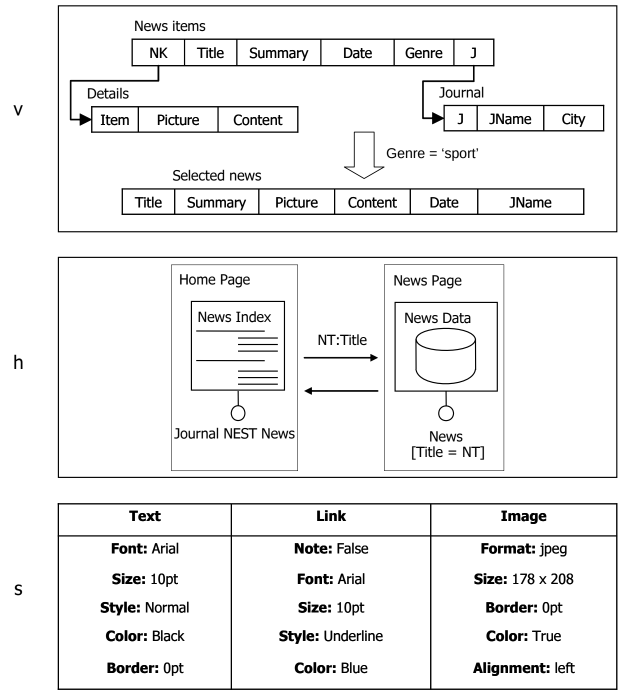


Fig. 4. A configuration $C = (v, h, s)$.

some adaptation method at disposal and we will refer, hereinafter, to the following general notion of adaptation.

Definition 5 (Adaptation) *An adaptation is a function* α *that associates to a profile* P *a configuration* $C_f = \alpha(P)$. *We say in this case that* C_f *matches with* P.

4 A Clustering Technique for Adaptation

In a real world scenario, the number of clients with different requirements of adaptation can increase rapidly. It is therefore a challenge to guarantee an effective and efficient responsiveness of the adaptation system. To this aim, we now propose in this Section a new strategy based on an automatic clusterization of profiles. Basically, each cluster represents a class of profiles that presents similar adaptation requirements. The clusterization is inspired by a method to efficiently classify XML documents on the basis of their structure [39]. We start by illustrating an overall adaptation process and then focus on the clusterization phase.

4.1 The Adaptation Process

Initially, we fix: (i) a default configuration C, which represents a basic Web interface of our Web application, (ii) an initial set of adaptation rules, each of which captures the criteria of adaptation for a basic profile, and (iii) a precedence partial order on the rules.

The process of content adaptation is graphically illustrated in Figure 6 and can be summarized as follows:

$R = P_d(\text{Hardware}(\text{display}(\text{width} : Y, \text{height} : Z))) : Y > 500 \sqcap Z > 300 \Rightarrow$
({ Ti, Su, Da, Co | NewsItems(NK, Ti, Su, Da, Ge, Ju),
Details(NK, Pi, Co) | true},
{ **Page** News { **IndexUnit** NewsIndex (**Attributes** Title, Date;
OrderBy Date) },
Page Details { **DataUnit** ContentData(**Attributes** Title, Date, Content) },
Link NewsToDetails (**From** NewsIndex **To** ContentData
Parameters NK) },
{ **Text**(**Color**: Black, **Size**: 8pt), **Link**(**Style**: Underline, **Color**: Black),
Image(**Size**: $Y \cdot 0.5 \times Z \cdot 0.5$, **Color**: true) })

Fig. 5. An example of adaptation rule.

1. the context of the client is captured and represented as a set of profiles $\{P_1, \ldots, P_k\}$, one for each coordinate of adaptation (e.g., the device, the user preferences, the location). All the profiles are expressed in terms of the model presented in Section 3;

2. each profile P_i is stored into a dedicated repository in which profiles are clusterized according to their similarity; in these clusters, each profile is associated with a configuration that matches with it;

3. all the configurations C_f that match with each P_i are collected into a set S_C that includes the default configuration C (this guarantees that S_C contains at least one configuration);

4. all the configurations in S_C are merged into a unique configuration C' by using a prioritized composition operator;

5. the resulting configuration C' is implemented into practical languages (e.g., SQL statements at the content level, XHTML or JSP statements at the navigation level, and CCS style sheets at the presentation level);

6. the adaptation statements are executed by the underlying systems and the final response is generated.

The step 1 and the steps 3-6 of this process have been investigated in details in [7] and in [6], respectively. In this paper we focus on the step 2. The goal is to extend the basic approach proposed in [7] to make the adaptation selection more efficient and effective.

4.2 Clusterization of profiles

In our approach, a cluster of profiles is organized in a tree, defined as follows.

Definition 6 (Adaptation Cluster) *An adaptation cluster* $\mathcal{C}$ *is a tree* $\{N_\mathcal{C}, E_\mathcal{C}, r_\mathcal{C}\}$ *where*

- $N_\mathcal{C}$ *is a set of nodes representing profiles,*
- $E_\mathcal{C}$ *is a set of edges* (n_i, n_j) *such that* $n_i \lhd n_j$, *and*
- $r_\mathcal{C}$ *is the root.*

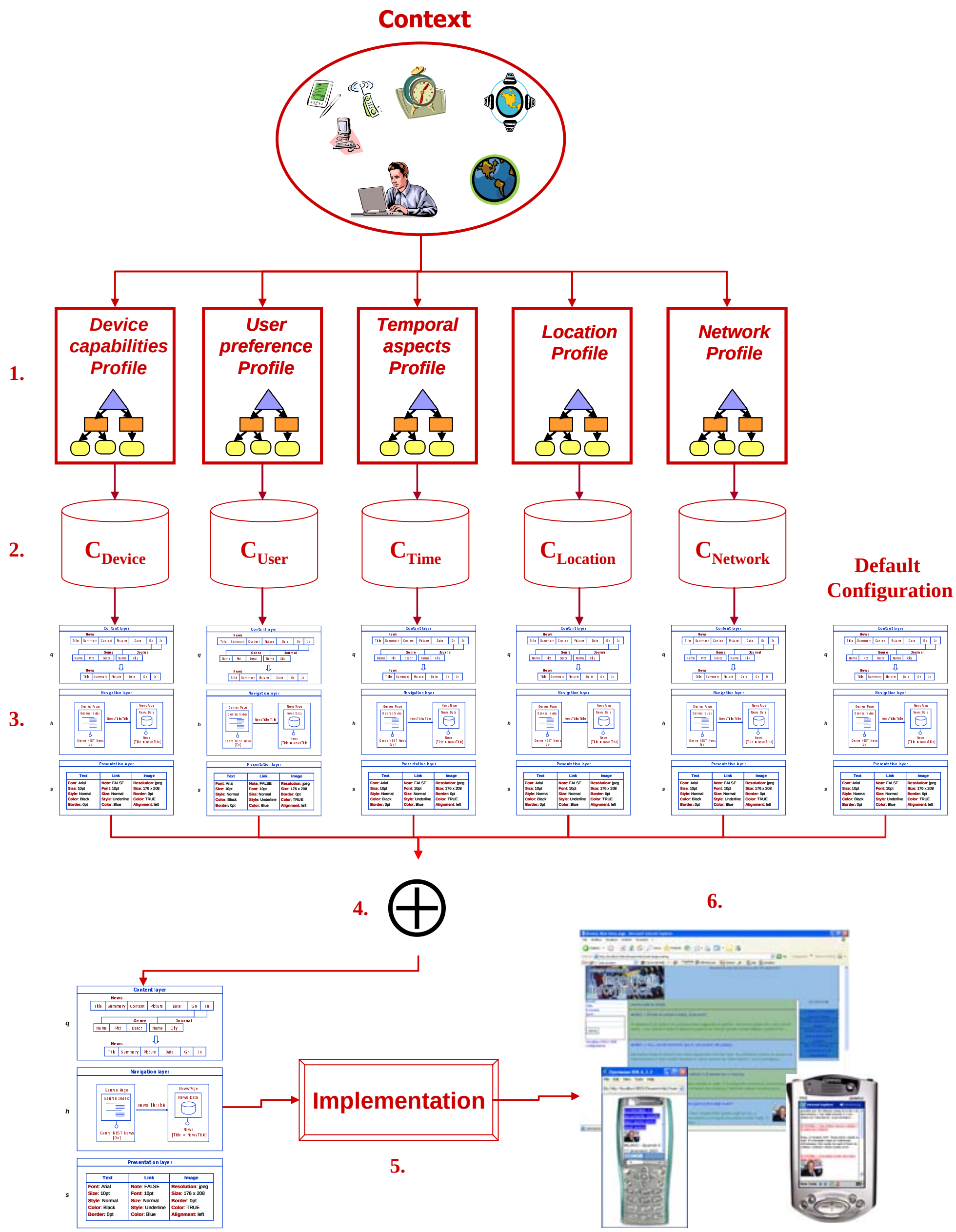
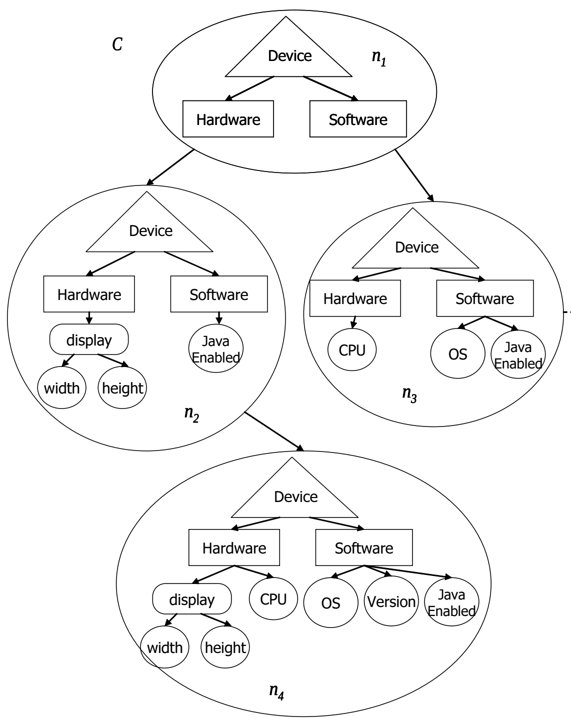


Fig. 6. The adaptation process.

In a cluster, a profile P_1 is parent of a profile P_2 if $P_1 \lhd P_2$, therefore the root of cluster represents the most general profile in the cluster. Each profile is associated with a configuration that matches it. For instance, let us consider the cluster $\mathcal{C}$ in Figure 7. It represents the group of profiles corresponding to hardware and software specifications of a device, at different levels of details. In the following, we will use the following auxiliary functions.

- $\rho(P, \mathcal{C})$ is a function that returns the parent of a profile P in a cluster $\mathcal{C}$;
- $\Lambda(P, \mathcal{C})$ is a function that returns the children of a profile P in a cluster $\mathcal{C}$;
- $\lambda(i, \mathcal{C})$ is a function that returns all nodes of a cluster $\mathcal{C}$ at level i, with $i > 0$.
- $\pi(\mathbf{C})$ is a function that returns all profiles occurring in a set of clusters $\mathbf{C}$.

As an example, if we apply the auxiliary functions above described to the cluster $\mathcal{C}$ in Figure 7, we obtain: $\rho(n_3, \mathcal{C}) = n_1$, $\Lambda(n_1, \mathcal{C}) = \{n_2, n_3\}$ and $\lambda(3, \mathcal{C}) = n_4$.



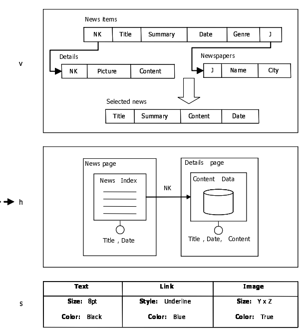


Fig. 7. An example of Cluster $\mathcal{C}$.

To measure the similarity of two profiles, we propose the following metric distance.

Definition 7 (Distance between profiles) *The distance between two profiles* P_1 *and* P_2*, denoted by* $d(P_1, P_2)$*, is defined as follows:* $d(P_1, P_2) = 1 - \frac{|P_1 \sqcap P_2|}{max\{|P_1|, |P_2|\}}$*, where* $|P|$ *denotes the number of edges of a profile* P*.*

It is easy to show that $d(P_1, P_2)$ is a metric [40]. Basically, if the structure similarity of P_1 and P_2 is large, the distance between them is small, and vice versa. As an example, in the cluster $\mathcal{C}$ in Figure 7, the distance between the profiles n_2 and n_4 is $d(n_2, n_4) = 0.\overline{3}$.

Now, given a set of adaptation clusters $\mathbf{C}$ and a profile P describing a new characteristic of our client, or a modification thereof, the effective search of the configuration C_f that match with P can be efficiently performed as follows.

- We select the cluster $\mathcal{C} \in \mathbf{C}$ such that (i) $r_{\mathcal{C}} \lhd P$, and (ii) the distance between P and $r_{\mathcal{C}}$ is minimum (*step (1)*); if there is no cluster whose root subsumes P then we return a default configuration associated with the default profile $P_{default}$;
- Starting from the root of $\mathcal{C}$, we return a configuration C_f as follows:
  - if there is no child of the root $r_{\mathcal{C}}$ of $\mathcal{C}$ such that $r_{\mathcal{C}} \lhd P$, then we return the configuration $\alpha(r_{\mathcal{C}})$ associated with $r_{\mathcal{C}}$;
  - otherwise, we search in the sub-tree of $\mathcal{C}$ having as root the child n' of $r_{\mathcal{C}}$ such that (i) $n' \lhd P$, and (ii) the distance between P and n' is minimum (*step (2)*);
  - when we find a profile n in $\mathcal{C}$ such that no child of n subsumes P, we return the configuration $\alpha(n)$ associated with n (*step (3)*).
- The detailed algorithm is shown in Figure 8.

Input: $P, \mathbf{C} = \{\mathcal{C}_1, ..., \mathcal{C}_n\}$
Output: C_f
begin
    $n = P_{default}$;
(1) **if there is** $\mathcal{C} \in \mathbf{C}$ **such that** $r_{\mathcal{C}} \lhd P$ **and** min(d(P,$r_{\mathcal{C}}$)) **then**
        n = $r_{\mathcal{C}}$;
        L = Λ(n,$\mathcal{C}$);
(2)   **while** L $\neq \emptyset$ **do**
            if it exists $n' \in$ L **such that** $n' \lhd P$ **and** $min(d(P, n'))$ **then**
                $n = n'$;
                $L = \Lambda(n, \mathcal{C}')$;
            else $L = \emptyset$;
(3) **return** $\alpha(n)$;
end

Fig. 8. The choice of a configuration C_f.

5 Management of Clusters

In this Section we illustrate a set of operations devoted to the generation and management of adaptation clusters.

5.1 A Threshold of Similarity

An important task in the our approach is the construction of an initial set of clusters. A nodal point is that clusters should be always maintained balanced: we should avoid situations in which we have many clusters with few profiles or, conversely, a limited number of clusters including many profiles. In the former case, the selection of a profile would require an exhaustive search. In the latter, update operations would become rather inefficient. To this aim, we propose a special threshold of similarity to estimate when a profile can be included in a cluster.

Definition 8 (Threshold of similarity) *The threshold of similarity* σ *of a set of profiles* $\boldsymbol{P} = \{P_1, \ldots, P_n\}$*, is defined by the following function:* $\tau(\boldsymbol{P}) = 1 - \frac{|P_1 \sqcap P_2 \sqcap \ldots \sqcap P_n|}{max\{|P_1|, |P_2|, \ldots, |P_n|\}}$.

This threshold measures the weighted average of common edges of the profiles in the cluster or, alternatively, the average distance between them. The idea is that a profile can be included in a cluster if the distance from its root is lower than the threshold of the cluster. In this way σ represents always the maximum distance between a profile P in a cluster $\mathcal{C}$ and the root of $\mathcal{C}$.

Note that if $\tau(\mathbf{P}) = 0$ we have a cluster for each profile. Conversely, if $\tau(\mathbf{P}) = 1$ we obtain just one cluster including all the profiles. From this observations, it turns out that the numbers of clusters is inversely proportional to the threshold σ.

The aim of the threshold σ is to give a feedback to the system administrator in order to evaluate the effectiveness and the efficiency of the adaptation process. Indeed, the number of clusters obtained is strictly related to the design of the adaptations: a right design of profiles and configurations implies a good and easy clustering and a reduction of the search space; conversely, an involved clustering of profiles implies an inaccurate adaptation design and an inefficient processing. Therefore, a balanced set of clusters is an important requirement for an effective design in terms of coherent adaptation to homogeneous sets of profiles, and an efficient adaptation process in terms of responsiveness of the system.

5.2 Update operations

Adaptation clusters can be managed by means of a set of predefined operations to insert, delete and merge profiles occurring in clusters and to initialize clusters.

Insert Given an initial set of clusters $\mathbf{C}$, a new profile P can be inserted into a cluster in $\mathbf{C}$ as follows.

```
Input: P, C = {C_1, ..., C_n}
begin
(1)   σ = τ(π(C) ∪ P);
(2)   if there is C ∈ C such that r_C ◁ P and min(d(P, r_C)) and d(P, r_C) ≤ σ then
              n = r_C; level = 1;
              L = Λ(n, C);
(3)           while L ≠ ∅ do
                     if there is n' ∈ L such that n' ◁ P andmin(d(P, n')) then
                     n = n'; L = Λ(n, C); level = level + 1;
                     else L = ∅;
(4)           N_C = N_C' ∪ {P}; E_C = E_C ∪ {(n, P)};
(5)           ∀n'' ∈ λ(level, C) such that P ◁ n'' do
                     E_C = E_C − {(ρ(n'', C), n'')};
                     E_C = E_C ∪ {(P, n'')};
      else C' = {N_C' = {P}, E_C' = ∅}; C = C ∪ C';
end
```

Fig. 9. The Insertion of a profile P in a cluster $\mathcal{C}$.

- We calculate the new threshold $\sigma = \tau(\pi(\mathbf{C}) \cup P)$ (*step (1)*);

- We select the cluster $\mathcal{C}$ having as root $r_{\mathcal{C}}$ such that (i) $r_{\mathcal{C}} \triangleleft P$, (ii) the distance between P and $r_{\mathcal{C}}$ is minimum, and (iii) this distance is lower than σ (*step (2)*); if there is no cluster satisfying these properties then we add to **C** a new cluster having P as root;
- Starting from the root of $\mathcal{C}$, we insert P in $\mathcal{C}$ as follows:
  - if there is no child n of the root $r_{\mathcal{C}}$ of $\mathcal{C}$ such that $n \triangleleft P$, then (i) P becomes the child of $r_{\mathcal{C}}$, and (ii) each child n of $r_{\mathcal{C}}$ such that $P \triangleleft n$ becomes child of P;
  - otherwise, we insert P in the sub-tree of $\mathcal{C}$ having as root the child n of $r_{\mathcal{C}}$ such that (i) $n \triangleleft P$, and (ii) the distance between P and n is minimum (*step (3)*);
  - once P has be inserted in $\mathcal{C}$ (*step (4)*), we move each n'' such that (i) $P \triangleleft n''$ and (ii) n'' is at the same level of P, as a child of P (*step (5)*).
- The detailed algorithm is shown in Figure 9.

Let's consider the cluster $\mathcal{C}$ shown in Figure 7. In Figure 10 we illustrate an example of Insert that introduces in the cluster $\mathcal{C}$ the new profile n_5.

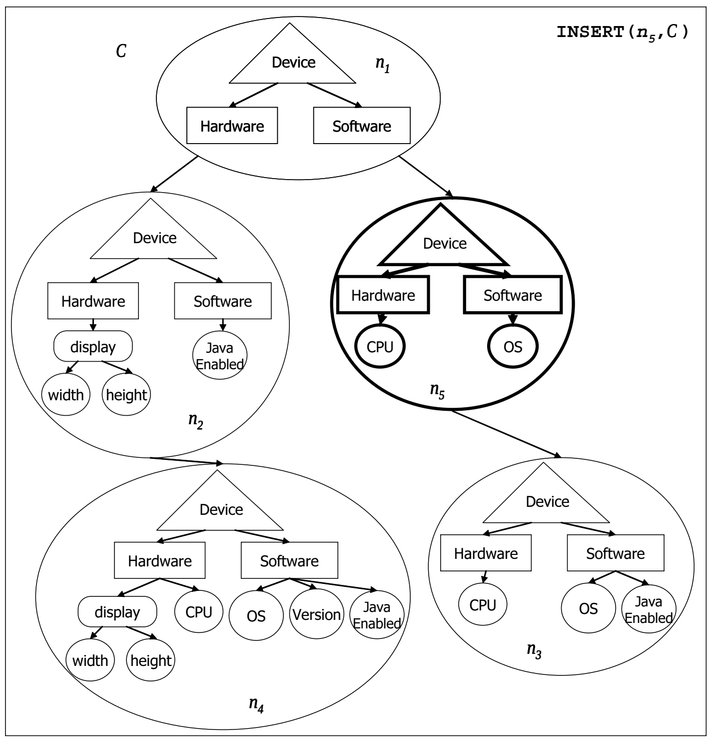


Fig. 10. An example of Insert.

Delete Given an initial set of clusters **C**, and a profile P occurring in **C**, P can be deleted from **C** as follows.

- We select the cluster $\mathcal{C}$ in **C** that contains P (*step (1)*);

- For each child n of P in $\mathcal{C}$, we delete the edge from P to n and add an edge from the parent $\rho(P, \mathcal{C})$ of P to n (*step (2)*);
- We delete the edge from the parent $\rho(P, \mathcal{C})$ of P to P (*step (3)*);
- We update the threshold σ associated with $\mathbf{C}$ (*step (4)*).
- The detailed algorithm is shown in Figure 11.

Input: $P, \mathbf{C} = \{\mathcal{C}_1, ..., \mathcal{C}_n\}$
begin
(1) **if there is** $\mathcal{C} \in \mathbf{C}$ **such that** $P \in N_{\mathcal{C}}$ **then**
(2) $\forall (P, n') \in E_{\mathcal{C}}$ **do** $E_{\mathcal{C}} - \{(P, n')\}$;
$E_{\mathcal{C}} \cup \{(\rho(P, \mathcal{C}), n')\}$;
(3) $N_{\mathcal{C}} = N_{\mathcal{C}} - P; E_{\mathcal{C}} = E_{\mathcal{C}} - \{(\rho(P, \mathcal{C}), P)\}$;
(4) $\sigma = \tau(\pi(\mathbf{C}) - P)$;
end

Fig. 11. The Deletion of a profile P in a cluster $\mathcal{C}$.

Referring to the cluster $\mathcal{C}$ shown in Figure 7, in Figure 12 we show an example of Delete that erases the profile n_2 from the cluster $\mathcal{C}$.

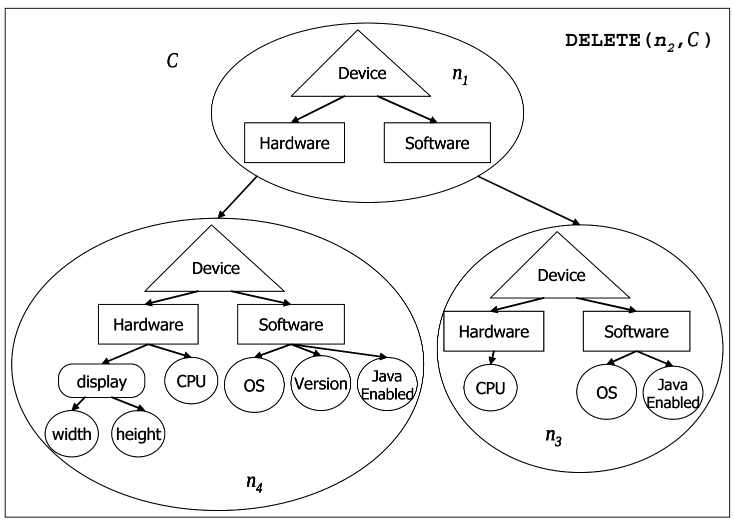


Fig. 12. An example of Delete.

Merge Given an initial set of clusters $\mathbf{C}$, and two profiles P_1 and P_2 occurring in $\mathbf{C}$, we merge P_1 and P_2 in $\mathbf{C}$ as follows.

- We build the profile P corresponding to the join between P_1 and P_2;
- We delete P_1 and P_2 from $\mathbf{C}$ and insert P in $\mathbf{C}$
- The detailed algorithm is shown in Figure 13.

Input: $P_1, P_2, \mathbf{C} = \{\mathcal{C}_1, ..., \mathcal{C}_n\}$
begin
  $P = P_1 \sqcup P_2$;
  Delete$(P_1, \mathcal{C})$; **Delete**$(P_2, \mathcal{C})$;
  Insert$(P, \mathcal{C})$;
end

Fig. 13. The Merge of two profiles P_1 and P_2 in a cluster $\mathcal{C}$.

Referring to the cluster $\mathcal{C}$ shown in Figure 7, in Figure 14 we show an example of Merge applied to the profiles n_2 and n_3 in the cluster $\mathcal{C}$.

Initialization Given a set of profiles **P**, we generate a set of adaptation clusters **C** from **P** as follows.

- We order **P** according to their size (number of edges);
- We set the threshold σ to 1, and we use a set $\mathcal{S}$ to evaluate incrementally the threshold σ (*step (1)*);
- We select a *pivot profile* P_v from **P** (the first profile), we include P_v in $\mathcal{S}$ and we add the cluster $\mathcal{C}_v$ with root P_v in **C** (*step (2)*);

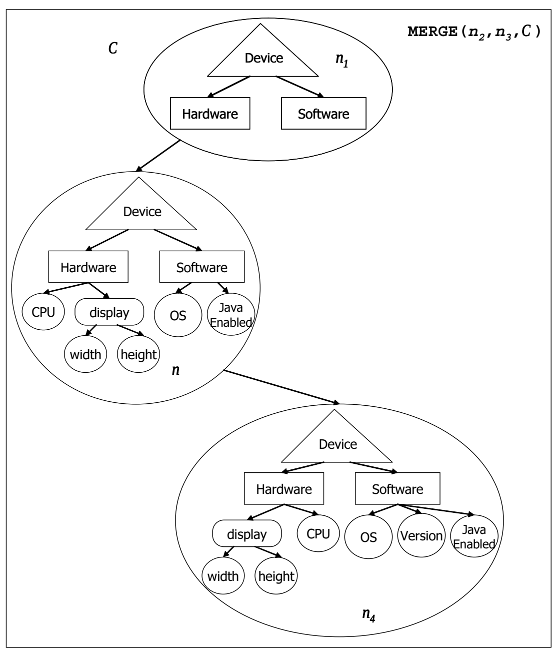


Fig. 14. An example of Merge.

- While **P** is not empty

- We select a profile P' from $\mathbf{P}$ (*step (3)*);
- If P' can not be inserted into a cluster in $\mathbf{C}$ then we compare P' with P_v (*step (4)*): (i) if $P_v \lhd P'$ then P' is the new root of the cluster $\mathcal{C}_v$ and the new pivot (*step (5)*), (ii) otherwise we decide if a new cluster need to be created or P' can be included in $\mathcal{C}_v$ (*step (5)*) as follows: if the distance between P_v and P' is greater than σ then (i) we create a new cluster $\mathcal{C}'_v$ rooted in P' and set P' as new pivot, (ii) otherwise we evaluate the meet P^* between P' and P_v, set P^* as new root of the cluster $\mathcal{C}_v$ and link P' and P_v to P^* in $\mathcal{C}_v$, update $\mathcal{S}$ and the threshold σ;
- Otherwise we add P' to $\mathcal{S}$ and we insert P' in $\mathbf{C}$ using the **Insert** operation;

- Finally, we return the set $\mathbf{C}$ (*step (7)*).
- The detailed algorithm is shown in Figure 15.

Input: $\mathbf{P} = \{P_1, \ldots, P_n\}$
Output: $\mathbf{C} = \{\mathcal{C}_1, ..., \mathcal{C}_m\}$
begin
(1) $\mathbf{C} = \emptyset; \sigma = 1; \mathbf{P} = \mathbf{P} - \{P_v\}; \mathcal{S} = \{P_v\};$
(2) $\mathcal{C}_v = \{\{P_v\}, \{\emptyset\}\}; \mathbf{C} = \mathbf{C} \cup \mathcal{C}_v;$
  while $\mathbf{P}$ *is not empty* **do**
(3) $\mathbf{P} - \{P'\};$
(4)   **if not** $\exists\, \mathcal{C}_i \in \mathbf{C}$ **such that** $r_{\mathcal{C}_i} \lhd P'$ **then**
(5)   **if** $P_v \lhd P'$ **then**
      $N_{\mathcal{C}_v} \cup \{P'\};\ E_{\mathcal{C}_v} \cup \{(P', P_v)\};\ P_v = P';$
    else
(6)   **if** $d(P_v, P') > \sigma$ **then**
        $P_v = P';$
        $\mathcal{C}'_v = \{\{P_v\}, \{\emptyset\}\}; \mathbf{C} = \mathbf{C} \cup \mathcal{C}'_v;$
      else
        $P^* = P_v \sqcap P';$
        $N_{\mathcal{C}_v} \cup \{P', P^*\};\ E_{\mathcal{C}_v} \cup \{(P^*, P_v), (P^*, P')\};$
        $\mathcal{S} \cup \{P_v, P', P^*\}; \sigma = \tau(\mathcal{S});$
  else $\mathcal{S} \cup \{P'\}$; **Insert**$(P', \mathbf{C})$;
(7) **return** $\mathbf{C}$;
end

Fig. 15. The Initialization of a set of clusters **C**.

As an example, let us consider the set of profiles $\mathbf{P}$ and the corresponding adaptations reported in Figure 16. The initialization of a set of clusters $\mathbf{C}$ proceed as follows. Initially, the pivot element chosen from $\mathbf{P}$ is the profile P_1, and we insert in $\mathbf{C}$ the cluster $\mathcal{C}_v$ having P_1 as root. In the first iteration we select P_2: we cannot insert P_2 in $\mathbf{C}$ (i.e. it doesn't exist a cluster $\mathcal{C} \in \mathbf{C}$ such that P_2 subsumes the root $r_{\mathcal{C}}$), so since P_1 (the pivot) is not subsumed by P_2, the partial threshold $\sigma = 1 - \frac{|P_1 \sqcap P_2|}{|P_2|} = 0.75$ and $d(P_1, P_2) < 0.75$, then we generate the profile P^* as the meet of P_1 and P_2, and we update $\mathcal{C}_v$. In the second iteration we select P_3 from $\mathbf{P}$. In this case, we can insert P_3 in $\mathbf{C}$ and we obtain $\sigma = 0.80$. In the third iteration, we select P_4 from $\mathbf{P}$. It is not possible (i) to insert P_4 in $\mathbf{C}$, and (ii) to compare P_4 with P_v and $\mathcal{C}_v$ because P_v is not subsumed by P_4 and the distance $d(P_v, P_4) > 0.80$. So we create a new

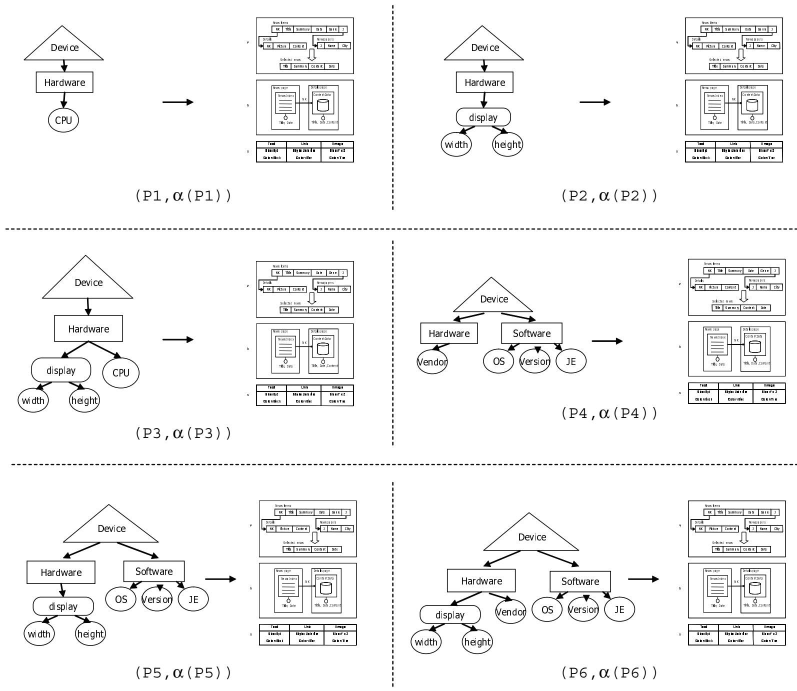


Fig. 16. A set of profiles.

cluster $\mathcal{C}'_v$ having P_4 as root, and we set P_4 as the new pivot. Proceeding similarly, at the end of the algorithm we obtain the clusters of Figure 17.

6 Implementation

We have developed an experimental tool called FAWIS (a short hand for *Flexible Adaptation of Web Information Systems*) that implements the approach described in this paper and supports the design and the development of adaptive data delivery applications over the Web. Further details on this tool can be found at `http://mais.dia.uniroma3.it/FAWIS`. The overall architecture of reference for our framework is depicted in Figure 18. The system is implemented in Java and runs on any Java-enabled platform. It involves the following components:

- a MultiProxy captures the client request to add or modify its context information;
- an HTTP Server translates a specific request of a client (a page or a specific object) into a query over the underlying database;
- the Context Manager (CM) is able to get and manage a description of the client context and generate a suitable configuration.
- the Adaptation Designer (AS) is responsible to support the Web designer, by means of a menu-driven interface, to the definition of a set of adaptation rules; it supervises the entire process of generation of a configuration;
- the Response Generator produces the final response, taking as input the configura-

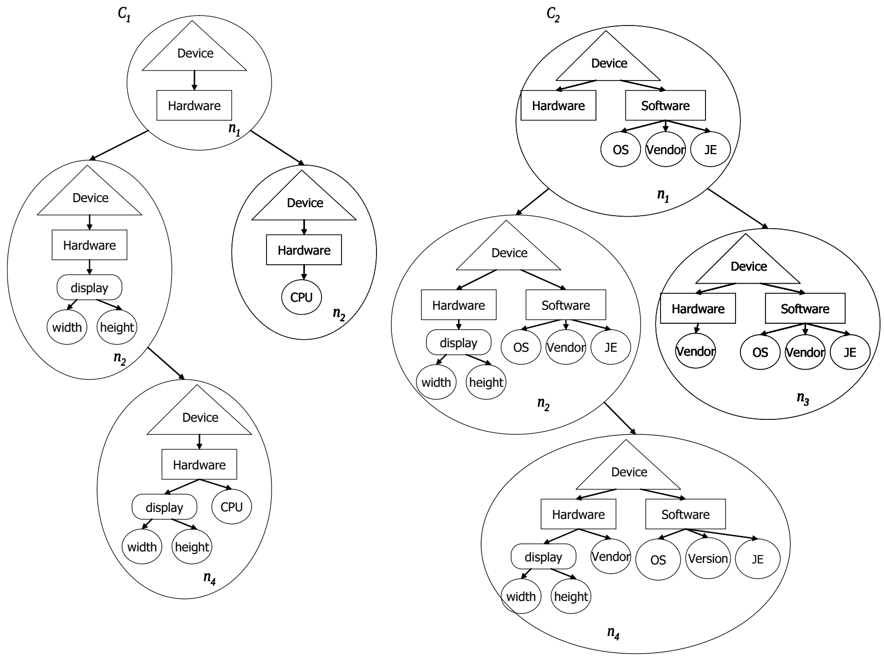


Fig. 17. Resulting Clusters from an Initialization.

tion C generated by the CM, composing C with the request, translated by the HTTP Server, and encoding the resulting one in specific languages.

Within the Context manager, three modules of adaptation have been currently implemented, which are devoted to the management of the device capabilities, the user preferences and network quality rates, respectively. CM includes a *Profile Interpreter (PI)* devoted to capture and translate a context into a uniform formalism and produce a set of profiles. In [7] we have addressed this problem. The profiles for the device coordinate has been built by using the WURFL (*Wireless Universal Resource File*) configuration database (freely available at: `http://wurfl.sourceforge.net/`). This database includes information about capabilities and features of a large set of wireless devices currently available on the market. *Qualitative* preferences [41] are used to represent the user profiles. The selection of contents based on qualitative preferences makes use of a special technique that we have studied elsewhere [42]. The network diagnostic is achieved by sending several test packets through the Packet Internet Groper (PING), and measuring the PING times (e.g. minimum, maximum, and average times). A set of *Dimension Adapters (DA)* are responsible to generate the configuration that matches with the respective incoming profile by using the described clustering approach. Each DA is supported by a repository of adaptation rules organized by means of adaptation clusters. The *Adaptation Coordinator* is responsible to compose the configurations, resulting from the DAs, by using the composition operator $\oplus$, defined in [6].

FAWIS specifies a WIS using several models, each one describing a different aspect of the system. We choose to implement the models in RDF(S), and their instances in RDF. The benefits of using RDF(S) are manifolds: application interoperability, direct availability of Web data appropriately annotated, reuse of existing vocabularies, future-proofing applications for which the design models evolve, etc. For example, RDF flexibility (e.g., loose schema definition) enables one to easily extend a model with other aspects. We make use of the User Agent Profile (UAProf), a Composite Capability/Preference Profiles (CC/PP) vocabulary [43], for modeling device capabilities and user preferences. The RDF(S) inheritance mechanism turned out to be enormously valuable for reusing models or adapting them for particular situations. In order to extract data from RDF models we use SPARQL [44] (with some additional primitives), a very expressive query language that exploits the full RDF(S) semantics represented in a model. Besides the standardization of model specifications, RDF(S) allows a more compact representation than the relational or XML representations by using inference rules to compute implicit information. FAWIS makes use of Jena, a Java framework for building RDF based applications (`http://jena.sourceforge.net/`).

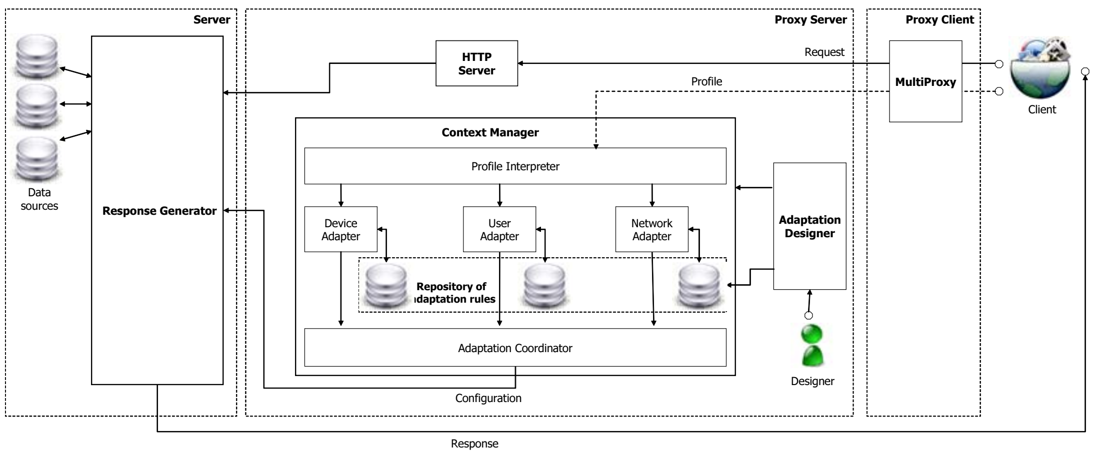


Fig. 18. The architecture of FAWIS.

7 A case study

7.1 Analysis of E-commerce sites

The described approach has been implemented in the FAWIS and experimented with a e-commerce application (`http://mais.dia.uniroma3.it/FAWIS/demo.htm`). The resulting Web site contains sections with news, articles and settings. Each subscribed user can build his own personal journal, containing preferred news and articles. On the settings side users can model and modify several context information and in particular the look of the interface, fill in and change their personal information about their interests, through the MultiProxy module. Figure 19 shows a snapshot of the module at work, on the left side it captures the request of the client (a typical HTTP request) and on the right side it proposes the setting tab for the user. Two different users can give completely different recommendations on the same page. Figure 20 illustrates the same Home Page of the Website at two different levels of details: a typical Desktop rendering, at the bottom side, and a typical Mobile Phone rendering, at the top side. The most important section of the Web site is the personalization of browsing. A typical page contains a description of news on several genres (e.g. Sport, Economics and so on). The system analyzes the Web site navigation of registered users. Figure 21 shows two different distributions of news in the page for the same registered user, due to different navigation sessions (e.g. in the top of the Figure there are news of various genres, in the bottom only the most consulted news of genre Sport).

7.2 Experimental Results

We have tested the effectiveness and the efficiency of the clustering approach illustrated in this paper, by using FAWIS in several practical cases. On the server side, we have used an Apple computer xServer, equipped with a Intel Core 2 Duo 1.86 Ghz processor, a 4 GB RAM, and a 500 GB HDD Serial ATA. The Web site used in the tests collects a repository of around 200,000 news items, many of which containing multimedia contents. On the client side, the Web site has been accessed by several clients.

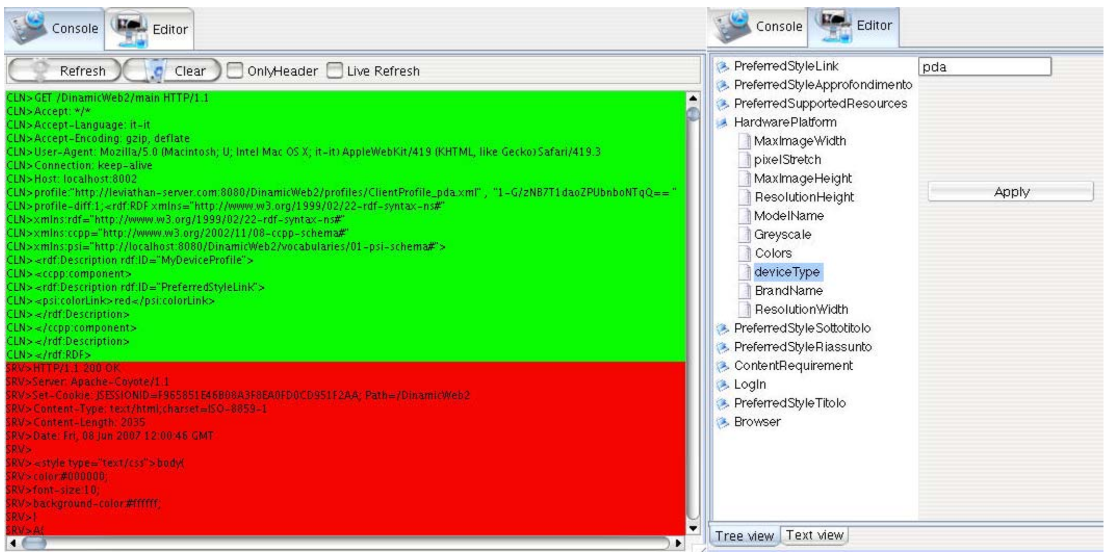


Fig. 19. The MultiProxy at work.

We have measured the medium response time for several profiles of devices, networks and user preferences. The devices have been classified and grouped in three main categories: mid-range desktops, PDAs, and cellular phones. After the identification of such classes, we have selected representatives of each class by choosing among the devices currently available on the market and produced by the different companies (such as Nokia, Siemens, Motorola, Sony Ericsson, Samsung, HP, Apple). We have then generated a large number of profiles for virtual users with different preferences about the content (selection of interesting news), navigation (the most popular Web pages) and presentation features. Finally, different networks with varying rates of service have been used. In this case we have classified the alternatives according to the bandwidth and delay values. Our top ontology involves three profile domains: device, user and network characterized by 20, 31 and 6 attributes, respectively. The

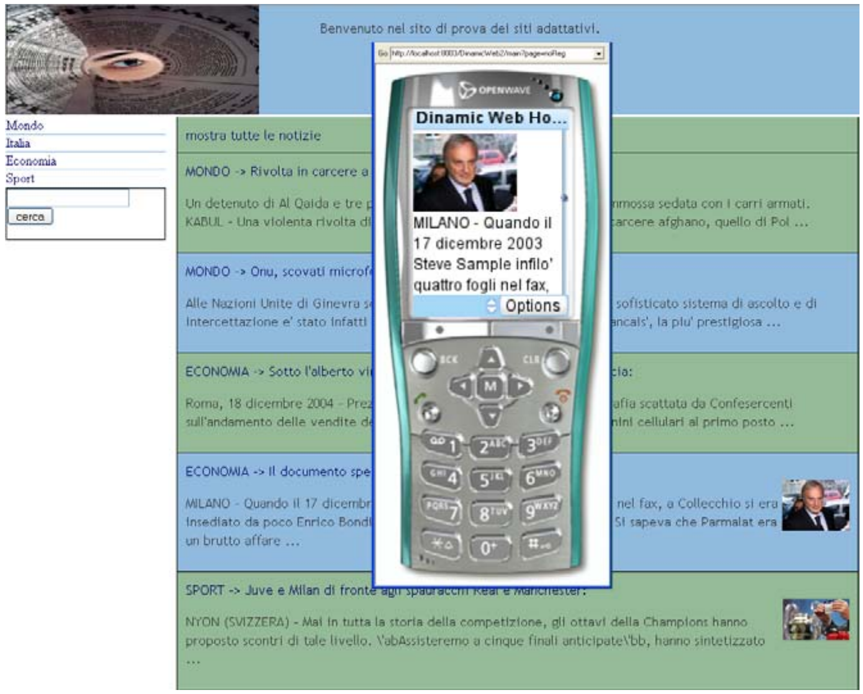


Fig. 20. The Website for two different users.

experiment are executed on different repositories of adaptation clusters, varying the threshold σ. Each repository contains around 10.000 profiles associated with the corresponding adaptations.

The results of our experiments are synthesized in three diagrams, as follows.

- Figure 22 reports the time in minutes (vertical axis) needed for clustering 10.000 profiles with respect to an increasing value of the threshold σ (horizontal axis). This is indeed a start-up operation: for small values of σ the operation is less expensive, because the clusterization is trivially a scan of the initial set of profiles, with no meaningful results. For large values of σ the clusterization is more expensive but relevant results are obtained (as described in the other diagrams).
- Figure 23 reports the number of clusters (vertical axis) with respect to an increasing value of the threshold σ (horizontal axis). This diagram shows different clusterization of the 10.000 items: it turns out that the number of final clusters decrease with the increase of the value of the threshold.
- Figure 24 reports the medium response time (in seconds) of the system in logarithmic scale (vertical axis) as the number of profiles increases (horizontal axis) in three cases of clustering based on different thresholds. In the case of absence of clusters ($\sigma = 0$) the response time rapidly increases with the number of profiles. This cost is strongly reduced when the clustering is used with a reasonable value of the threshold ($\sigma = 0.35$) and raises slowly as the number of profiles increases ($\sigma = 0.5$).

Fig. 21. The same page with different news for the same user.

7.3 Efficiency and Effectiveness

In general, the results obtained so far are good. The efficiency and effectiveness of the approach are supported by the capability demonstrated by the system to generate satisfactory results for different contexts, even not fixed in advance. This is due to the ability of the approach to select configurations for "compatible" contexts, as described in Section 4.

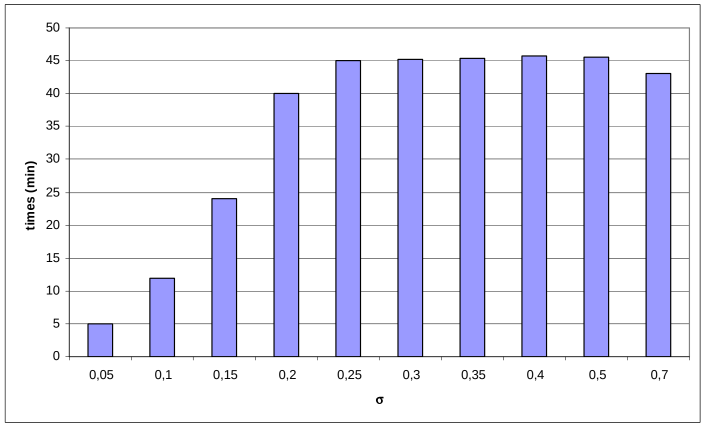


Fig. 22. Time in minutes needed for clustering 10.000 profiles with respect to the threshold.

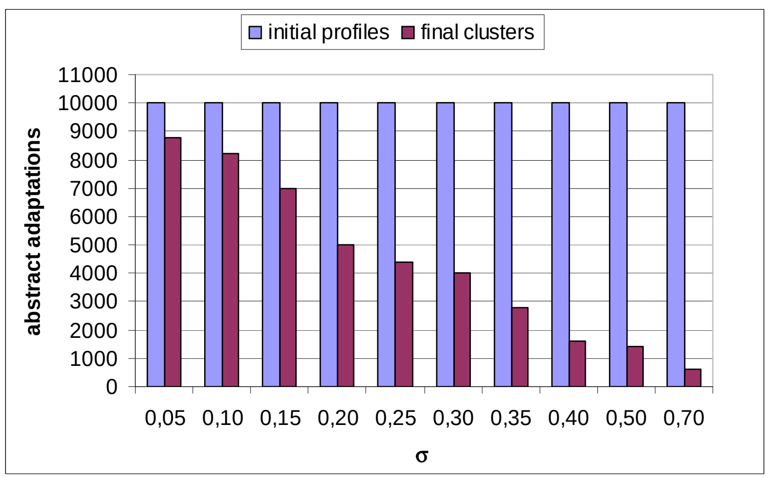


Fig. 23. Number of clusters with respect to the threshold.

The effectiveness is estimated in terms of a right design of the abstract adaptations. As said in Section 5, the number of clusters obtained is strictly related to the design of the adaptations, which are in turn related to the profiles in a cluster. A right design implies a good and easy clustering, otherwise the difficult grouping of profiles implies an inaccurate adaptation design. To this aim the threshold is a feedback. The diagram in Figure 22 demonstrates the complexity of the start-up process to generate a relevant set of clusters ($\sigma = 0.35$ with a time of 45 min). The diagram in Figure 23 illustrates the relevant decrease of search space: for $\sigma = 0.35$ we obtain 2800 balanced clusters, reducing the search space of 98% (about 20 profiles for cluster in average).

The efficiency is evaluated in terms of the responsiveness of the system. The diagram in Figure 24 compares the response times in the case of presence/absence of clustering, varying the threshold σ. It supports the efficiency of the approach. In presence of a balanced clustering ($\sigma = 0.35$) the response times present an upper-bound of 0.6 sec (-0.2 in logarithmic scale). In absence of clustering ($\sigma = 0$) the diagram grows with an exponential rate, around values of 2.3 min (2.2 in logarithmic scale).

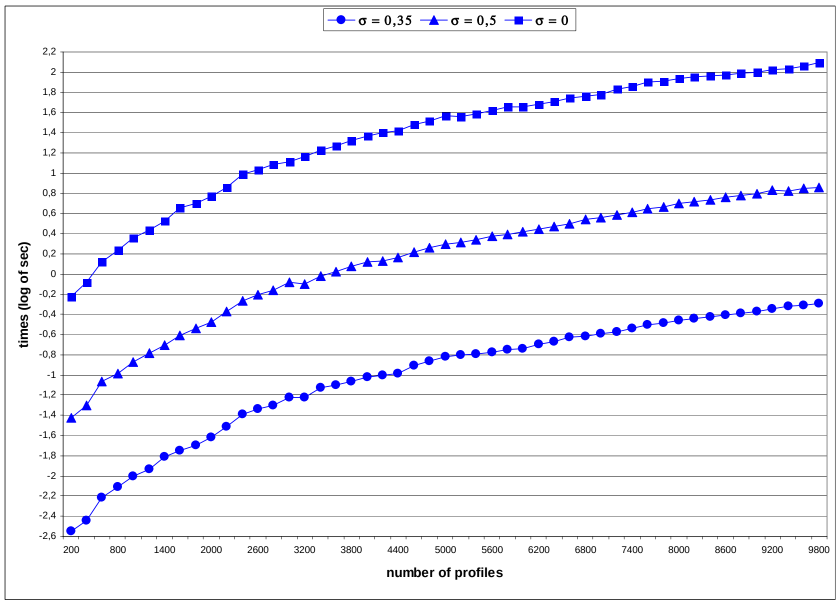


Fig. 24. Medium response time with respect to the number of profiles for three different thresholds.

8 Conclusions and Future Work

In this paper we have presented a novel cluster-based approach to the efficient adaptation of content delivery in Web Information Systems. Different contexts and orthogonal requirements of adaptation, possibly not fixed in advance, can be taken into account in this process. The approach is based on a general adaptation methodology that consider the context as a set of autonomous components (such as, the device, the user preferences, the network, the location) called profiles and generates a response to a user request in a form that meet the requirements of the various profiles of a user context. In order to make this process more efficient and effective, we have proposed an approach in which profiles are clusterized on the base of their similarity. The approach has been implemented in a practical tool and number of practical experiments done with this tool have shown that this mechanism strongly improve the adaptation process, especially in a e-commerce scenario where the number of profiles to take into account is huge and fast responsiveness to rapidly changing requirements of adaptation is strongly needed. Clusterization is widely used in Web usage mining but, to our knowledge, this is the first cluster-based optimization approach proposed for Web adaptation. A great advantage came from the adoption of Semantic Web principles both at the logical level, where ontologies are used to model context information, and at the practical level, where Sematic Web enabling technologies have been largely used.

The results presented in this paper are subject of further conceptual and practical investigation. From a conceptual point of view, we are currently investigating other types of classifications of profiles that can be exploited to optimize the adaptation process. In particular we are investigating alternative representations of contexts and special index structures for accessing more efficiently stored profiles and configurations. From a practical point of view we are improving the functionality and the usability of the tool.

Acknowledgements

We would like to thank the anonymous referees for their helpful comments.

References

1. T. Isakowitz, M. Bieber, and F. Vitali (1998), *Web Information Systems*, in Communications of the ACM 41(1):78–80.
2. M. Hinz, Z. Fiala and F. Wehner (2004), *Personalization-Based Optimization of Web Interfaces for Mobile Devices*, in Proceedings of 6th International Symposium on Mobile Human-Computer Interaction (MobileHCI 2004), Vol. 3160, Lecture Notes in Computer Science, Springer, pp. 204-215.
3. R. Jan, C. Lin and M. Chern (2006), *An optimization model for Web content adaptation*, in International Journal of Computer and Telecommunications Networking 50(7): 953–965.
4. M. Koutri, N. Avouris and S. Daskalaki (2005), *A survey on Web usage mining techniques for Web-based adaptive hypermedia systems*, in S.Y.Chen and G.D.Magoulad (ed), Adaptable and Adaptive Hypermedia Systems, IRM Press, pp. 125-149.
5. B. Krishnamurthy and C. E. Wills (2002), *Improving Web Performance by Client Characterization Driven Server Adaptation*, in Proceedings of 11th International World Wide Web Conference (WWW 2002), ACM, pp. 305-316.
6. R. De Virgilio, R. Torlone, and G.-J. Houben (2006), *A rule-based approach to content delivery adaptation in Web information systems*, in Proceedings of 7th International Conference on Mobile Data Management (MDM 2006), IEEE Computer Society, pp. 21-24.
7. R. De Virgilio and R. Torlone (2006), *Modeling Heterogeneous Context Information in Adaptive Web Based Applications*, in Proceedings of 6th ACM International Conference on Web Engineering (ICWE 2006), ACM, pp. 56-63.
8. S. Guha, R. Rastogi, and K. Shim (1999), *ROCK: a robust clustering algorithm for categorical attributes*, in Proceedings of 15th International Conference on Data Engineering (ICDE 1999), IEEE Computer Society, pp. 512-521.
9. A. K. Jain and R. C. Dubes (1988), *Algorithms for Clustering Data*, Prentice Hall Publishers.
10. R. T. Ng and J. Han (1994), *Efficient and Effective Clustering Methods for Spatial Data Mining*, in Proceedings of the 20th International Conference on Very Large Data Bases (VLDB 1994), Morgan Kaufmann, pp. 144-155.
11. O. Zamir, O. Etzioni, O. Madani and R.M. Karp (1997), *Fast and intuitive clustering of Web documents*, in Proceedings of the 3th Int Conference on Knowledge Discovery and Data Mining (KDD 1997), AAAI Press, pp. 287-290.
12. D. Guillaume and F. Murtagh (2000), *Clustering of XML Documents*, Computer Physics Comm.,

vol. 127, no.2-3, pp.215-227.
13. W. Retschitzegger, and W. Schwinger (2000), *Towards Modeling of DataWeb Applications - A Requirements' Perspective*, in Proceedings of the Americas Conference on Information Systems (AMCIS 2000).
14. F. Garzotto, P. Paolini, and D. Schwabe (1993), *HDM - A Model-Based Approach to Hypertext Application Design*, in ACM Transaction on Information Systems 11(1): 1-26.
15. T. Isakowitz, E. A. Stohr, and P. Balasubramanian (1995), *RMM: A Methodology for Structured Hypermedia Design*, in Commun. ACM 38(8): 34–44.
16. D. Schwabe, and G. Rossi (1995), *The Object-Oriented Hypermedia Design Model*, in Commun. ACM 38(8): 45–46.
17. F. Frasincar, G. J. Houben, and R. Vdovjak (2002), *Specification Framework for Engineering Adaptive Web Applications*, in Proceedings of 11th International Conference on World Wide Web (WWW 2002), ACM, pp. 117-123.
18. I. Garrigos, J. Gomez, and C. Cachero (2003), *Modelling Dynamic Personalization in Web Applications*, in Proceedings of the 3th International Conference of Web Engineering (ICWE 2003), Vol. 2722, Lecture Notes in Computer Science, Springer, pp. 472-475.
19. S. Casteleyn, O. De Troyer, and S. Brockmans (2003), *Design Time Support for Adaptive Behavior in Web Sites*, in Proceedings of 18th ACM Symposium of Applied Computing (SAC 2003), ACM, pp. 1222-1228.
20. D. Schwabe, R. de Almeida Pontes, and I. Moura (1999), *OOHDM-Web: an environment for implementation of hypermedia applications in the WWW*, in ACM SIGWEB Newsletter 8(2):18-34.
21. S. Ceri, F. Daniel, V. Demaldé, and F. M. Facca (2005), *An approach to user-behavior-aware Web applications*, in Proceedings of 5th International Conference on Web Engineering (ICWE 2005), Vol. 3579, Lecture Notes in Computer Science, Springer, pp. 417-428.
22. R. E. Filman, T. Elrad, S. Clarke, and M. Aksit (2004), *Aspect-Oriented Software Development*, Addison-Wesley.
23. P. Brusilovsky (1996), *Adaptive Hypermedia: An attempt to analyze and generalize*, in Proceedings of 1th International Conference on Multimedia, Hypermedia and Virtual Reality, Vol 1077, Lecture Notes in Computer Science, Springer, pp. 288-304.
24. G. Kappel, W. Retschitzegger and W. Schwinger (2001), *Modeling Ubiquitous Web Applications: The WUML approach*, in International Workshop on Data Semantics in Web Information Systems (DASWIS 2001), Vol. 2465, Lecture Notes in Computer Science, Springer, pp. 183-197.
25. N. Koch and A. Kraus (2003), *Towards a Common Metamodel for the Development of Web Applications*, in Proceedings of the 3th International Conference on Web Engineering (ICWE 2003), Vol. 2722, Lecture Notes in Computer Science, Springer, pp. 497-506.
26. P. Dolog and W. Nejdl (2003), *Using UML and XMI for Generating Adaptive Navigation Sequences in Web-Based Systems*, in Proceedings of the 6th International Conference on The Unified Modeling Language, Modeling Languages and Applications (UML 2003), Vol. 2863, Lecture Notes in Computer Science, Springer, pp. 205-219.
27. J. Han and M. Kamer (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufmann.
28. P. Brusilovsky and M. T. Maybury (2002), *From adaptive hypermedia to the adaptive Web*, in Communications of the ACM 45(5):30–33.
29. B. Vassiliadis, C. Makris, A. Tsakalidis, N. Bogonikolos (2003), *User Modelling for Adapting and Adaptable Information Retrieval*, in Journal of Applied System Studies, 4(1): 43-63.
30. W3C Working Group on Device Independence (2003), *Device Independence Principles*, in World Wide Web Consortium, Internet document: `http://www.w3.org/TR/di-princ/`.
31. T. A. Wiggerts (1997), *Using clustering algorithms in legacy systems remodularization*, in Proceedings of the 4th Working Conference on Reverse Engineering (WCRE 1997), IEEE Computer Society, pp. 33-43.
32. C. Cappiello, M. Comuzzi, E. Mussi and B. Pernici (2006), *Context Management for Adaptive Information Systems*, in Electronic Notes in Theoretical Computer Science 146(1): 69-84.

33. A. Held, S. Buchholz and A. Schill (2002), *Modeling of Context Information for Pervasive Computing Applications*, in Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002). Internet document: `http://www.rn.inf.tu-dresden.de/uploads/SCI2002-paper512JH.pdf`.
34. X. H. Wang, D. Zhang, T. Gu, H. K. Pung (2004), *Ontology Based Context Modeling and Reasoning using OWL*, in Proceedings of the 2th PerCom Workshop on Context Modeling and Reasoning (CoMoRea 2004), IEEE Computer Society, pp. 18-22.
35. D. L. McGuinness and F. van Harmelen (2004), *OWL Web Ontology Language Overview*, World Wide Web Consortium, Internet document: `http://www.w3.org/TR/2004/REC-owl-features-20040210`.
36. R. De Virgilio and R. Torlone (2005), *A General Methodology for Context-Aware Data Access*, in Proceedings of the 4th International ACM Workshop on Data Engineering for Wireless and Mobile Access (Mobide 2005), ACM, pp. 9-15.
37. S. Ceri, P. Fraternali, A. Bongio , M. Brambilla, S. Comai, and M. Matera (2003), *Designing Data-Intensive Web Applications*, Morgan Kaufmann.
38. A. Al-bar and I. Wakeman (2001), *A Survey of Adaptive Applications in Mobile Computing*, in Proceedings of 21th International Conference on Distributed Computing Systems Workshops (ICDCSW 2001), IEEE Computer Society, pp. 246-256.
39. W. Lian, D. W. Cheung, N. Mamoulis, S. Yiu (2004), *An Efficient and Scalable Algorithm for Clustering XML Documents by Structure*, in IEEE Transactions on Knowledge and Data Engineering 16(1): 82-96
40. H. Bunke and K. Shearer (1998), *A graph distance metric based on the maximal common subgraph*, in Pattern Recognition Letters 19(3-4): 255-259.
41. W. Kießling (2002), *Foundations of preferences in database systems*, in Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002), Morgan Kaufmann, pp. 311-322.
42. R. Torlone and P. Ciaccia (2002), *Which are my Preferred Items?*, in Proceedings of the 2th International Workshop on Recommendation and Personalization in e-Commerce (RPeC 2002). Internet document: `http://torlone.dia.uniroma3.it/pubs/repec02.pdf`.
43. C. Kiss (2007), *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0*, World Wide Web Consortium, Internet document: `http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/`.
44. E. P. Hommeaux and A. Seaborne (2007), *SPARQL Query Language for RDF*, World Wide Web Consortium, Internet document: `http://www.w3.org/TR/2007/WD-rdf-sparql-query-20070326/`.