# A framework for the Management of Context Data in Adaptive Web Information Systems

Roberto De Virgilio and Riccardo Torlone
Università Roma Tre, Rome, Italy
{devirgilio,torlone}@dia.uniroma3.it

## Abstract

*Context-awareness is considered today a desirable facility of modern Web Information systems, given the large variety of non-traditional client devices used to access such applications. In this scenario, a fundamental requirement is the ability to capture and manipulate, in a flexible way, diverse context information, such as, among many others, the device capabilities, the preferences of the user, the network QoS, and the location. In this paper we propose a general framework supporting the representation and management of a large variety of context information. To this end, we first introduce a general data model that embeds the basic constructs commonly used to represent context information. We then define a number of basic primitives for the manipulation of context data. We finally describe how this framework can be profitably used to support different well-known applications where context management can provide an important add-on.*

## 1 Introduction

The increasing popularity of mobile devices, such as laptops, mobile phones, and personal digital assistants is enabling new classes of Web applications targeting environments characterized by being dynamic, mobile, reconfigurable, and personalized spontaneously. These applications and their targeted environments raise challenging problems for developers, as they have to be aware of the variations in the context of execution (such as the location, the time, the users' activities, and the devices' capabilities) in order to tune and adapt their behavior and functionality. In this framework, it is widely recognized that the management of context information is a fundamental requirement to take into account effectively the limited resources of mobile systems, to select data relevant to the user, to improve the interoperability with the environment, and, in general, to make the interaction with the system truly adaptive to highly change scenarios of use. These "adaptive" Web information systems have to provide dynamic behavior based on run-time interpretation of different models. This scenario changes the role of context information and semantics as compared to traditional information systems [13, 16], as now the physical environment immediately affects and interacts with the processing of data and communication. Unfortunately, current technologies do not fully support flexible and self-adapting models based on context. For example, if a mobile user today wants to use the computing resources of a new environment, he/she has to obtain the necessary information, assess it (format, semantics) and figure out manually how to continue his/her activities with the local resources of that new environment. This is unacceptable in pervasive computing environment and neglects the advances which have been made in other research domains dealing with context information and semantics.

In the plethora of context representations, different models exist, often just small variation of other ones. In particular the most relevant approaches to Web Information Systems Adaptation [4, 9, 12, 14, 15] identify a context as a set of *profiles* [19]. The majority of these techniques provide specific solutions that are suited for a particular class of predefined adaptation requirements generating rapidly the adaptation on the basis of certain aspects of the context, often considered a a trivial set of input values. Moreover there is still a lack of uniformity among the different approaches, that makes most of them unusable in practice. To this aim, in [6] we have provided a rule based technique to Web Adaptation and in [5] we have proposed and investigated the *General Profile Model* (GPM), a conceptual model for the uniform description of the various aspects of a context, focusing on the translation of context data between different formats. However in many application scenarios, more powerful manipulation capabilities of context data are needed. These includes: the integration of different (possibly in conflict) context requirements coming from various sources and the effective storage and efficient retrieval of the context requirements to select the adaptation that best fits them (i.e. e-commerce applications). Therefore, we need an explicit representation of context information and a rich set of operations to manipulate them. We call this overall capability *Context Management*.

This paper proposes a general framework for Context Management in Adaptive Web Applications. This framework provides two main contributions. First, it argues that modern adaptive Web applications need a support for a broad spectrum of context data available through the network. Therefore, by extending GPM, we introduce a middleware data model that serves as a formal foundation to facilitate the representation and maintenance of heterogeneous context data. We claim that this simple model is general enough to capture the vast majority of the formats of interest. Such description permits to define different *Profile Models* and *Profile Mappings*. The former describe the primitives involved and the structure of a context model, the latter specify the relationship between different contexts. The other relevant contribution of the paper is an algebra of operators to execute several manipulations of context information. The operators can be composed to support several problems of context management. To illustrate the pervasiveness and scope of the framework, we provide some examples that arise in various Web application scenarios.

The rest of the paper is organized as follows. In Section 2, we illustrate our context modeling approach illustrating GPM and the notions of Profile Model and Profile Mapping. In Section 3 we describe the algebra of operators to manipulate context information. Finally, in Section 4 we illustrate an implementation of the framework, in Section 5 we show several application scenarios, in Section 6 we discuss related work and in Section 7 we draw some conclusions and sketch future work.

## 2 Context Modeling

In this section we briefly present GPM, a *metamodel* that includes all constructs of other context models. Respect to [5] we introduced new constructs, in particular to manage constraints. Moreover, based on such description we then illustrate the special notions of *Profile Model*, representing the particular model of context data, and *Profile Mapping* that describes how two contexts are related to each other (i.e. useful to operate integrations between context data).

### 2.1 General Profile Model

A *general profile* is a description of an autonomous aspect of the context into which the Web site is accessed and that should influence the structuring and presentation of its contents [22]. Examples of profiles are descriptions of the user, the device, the location, and so on.
GPM presents a quite limited set of constructs, that we call *basic primitives*, to describe a conceptual representation of

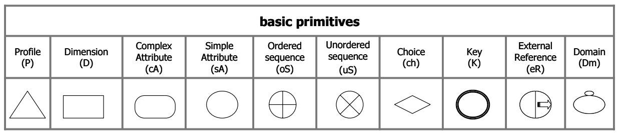


**Figure 1. GPM basic primitives**

a context, as shown in Figure 1. Principal constructs are the *dimension* and the *attribute*. A *dimension* is a property that characterizes a profile. Each dimension is described by means of a set of *attributes*. Attributes can be *simple* or *composite*. A simple attribute has a *domain* of values associated with it (printable values such as string, integer, boolean and so on), whereas a composite attribute has a set of (simple or composite) attributes associated with it. It is possible to represent *ordered* and *unordered sequences* of attributes and a *choice* of a set of attributes, whose instances can be chosen among instances of the attributes (for instance `rdf:Alt` of an RDF file). It is possible to distinguish different roles for a simple attribute: it can be a *key* or an *external reference* to a component of a profile. The *cardinality* is expressed as a pair of integer values (*Min*,*Max*) that corresponds to a primitive whose instances are sets of instances of the primitive associated with it (these sets must have a cardinality included between *Min* and *Max*).

Let us 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*. We denote by $D : Y(X)$ a *dimension schema*, where $D \in \mathbf{D}$, $X = A_1, \ldots, A_n \in \mathbf{A}$ and $Y \in \{\oplus, \otimes, \diamond\}$. If $Y = \oplus$ ($\otimes$) then $X$ denotes an ordered (unordered) sequence of the attributes in $X$, otherwise if $Y = \diamond$ then $X$ denotes a choice on the set of attributes in $X$. If $A_i$ ($1 \leq i \leq n$) is a simple attribute, then it is associated with a set of values (the domain), otherwise, it has associated with $Y(A_{i_1}, \ldots, A_{i_k})$ that denotes an ordered ($Y = \oplus$), unordered ($Y = \otimes$) sequence or a choice ($Y = \diamond$) of (sub)attributes.

**Definition 1 (Profile and context)** *Given a set of dimensions $D_1, \ldots, D_n$ over a set of attributes $A_{i,1} \ldots A_{i,k_i}$ ($1 \leq i \leq n$) respectively, a (general)* profile *over $D_1, \ldots, D_n$ is function that associates with each simple attribute of every dimension a value (or a set of values) taken from its domain. A* context *is a collection of profiles.*

An *instance* of a profile results by associating with each simple attribute of every dimension a value (or a set of values) taken from its domain, while a *schema* of a profile results by associating with each simple attribute of every dimension a particular domain. Therefore, a context schema is a set of profile schemes, and a context instance is a set of profile instances. As an example, Figure 2 reports a graphical representation for the context schema of

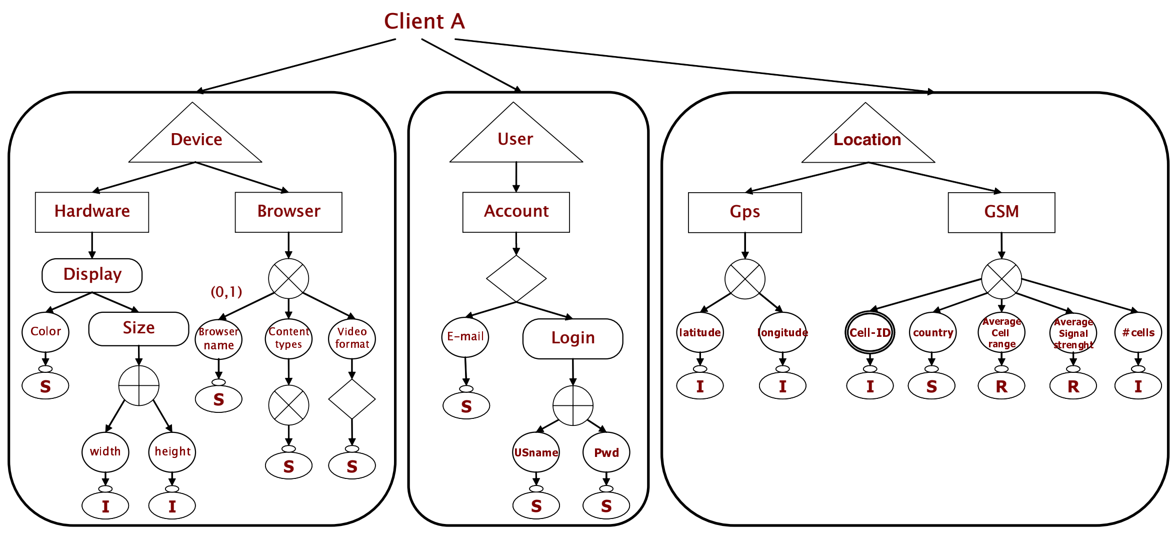


**Figure 2. Example of a context schema**

a client $A$ composed by profile schemes for the device, the user and the location. The profile schema for a client device is represented by means of the `hardware` and `browser` dimensions. In turn, the `hardware` dimension can be described by means of a composite attribute like `display`, composed by another composite attribute `size` as an ordered sequence of simple attributes `width` and `height`. We indicate the basic type associated inside the domain (for instance $I$ means integer, $R$ real and $S$ string). The `content types` attribute is associated to the integer domain by an unordered sequence: this means that the profile associates to the attribute a set of integer values (for instance the `rdf:Bag` in a RDF file). Similarly, the `video-format` attribute is associated to a choice of possible values (the `rdf:Alt` in a RDF file). The user profile schema presents the dimension `account`, described by a choice between the simple attribute `e-mail` (so an access without a registration) or the composite attribute `login`, composed by the simple attributes `username` and `password` (so an access with a registration), and so on. The location profile schema presents the dimension `GSM` to characterize location information from the GSM cells: each cell has a unique ID, expressed by the key `Cell-ID`. Figure 3 reports three profile instances for the profile schema of the device.

An important aspect of the formalism is that different profiles (schemes or instances) can be compared making use of a subsumption relationship $\lhd$. Intuitively, given two profiles (schemes or instances) $P_1$ 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-

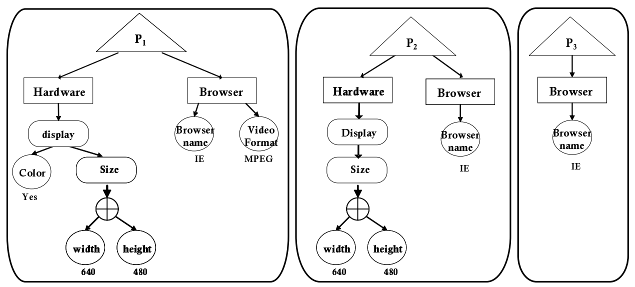


**Figure 3. An example of subsumption between profile instances**

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.

**Definition 2 (Subsumption of Profiles)** *Given two profiles (schemes or instances) $P_1$ 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 profile instances reported in Figure 3, 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.

## 2.2 Profile Models

Through our formalism, we want to describe the model of a profile by means of the constructs involved and how these constructs are composed. Since a profile is principally structured in terms of dimensions and attributes, the respective model has to describe how a dimension is composed in attributes both by means of minimum and maximum cardinality and by the *level* of composition. Iteratively we consider the other possible compositions such as simple attributes in different domains, sequences of components and so on. To this aim we first introduce the notion of *weight function*. Given two basic constructs $a$ and $b$ of GPM the weight $\omega(a,b)$ returns $\{c : (min, max)\}$, where $c$ is the level of composition of $b$ with respect to $a$ and $(min, max)$ the minimum and maximum cardinality of $b$ with respect to $a$ in each level of composition . The function $\omega(a,b)$ returns the null value $\emptyset$ if there is no composition. In a weight, admissible values for $c$ are positive integer in the range $[1,n]$ ($n$ if infinite), and the cardinalities ($min$ and $max$) are in $[0,n]$. It is possible to establish an ordering between two weights $\omega_1(a,b) = c_1 : (min_1, max_1)$ and $\omega_2(a,b) = c_2 : (min_2, max_2)$ such that $\omega_1(a,b) \leq \omega_2(a,b)$ if $c_1 \leq c_2$ and $min_1(max_1) \subseteq min_2(max_2)$, where it is $1 \subset 0 \subset n$. A *Profile Model* is then defined as follows.

**Definition 3 (Profile Model)** *We consider a Profile Model $PM$ as a couple $\langle L, \omega \rangle$ where $L$ is the set of basic constructs of our GPM involved in the Profile Model, such as $\{ \triangle, \square, \bigcirc, \ldots \}$, and $\omega$ a weight function.*

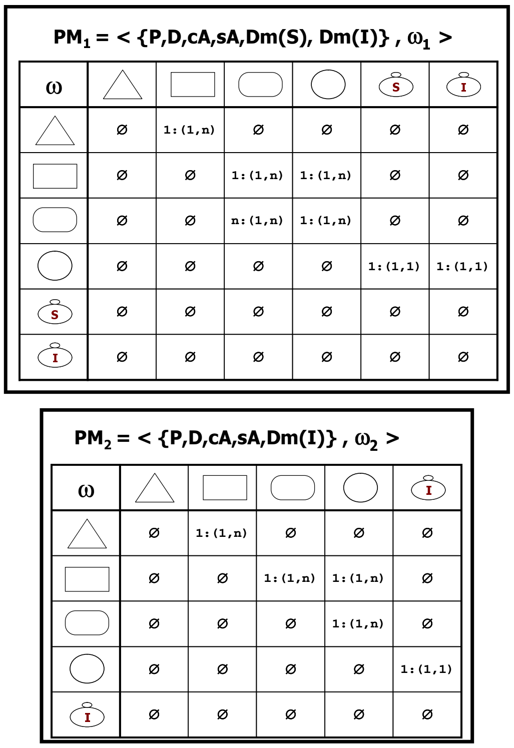
For instance, Figure 4 shows the graphical representation of a Profile Model $PM_1$. The set $L$ contains the basic primitives involved, and the weights assigned by $\omega$ are represented in a matrix. In $PM_1$ each dimension can be composed by simple attributes of string or integer values, or by composite attributes. Each composite attribute can be composed by simple attributes of string or integer values or, recursively, by composite attributes of the same type. The function $\omega$ assign to each couple of primitives the weight $c : (1,n)$. Each component is composed by $n$ other components (at least 1) along $c$ levels. For instance $\omega(cA, cA)$ says that a composite attribute is composed by other ones (along $n$ levels, so infinitive times). For instance $PM_1$ describes the profile schema of device of Figure 2.
As for profiles, we can define the subsumption relationship $\lhd$ between Profiles Models.

**Definition 4 (Subsumption of Profile Models)** *Given two Profile Models $PM_1 = \langle L_1, \omega_1 \rangle$ and $PM_2 = \langle L_2, \omega_2 \rangle$, we say that $PM_1$ is subsumed by $PM_2$, $PM_1 \lhd PM_2$, if $L_1 \subseteq L_2$ and for each $(a,b) \in L_1$ then $\omega_1(a,b) \leq \omega_2(a,b)$*

As an example, given the Profile Models reported in Figure 4, we have that $PM_2 \lhd PM_1$.

## 2.3 Profile Mappings

Given two profile schemes $PS_1$ and $PS_2$, a *morphism* over $PS_1$ and $PS_2$ is a binary relation over the components


$PM_1$ = < {P,D,cA,sA,Dm(S), Dm(I)} , $\omega_1$ >

| ω | △ | □ | ▢ | ◯ | S | I |
|---|---|---|---|---|---|---|
| △ | ∅ | 1:(1,n) | ∅ | ∅ | ∅ | ∅ |
| □ | ∅ | ∅ | 1:(1,n) | 1:(1,n) | ∅ | ∅ |
| ▢ | ∅ | ∅ | n:(1,n) | 1:(1,n) | ∅ | ∅ |
| ◯ | ∅ | ∅ | ∅ | ∅ | 1:(1,1) | 1:(1,1) |
| S | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |
| I | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |

$PM_2$ = < {P,D,cA,sA,Dm(I)} , $\omega_2$ >

| ω | △ | □ | ▢ | ◯ | I |
|---|---|---|---|---|---|
| △ | ∅ | 1:(1,n) | ∅ | ∅ | ∅ |
| □ | ∅ | ∅ | 1:(1,n) | 1:(1,n) | ∅ |
| ▢ | ∅ | ∅ | ∅ | 1:(1,n) | ∅ |
| ◯ | ∅ | ∅ | ∅ | ∅ | 1:(1,1) |
| I | ∅ | ∅ | ∅ | ∅ | ∅ |

**Figure 4. An example of Profile Models**

(dimension or attribute) of the two profile schemes. More precisely, we define a morphism as follows

**Definition 5 (Morphism of profiles)** *Given two profile schemes $PS_1$ and $PS_2$, a morphism $\mathcal{M}$ is a set of pairs (s,t) where s is a component (dimension or attribute) of $PS_1$ and t is a component of $PS_2$.*

We use the traditional notion of morphism [1] such as to define a set of correspondences between two profile schemes. Given a morphism $\mathcal{M}$ between two profile schemes $PS_1$ and $PS_2$, for each pair (s,t) of $\mathcal{M}$ we define a pair $(PS_1^{[s]}, PS_2^{[t]})$ where $PS_1^{[s]}$ is the profile schema representing the path from the root of $PS_1$ to the component s (of $PS_1$) and $PS_2^{[t]}$ is the profile schema representing the path from the root of $PS_2$ to the component t (of $PS_2$). Through particular *combinational functions*, we define a *Profile Mapping* as

**Definition 6 (Profile Mapping)** *Given two profile schemes $PS_1$ and $PS_2$, a morphism $\mathcal{M}$ between $PS_1$ and $PS_2$ and a set of combinational functions* **F**, *a mapping* **M** *is a set of pairs $\langle [(PS_1^{[s_1]}, PS_2^{[t]}), (PS_1^{[s_2]}, PS_2^{[t]}), \ldots], \mathcal{F} \rangle$, where $(PS_1^{[s_i]}, PS_2^{[t]})$ is from each $(s_i, t)$ in $\mathcal{M}$ and $\mathcal{F}$ is a function of* **F** *such that $P_2^{[t]} = \mathcal{F}(P_1^{[s_1]}, P_1^{[s_2]}, \ldots)$.*

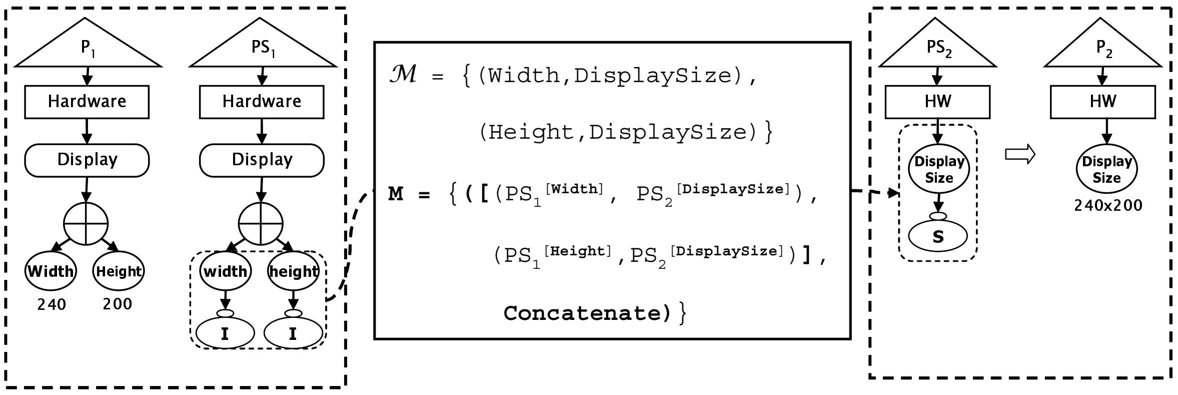


**Figure 5. An example of Profile Mapping**

Instead of representing the relationship as a set of pairs (of components), a Profile Mapping represents a set of objects (each of which can relate components in the profile schemes). This reification is often needed for satisfactory expressiveness. It allows us to attach custom semantics to a mapping. We can do this by having a combinational function for each object $m$ in a Profile Mapping, which is an expression whose variables include the objects that $m$ directly or indirectly references in $PS_1$ and $PS_2$. For example, in Figure 5 we could associate an expression with the morphism {*(Height,DisplaySize), (Width,DisplaySize)*} that says *DisplaySize* equals the *concatenation* of *Width* and *Height*, so $P_2^{[DisplaySize]} =$ **Concatenate**$(P_1^{[Width]}, P_1^{[Height]})$.

## 3 Context Management Algebra

We can define an algebra of operators to execute manipulations on profiles (schemes or instances), Profile Models and Profile Mappings.

### 3.1 Meet ($\sqcap$)

Intuitively, given two profiles $P_1$ and $P_2$, the *meet* ($\sqcap$) represents the greatest profile including the information in common between $P_1$ and $P_2$. The *meet* ($\sqcap$) of two Profile Models $PM_1$ and $PM_2$ represents the greatest Profile Model that is able to generate the productions (profiles) in common between $PM_1$ and $PM_2$.

**Definition 7 (Meet of Profiles)** *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 8 (Meet of Profile Models)** *Given two Profile Models $PM_1 = \langle L_1, \omega_1 \rangle$ and $PM_2 = \langle L_2, \omega_2 \rangle$, the meet of $PM_1$ and $PM_2$, denoted by $PM_1 \sqcap PM_2$, is a Profile Model $PM = \langle L, \omega \rangle$ where $L = L_1 \cap L_2$ and for each $(a, b) \in L$, $\omega(a, b) = min\{\omega_1(a, b), \omega_2(a, b)\}$ (otherwise $\emptyset$, if $\omega_1(a, b)$ or $\omega_2(a, b)$ is $\emptyset$)*

### 3.2 Join ($\sqcup$)

Given two profiles $P_1$ and $P_2$, the *join* ($\sqcup$) represents the least profile including all the information of both $P_1$ and $P_2$. The *join* ($\sqcup$) of two Profile Models $PM_1$ and $PM_2$ represents the least Profile Model that is able to generate all the productions (profiles) of $PM_1$ and $PM_2$.

**Definition 9 (Join of Profiles)** *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 $P \lhd P'$.*

**Definition 10 (Join of Profile Models)** *Given two Profile Models $PM_1 = \langle L_1, \omega_1 \rangle$ and $PM_2 = \langle L_2, \omega_2 \rangle$, the join of $PM_1$ and $PM_2$, denoted by $PM_1 \sqcup PM_2$, is a Profile Model $PM = \langle L, \omega \rangle$ where $L = L_1 \cup L_2$ and for each $(a, b) \in L$, $\omega(a, b) = max\{\omega_1(a, b), \omega_2(a, b)\}$ (otherwise $\omega_1(a, b)$ if $\omega_2(a, b) = \emptyset$, or $\omega_2(a, b)$ viceversa)*

### 3.3 Difference ($-$)

Given two profiles $P_1$ and $P_2$, the *difference* ($-$) represents the greatest profile including the information of $P_1$ not in common with $P_2$. The *difference* ($-$) of two Profile Models $PM_1$ and $PM_2$ represents the greatest Profile Model that is able to generate the productions (profiles) of $PM_1$ not in common with $PM_2$.

**Definition 11 (Difference of Profiles)** *The* difference *of two profiles $P_1$ and $P_2$, denoted by $P_1 - P_2$, is a profile $P$ such that, $P \lhd P_1$, $P \sqcap (P_1 \sqcap P_2) = \emptyset$ and, for each profile $P' \neq P$ such that $P' \lhd P_1$, it is the case that $P' \lhd P$.*

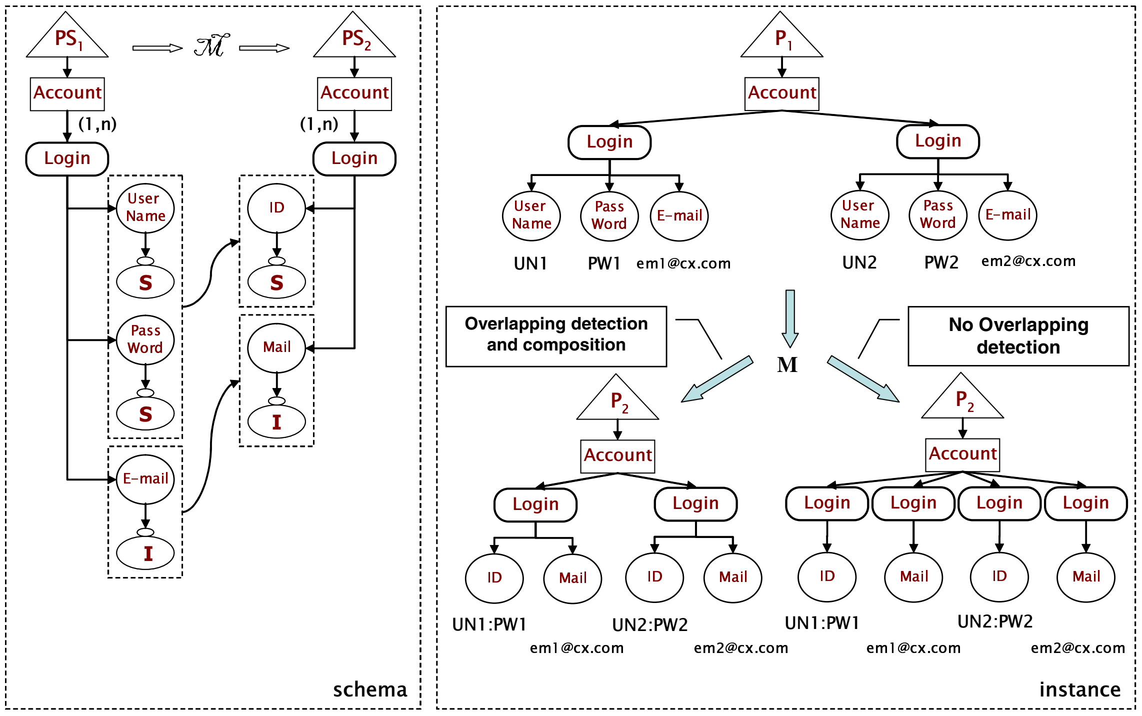


**Figure 6. An example of overlapping and composition**

**Definition 12 (Difference of Profile Models)** *Given two Profile Models $PM_1 = \langle L_1, \omega_1 \rangle$ and $PM_2 = \langle L_2, \omega_2 \rangle$, the difference of $PM_1$ and $PM_2$, denoted by $PM_1 - PM_2$, is a Profile Model $PM = \langle L, \omega \rangle$ where if $L_1 - L_2 \neq \emptyset$ then $L = \{\triangle, \Box, \bigcirc\} \cup L_1 - L_2$, $L = L_1$ otherwise, and for each $(a,b) \in L$, $\omega(a,b) = \omega_1(a,b)$.*

## 3.4 Distance ($\delta$)

To compare the similarity between profiles, we propose the following metric distance.

**Definition 13 (Distance between profiles)** *The distance ($\delta$) between two profiles $P_1$ and $P_2$, denoted by $\delta(P_1, P_2)$, is defined as follows: $\delta(P_1, P_2) = 1 - \frac{|P_1 \sqcap P_2|}{max\{|P_1|, |P_2|\}}$, where $|P|$ denotes the number of edges of a profile $P$.*

It is easy to show that $\delta(P_1, P_2)$ is a metric. If the structure similarity of $P_1$ and $P_2$ is large, the distance between them is small, and vice versa. For instance, in Figure 2 the distance between $P_1$ and $P_2$ is $0.5\overline{3}$.

The comparison between two Profile Models can be measured by the following idea of distance.

**Definition 14 (Distance between Profile Models)** *Given two Profile Models $PM_1 = \langle L_1, \omega_1 \rangle$ and $PM_2 = \langle L_2, \omega_2 \rangle$, the distance of $PM_1$ from $PM_2$, $\delta(PM_1, PM_2)$, is given by $|L_1 - L_2| + \Delta_\omega$, where $\Delta_\omega$ is the number of couples $(a,b) \in L = L_1 \cap L_2$ such that $\omega_1(a,b) \neq \omega_2(a,b)$.*

The distance $\delta(PM_1, PM_2)$ measures how much the Profile Model $PM_1$ can't interpret the Profile Model $PM_2$. In other words, this distance evaluates how many primitives and combinations of them in $PM_2$ are not comprehensible by $PM_1$. For instance, in the Figure 4 the distance of $PM_2$ from $PM_1$, $\delta(PM_2, PM_1)$, is 1. $PM_2$ can't interpret the composition of composite attributes in other ones.

## 3.5 Overlapping ($\rightleftarrows$)

An important task is the detection of *overlapping elements* in a Profile Mapping, which would generate redundancies and inconsistencies at instance level.

**Definition 15 (Overlapping)** *Given two profiles $PS_1$ and $PS_2$, a mapping* **M** *between $PS_1$ and $PS_2$, and two pairs of* **M**

- $m_1 = \langle [(PS_1^{[s_1]}, PS_2^{[t_1]}), (PS_1^{[s_2]}, PS_2^{[t_1]}), \ldots], \mathcal{F}_1 \rangle$,
- $m_2 = \langle [(PS_1^{[s'_1]}, PS_2^{[t_2]}), (PS_1^{[s'_2]}, PS_2^{[t_2]}), \ldots], \mathcal{F}_2 \rangle$,

*there is an overlapping in* **M** *between $m_1$ and $m_2$, denoted by $m_1 \rightleftarrows m_2$, if $PS_2^{[t_1]} \sqcap PS_2^{[t_2]} \neq \emptyset$ and $\exists PS_1^{[s_i]}, PS_1^{[s'_i]}$ such that $PS_1^{[s_i]} \sqcap PS_1^{[s'_i]} \neq \emptyset$.*

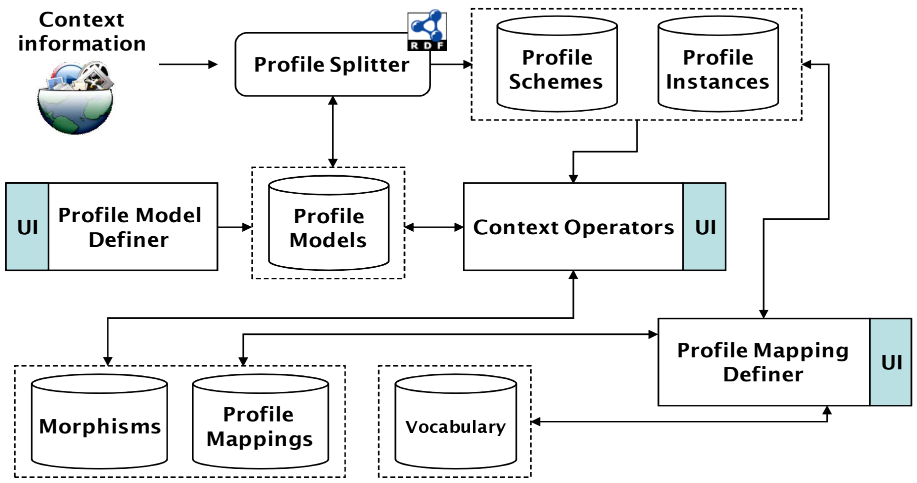


**Figure 7. Tool for Context Management**

## 3.6 Composition ($\uplus$)

We can compose pairs of a Profile Mapping. The composition is defined as follows

**Definition 16 (Composition)** *Given a mapping* $\mathbf{M}$ *and two pairs of* $\mathbf{M}$

- $m_1 = \langle[(PS_1^{[s_1]}, PS_2^{[t_1]}), (PS_1^{[s_2]}, PS_2^{[t_1]}), \ldots], \mathcal{F}_1\rangle$,
- $m_2 = \langle[(PS_1^{[s'_1]}, PS_2^{[t_2]}), (PS_1^{[s'_2]}, PS_2^{[t_2]}), \ldots], \mathcal{F}_2\rangle$

*the composition* $m$ *of* $m_1$ *and* $m_2$*, denoted by* $m_1 \uplus m_2$*, is the pair* $\langle[(PS_1^{[s_1]}, PS_2^{[t_1]}), (PS_1^{[s_2]}, PS_2^{[t_1]}), \ldots, (PS_1^{[s'_1]}, PS_2^{[t_2]}), (PS_1^{[s'_2]}, PS_2^{[t_2]}), \ldots], \mathcal{F}_1 \odot \mathcal{F}_2\rangle$*, where* $\mathcal{F}_1 \odot \mathcal{F}_2$ *means that* $P_2^{[t_1]} = \mathcal{F}_1(PS_1^{[s_1]}, PS_1^{[s_2]}, \ldots, \mathcal{F}_2(PS_1^{[s'_1]}, PS_1^{[s'_2]}, \ldots))$.

Intuitively we compose the single instances that each pair should produce. In particular cases, the composition is driven by the detection of overlapping elements in a mapping. Let's consider Figure 6; we want to map the concatenation of `UserName` and `PassWord` with `ID` and to copy the `E-mail` into `Mail` from the profile schema $PS_1$ to $PS_2$. From the morphism $\mathcal{M} = \{(UserName, ID), (PassWord, ID), (E-mail, Mail)\}$, let's consider a mapping $\mathbf{M}$ as $\{m_1 = ([(PS_1^{[UserName]}, PS_2^{[ID]}), (PS_1^{[PassWord]}, PS_2^{[ID]})]$, **concatenate**$), m2 = ([PS_1^{[E-mail]}, PS_2^{[Mail]}]$, **copy**$)\}$. In the mapping there is an overlapping between $m_1$ and $m_2$. Figure 6 illustrates the resulting profile instance $P_2$ both with overlapping detection and composition and without overlapping detection. This example demonstrates the inefficiency of the mapping if the overlapping is not detected.

# 4 Implementation

We have designed a tool for Context Management, by extending the one presented in [5]. It supports the management of multiple context models and mappings. Given its flexibility, extensibility and expressivity power, we implement instances and schemes in RDF and RDF(S) respectively. The architecture of the tool is shown in Figure 7. The main operations offered by the tool are the following:

- To split an incoming context into autonomous profiles. This task is supported by the *Profile Splitter* that captures a context and produces different profiles respect to the different coordinates, fixed in advance (i.e device capabilities, user preferences and network characteristics). This module implements schemes and instances in RDF(S) and RDF, making use of the Jena framework (`http://jena.sourceforge.net/`).
- The definition of Profile Models, supported by the *Profile Model Definer*. The designer chooses the appropriate set of constructs for a context model and sets an appropriate weight function $\omega$.
- The definition of Profile Mappings, supported by the *Profile Mapping Definer*. Respect to the incoming Profile schemes ($PS_i$) and a *Vocabulary* containing the terms for target Profile schemes ($PS_j$) used by the adaptive application, the designer defines the Morphisms between $PS_i$ and $PS_j$ and the relative Profile Mappings.
- The application of different operators, available from our algebra, to profile schemes and instances. This task is supported by the *Context Operators* module that

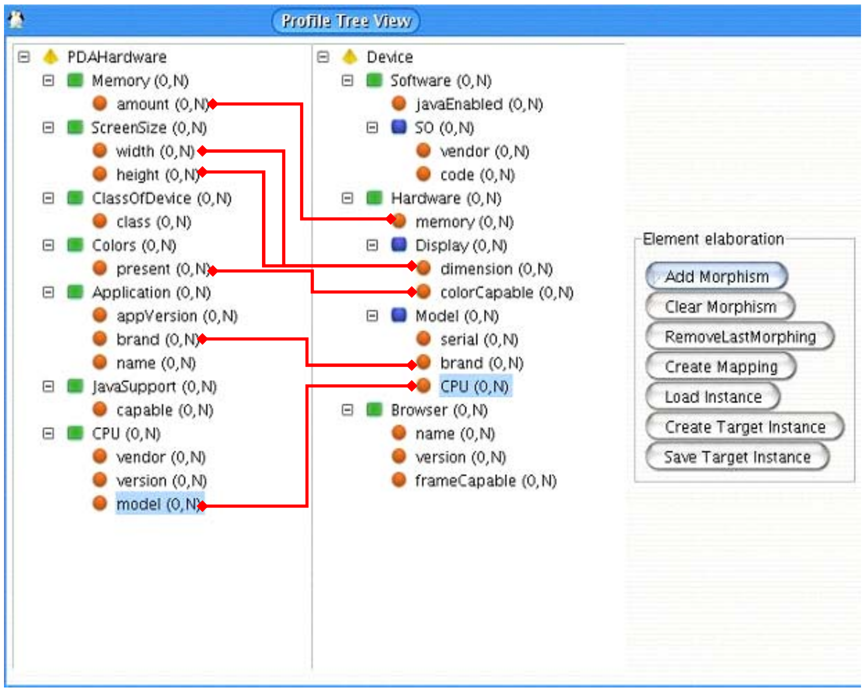


**Figure 8. A snapshot of the application**

takes as input the profile schemes and instances resulting by the Splitter and applies the chosen operator, involving Profile Models or Profile Mappings in case.

All modules of this tool present a *User Interface* to support the designer to perform models and mappings and apply the available operators for context management. For instance, Figure 8 shows a snapshot of the tool, where the user is defining a Profile Mapping between two Profile Schemes.

## 5 Application Scenarios

Usually, context-aware applications are characterized by heterogeneous structural models that are analyzed and matched either manually or semi-automatically at design time. In such applications context management is an important prerequisite. Let's consider the common scenario depicted in Figure 9. Typically, different sources (heterogeneous in case [19]) send several context information to be captured and interpreted. This first step is called *Context Management*. The resulting set of autonomous profiles is taken as input by the *Adaptation* step that generates the adaptation that best fits the context requirements. Finally in the *Content Delivery* step the generated adaptation is applied to the content to produce the final response. In this scenario the Context Management is a crucial step. In this section we present some well-known applications where our framework can provide an important add-on for Context Management. There are *context engineering* and *context information integration*, including schema and data integration, and an emerged new application, *mobile devices communication*.

### 5.1 Context engineering

In this scenario Web applications are confronted with heterogeneous contexts. This activity requires support of context management because context engineering has to deal with multiple, distributed and evolving contexts. In particular we refer to context translation, context data exchange and context evolution and versioning.

**Translation of Contexts.** Given a *source* profile $PI_s$ of a profile schema $PS_s$, described according to a Profile Model $PM_1$, we need to generate a *target* profile $PI_t$ of a target profile schema $PS_t$ described according to a Profile Model $PM_2$ , containing the same information as $PI_s$. Now we can express a translation using our framework. Let's consider an operation ($\Gamma$) that takes as input a profile (schema or instance) $P_1$ according to a Profile Model $PM_1$ and returns a profile (schema or instance) $P_2$ according to a Profile Model $PM_2$ containing the same information of $P_1$, denoting by $\Gamma_{PM_1 \rightarrow PM_2}(P_1) = P_2$. $\Gamma$ represents an elementary translation of a profile (schema or instance) from a representation into another. For instance the Figure 10 represents a translation of a profile schema from a Profile

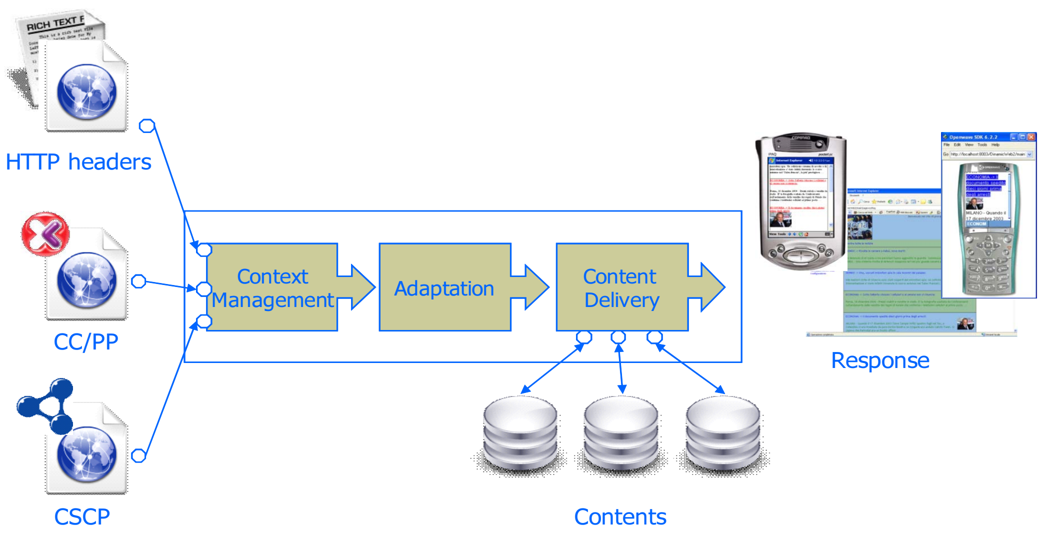


**Figure 9. A scenario of reference**

Model that articulates the schema on n levels, making use of composite attributes, into a Profile Model that articulates the schema on 2 levels, making only use of simple attributes (and external references) and no composite ones. Let's consider a prefixed set of $\Gamma$ operations representing atomic translations, for instance of some primitives. Intuitively given an elementary operation $\Gamma_{PM_i \rightarrow PM_j}$, if we apply it to a profile (schema or instance) $P$ described by a Profile Model $PM$ such that $PM_i \lhd PM$ then the operation will translate the part $P'$ of $P$ described by $PM_i$ into $P''$ described by $PM_j$. The resulting profile will be described by a Profile Model such as $(PM - PM_i) \sqcup PM_j$. So we can denote also the following $\Gamma_{PM_1 \rightarrow PM_2}(PM) = (PM - PM_1) \sqcup PM_2$. Therefore, we perform a translation from $PM_1$ into $PM_2$ by means a sequence of operations $\digamma = \{\Gamma_{PM_i \rightarrow PM_j}, \Gamma_{PM_n \rightarrow PM_m}, \ldots\}$ such that the distance between $PM_2$ and the resulting Profile Model $\digamma(PM_1)$ is zero.

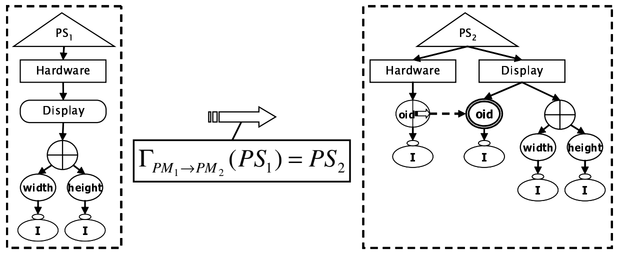


**Figure 10. An example of $\Gamma$ operation**

**Context data Exchange** Given two profile schemes $PS_1$ and $PS_2$, described by the same Profile Model $PM$, we want to generate a profile instance $PI_2$ according to $PS_2$ from $PI_1$ according to $PS_1$ such that $PI_2$ contains the same information of $PI_1$. Let's consider an operation *Map*, denoted by $(\mu)$, that takes $PS_1$ and $PS_2$ as input and returns a Profile Mapping $\mathbf{M}$ between them, such as $\mathbf{M} = \mu(PS_1, PS_2)$. This operation is supported of a prefixed set of combinational functions to combine the morphisms between the input schemes. In some cases, the definition is quite simple. For example, the equality of two objects may be based on equality of their identifiers or names or using technology from a variety of fields (graph isomorphism, natural processing language, domain-specific thesauri, machine learning, data mining, etc...). Also the choice of the expression to combine components (such as a concatenation) can be automatically computed. In other cases, it is quite complex and perhaps subjective. In practice, the Map could be not an algorithm that returns a mapping but rather is a design environment to heal a human designer to develop a mapping. For further details, a survey of approaches to automatic schema matching is [18]. Therefore, we make use of $\mu$ to produce the mapping $\mathbf{M} = \mu(PS_1, PS_2)$. Then we detect and compose the overlapping elements of $\mathbf{M}$ by using respectively the primitives $\rightleftarrows$ and $\uplus$. Finally we iterate over the components of $PS_1$ involved in $\mathbf{M}$ and produce the resulting $PI_2$ as join between the instances produced by each couple $m$ of $\mathbf{M}$. We denote $PI_2 = \tau(\mathbf{M}, PI_1, PS_2)$. We extensively studied the translation and context data exchange problems in [5]. However here we have revisited them by using the algebra discussed in the previous section.

**Context evolution and versioning** It is natural that application requirements and the way in which context information are engineered dynamically evolve over time. Multiple

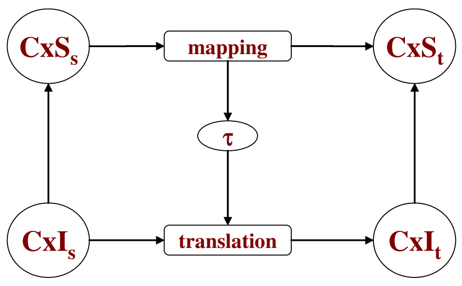


**Figure 11. Context evolution and versioning**

versions of the same context often exist. Some applications keep their context up to date, while others may continue to use old context versions and update them on their own. These situations arise because Web engineers and developers usually do not have a global view of how and where the contexts have changed. The context management is to help here. It mainly focuses on discovering differences (i.e. what dimensions have been added, deleted or renamed) between two context versions. As shown in Figure 11, in this scenario it is useful (i) to produce a mapping $\mathbf{M}$ between the old version ($CxS_s$) and the new version ($CxS_t$) of the context, (ii) generate a transformation $\tau$ by using $\mathbf{M}$, and (iii) translate the underlying context data instances ($CxI_s$ in $CxI_t$). This problem is comparable with the alignment of different versions of the same ontology [17].

## 5.2 Context Information Integration

Information integration is one of the oldest classes of applications. Under the information integration we gather problems as schema and data integration [10]. We face this problem in our scenario, where context management is a plausible solution. The problem is to create a profile instance $PI_3$, with a schema $PS_3$, that represents all the information expressed in two given profile instances $PI_1$ and $PI_2$, having respectively schemes $PS_1$ and $PS_2$. As shown in Figure 12, there are three main activities: (i) create the Profile Mappings $\mathbf{M}_1$ between $PS_1$ and $PS_3$ and $\mathbf{M}_2$ between $PS_2$ and $PS_3$ and generate the instances $PI'_1$ and $PI'_2$, resulting by applying the mappings $\mathbf{M}_1$ and $\mathbf{M}_2$ respectively to $PI_1$ and $PI_2$, (ii) resolving conflicts (i.e., where the same information was represented differently in $PI'_1$ and $PI'_2$) and (iii) generate $PI_3$ from $PI'_1$ and $PI'_2$. We can express these activities using our framework as follows. We return $\mathbf{M}_1 = \mu(PS_1, PS_3)$ and $\mathbf{M}_2 = \mu(PS_2, PS_3)$; then we generate $PI'_1 = \tau(\mathbf{M}_1, PI_1, PS_3)$ and $PI'_2 = \tau(\mathbf{M}_2, PI_2, PS_3)$. Since $PS_3$ has a vocabulary $\mathbf{V} = \{\mathbf{D}, \mathbf{A}\}$, a conflict arises if the same simple attribute $A_i \in \mathbf{A}$ has different values in $PI'_1$ and $PI'_2$; in this case we can assume each profile having a priority and for example in this case we fix $PI'_1$ with the major

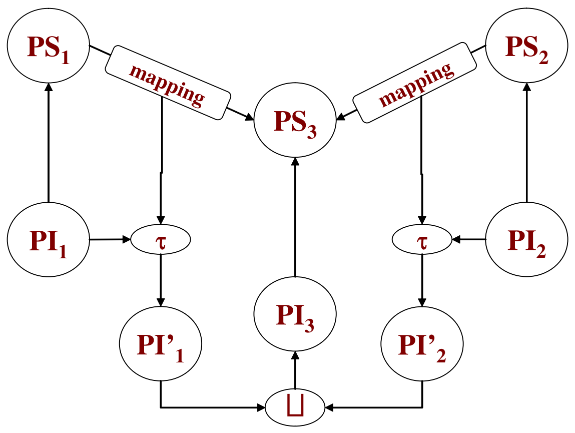


**Figure 12. Context integration**

priority. So for each simple attribute $A_i \in \mathbf{A}$ such that $PI'_1(A_i) \neq PI'_2(A_i)$, with $PI'_1(A_i)$ and $PI'_2(A_i)$ not null, then $PI'_2(A_i) = PI'_1(A_i)$. Finally we generate $PI_3$ as join between $PI'_1$ and $PI'_2$ such as $PI_3 = PI'_1 \sqcup PI'_2$.

## 5.3 Mobile devices communication

Applications running on mobile devices take advantage of a context management framework for providing services to users. Let's regard to ambient computing. The applications must always keep track of changes (i.e. user locations). Characterizing the context in ambient computing goes through finding information about the current situation in the environment by using various devices available in that environment (e.g. sensors). Similar to Web services descriptions, context information will provide descriptions of the devices, such as a temperature service, and the way to interact with them. Let's consider temporal management of RFID data [23]. RFID data are dependent by time and dynamically change. RFID data management systems have to achieve large scale temporal data created by RFID applications. These applications need to have an expressive data model to structure incoming context information from different sources, an automatic data acquisition (i.e. filtering) and transformation, and adaptable and portable logic. Moreover, given the large scale of data, RFID applications has to reduce the cost of managing RFID data, for instance using clustering techniques. To this aim our framework introduces a suitable cluster based solution. Given a set of contexts $\mathbf{Cx} = \{Cx_1, \ldots, Cx_n\}$, we want to classify them in a set of clusters $\mathbf{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ that share common characteristics. By using our framework, we define a *Profile Cluster* as a tree $\{N_\mathcal{C}, E_\mathcal{C}, r_\mathcal{C}\}$ where (i) $N_\mathcal{C}$ is a set of nodes representing profiles, (ii) $E_\mathcal{C}$ is a set of edges $(n_i, n_j)$ such that $n_i \lhd n_j$, and (iii) $r_\mathcal{C}$ is the root. In a cluster a profile $P_1$ is parent of a profile $P_2$ if $P_1 \lhd P_2$, therefore the root represents the most general profile in the cluster. Moreover let's consider a set of profiles $\mathbf{P} = \{P_1, \ldots, P_n\}$, and an operation *P-CLS* ($\eth$) that classifies $\mathbf{P}$ in a set of Profile Clusters $\eth(\mathbf{P}) = \mathbf{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$. Principally *P-CLS* makes use of the primitives $\lhd$ and $\delta$. So from $\mathbf{Cx}$ we produce several sets of profiles as $\mathbf{P}_i = \{P_{i,1}, \ldots, P_{i,n}\}$ grouping respect the same type of coordinate $i$ (device capabilities, user preferences, network characteristics, ...). Therefore, we generate for each $\mathbf{P}_i$ a set of clusters by applying the $\eth$ operator (i.e $\mathcal{C}_i$=$\eth(\mathbf{P}_i)$).

Referring to Adaptive Web applications, a relevant problem is the definition of effective methodologies for choosing efficiently the most suitable adaptation for a given context. To this aim, we need to classify contexts on the basis of their characteristics. At a logical level, each class corresponds to contexts that require similar adaptations. Also in this case our framework is a valid support. We extensively studied this problem in [7].

## 6 Related work

Modeling context information based on formal descriptions is a core aspect of adaptive Web applications [6]. Being context information the condition to activate an adaptation process, in this paper we have focused on management of context data in Adaptive Web-based Information Systems. A number of approaches and methodologies for modeling context information have been proposed in the last years, testifying the relevance of the task. Theodorakis et al. [20] present a model for representing contexts in information bases along with a set of operations for manipulating contexts. These operations support creating, updating,combining and comparing contexts. Contexts are a special kind of objects that represent real world divisions or environments. In [8] the authors consider a context as a set of user profiles, and organize them using a concept of dimension (similar as in our approach). The dimension is used to organize the data of a profile in totally ordered sets. Henricksen et al. [11] focus on the problems associated with previous context models, including their lack of formality and generality. They individuate principal characteristics of context information (temporal aspects, incorrectness, inconsistency, incompleteness, ...) and propose an object-oriented approach (entity-object and associations) to manage it. In [21] the authors distinguish different context scenarios (oriented to customer's usage), a context-free expression with a set of parameters and operations of copy and filtering. Principally the approach is oriented to increase the customers' usage comfort. In [2] the authors propose a rule-based mechanism, supported by first order logic, that allows the modification of the context (modeled as a set of attribute/value pairs) to solve conflicts in the adaptation requirements. Finally Cappiello et al. [3] provide a framework to define and manage the context. The approach, using ontology and OWL, illustrates a top ontology as container of the general concepts defined independently of a domain of application and that could be used in different application domains. Principally context information are in user preferences, geographical position and channel capabilities. All of these approaches concern with the development of a context management infrastructure, considered a relevant issue. However, we believe that in general the majority of the above mentioned approaches present a common lack of formality and generality. They provide specific solutions that are suited for a particular class of predefined context information (i.e. commonly device characteristics and user preferences). The management of context data often is reduced to operations of copy and filtering. That is why we have addressed in this paper the problem from a generic perspective aimed at facilitating a uniform and generic solution. Our approach gets inspiration from the data management framework of Bernstein [1]. The motivations and goals are similar but the scenario of reference is quite different. Bernstein proposes a framework to manage all existing models of data in several research areas. Our approach focuses on the management of context data in adaptive Web applications, that is a scenario where the framework of Bernstein would be an instrument too complex and overmuch, with consequent loss of efficiency and effectiveness.

## 7 Conclusions and Future Work

In this paper, we described a new approach to manipulating context information. We proposed a data model that consists of a general abstraction of existing formats that allows to define different context models and mappings. The data model is enriched by an algebra of operators to embed the main functionalities of context management. We believe that this framework is an important support in different application scenarios such as contexts translation, contexts integration or contexts classification by means of the operators embedded. In addition to implementation, there are many other areas where work is needed to fully realize the potential of this approach. Some of the more pressing ones are (i) extending the set of basic primitives to capture a major number of formats, (ii) more detailed semantics and consequently extension of the algebra of operators, and (iii) trying to apply context management to especially challenging problems, to identify limits to the approach and opportunities to extend it.

## References

[1] P. A. Bernstein. Applying model management to classical meta data problems. In *Proc. of the 1th Bien-*

*nial Conference on Innovative Data Systems Research (CIDR'03), CA, USA*, 2003.

[2] C. Bettini and D. Riboni. Profile Aggregation and Policy Evaluation for Adaptive Internet Services. In *Proc. of the 1th Int. Conference on Mobile and Ubiquitous Systems (MobiQuitous'04), Cambridge, MA, USA*, 2004.

[3] C. Cappiello, M. Comuzzi, E. Mussi and B. Pernici. Context Management for Adaptive Information Systems. In *Electr. N. Theor. Comput. Sci. 146(1): 69-84*, 2006.

[4] S. Ceri, F. Daniel, V. Demaldé, and F. M. Facca (2005). An approach to user-behavior-aware Web applications. In *proc. of 5th International Conference on Web Engineering (ICWE'05).*

[5] R. De Virgilio and R. Torlone. Modeling Heterogeneous Context Information in Adaptive Web Based Applications. In *Proc. of 6th Int. Conference on Web Engineering (ICWE'06), Palo Alto, California, USA*, 2006.

[6] R. De Virgilio, R. Torlone and G. J. Houben. Rule based Adaptation of Web Information Systems. In *World Wide Web Journal 10(4): 443-470*, 2007.

[7] R. De Virgilio, R. Torlone, D. Valeriano, D. Di Federico. A Cluster-based Approach to Web Adaptation in Context-Aware Applications. In *International Journal of Web Engineering 6(4): 360-388*, 2007.

[8] T. Feyer, O. Kao, K. D. Schewe and B. Thalheim. Design of Data-Intensive Web-Based Information Services. In *Proc. of the 1th Int. Conference on Web Information Systems Engineering (WISE'00), Hong Kong*, 2000.

[9] F. Frasincar, G. J. Houben, and R. Vdovjak (2002). Specification Framework for Engineering Adaptive Web Applications. In *Proc. of 11th Int. Conference on World Wide Web (WWW'02).*

[10] A. Halevy, N. Ashish, D. Bitton, M. Carey, D. Draper, J. Pollock, A. Rosenthal and V. Sikka. Enterprise information integration: successes. challenges and controversies. In *Proc. of the 24th International Conference on Management of Data (SIGMOD'05), Baltimore, USA*, 2005.

[11] K. Henricksen, J. Indulska and A. Rakotonirainy. Modeling Context Information in Pervasive Computing Systems. In *Proc. of the 1th Int. Conference on Pervasive Computing (PERVASIVE'02), Zurich*, 2002.

[12] G. Kappel, W. Retschitzegger and W. Schwinger (2001). Modeling Ubiquitous Web Applications: The WUML approach. In *Int. Workshop on Data Semantics in Web Information Systems (DASWIS01).*

[13] R. Kaschek, K. D. Schewe and B. Thalheim. Integrating Context in Modelling for Web Information Systems. In *WES Vol. 3095, pp. 77-88*, 2003.

[14] N. Koch. An object-oriented hypermedia reference model formally specified in UML. In *Information modeling for internet applications, pages 59-78, IGI Publishing*, 2003.

[15] N. Koch and M. Wirsing. The Munich Reference Model for Adaptive Hypermedia Applications Source. In *Proc. of the 2th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'04), Malaga, Spain*, 2002.

[16] R. Pitrik. An Integrated View on the Viewing Abstraction: Contexts and Perspectives in Software Develepment, AI, and Databases. In *J. of Systems Integration, 5(1):23-60*, 1995.

[17] N. Noy and M. Musen. Ontology versioning in an ontology management framework. In *IEEE Intelligent Systems, 19(4): 6-13*, 2004.

[18] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB J. 10(4):334–350*, 2001.

[19] T. Strang and C. Linnhoff-popien. A context modeling survey. In *Proc. of the 1th Int. Workshop on Advanced Context Modelling, Reasoning and Management, England*, 2004.

[20] M. Theodorakis, A. Analyti, P. Constantopoulos and N. Spyratos. Context in Information Bases. In *Proc. of the 3th Int. Conference on Cooperative Information Systems (CoopIS'98), New York, USA*, 1998.

[21] A. B. Zdanowicz, R. Kaschek, K. D. Schewe and B. Thalheim. Context-aware Web Information Systems. In *Proc. of the 1th Asian-Pacific conference on Conceptual modelling, Dunedin, New Zealand*, 2004.

[22] W3C. Device independence principles. *http://www.w3.org/TR/di-princ/* (2003)

[23] F. Wang and P. Liu. Temporal management of RFID Data. In *Proc. of the 31th International Conference on Very Large Database (VLDB'05), Trondheim, Norway*, 2005.