Towards Highly Adaptive Services for Mobile Computing*

Alessandra Agostini[1], Claudio Bettini[1], Nicolò Cesa-Bianchi[2], Dario Maggiorini[1], Daniele Riboni[1], Michele Ruberl[3], Cristiano Sala[3], and Davide Vitali[1]
[1]*DICo, University of Milan, Italy*
[2]*DSI, University of Milan, Italy*
[3]*B Human Web Factory, Milan, Italy*

Abstract: The heterogeneity of device capabilities, network conditions and user contexts that is associated with mobile computing has emphasized the need for more advanced forms of adaptation of Internet services. This paper presents a framework that addresses this issue by managing distributed profile information and adaptation policies, solving possible conflicts by means of an inference engine and prioritization techniques. The profile information considered in the framework is very broad, including user preferences, device and network capabilities, and user location and context. The framework has been validated by experiments on the efficiency of the proposed conflict resolution mechanism, and by the implementation of the main components of the architecture. The paper also illustrates a specific testbed application in the context of *proximity marketing*.

1. INTRODUCTION

The continued growth in the amount of content and the number of information services available on-line has made effective personalized content delivery a hot research topic. Considering the increasing capabilities of mobile infrastructure and device hardware, mobile devices will probably

* This work has been partially supported by Italian MIUR (FIRB "Web-Minds" project N.RBNE01WEJT_005)

become the most common clients for on-line information systems. User-orientation and personalization in mobile information systems has been recognized as a major research challenge [20]. Indeed, due to the heterogeneity of these devices, new aspects should be taken into account for effective adaptation, among which device capabilities and status (e.g., screen resolution, battery level, network available bandwidth). Mobility also leads to a much wider variety of user contexts including but not limited to spatio-temporal data (e.g., location, speed, direction), and social setting situations (e.g., business meeting, home, shopping). If known by the service provider, this data can be extremely valuable for adapting content delivery. In our framework, we extend the notion of *profile* data to include all the information that can contribute to achieve an effective adaptation.

Current approaches to mobile oriented adaptation are still quite limited. In most cases, they are technically based on transcoding, and conceptually based on the assumption that device capabilities can be deduced by the HTTP request headers. Moreover, most approaches assume that user profile data is available server-side. We believe that, despite a lot of information can be gathered server-side, either explicitly given by the user or deduced by historical data on interactions with the same user, this information cannot include many of the relevant aspects we have mentioned above. In our view, profile data is naturally distributed and should not be forced to be stored and managed only server-side. In our framework, each source of profile data (e.g., user, network operator, service provider) has an associated trusted profile manager, which is typically running on a wired infrastructure, and that can communicate with other profile managers. Hence, profile data can be stored and managed locally and selectively made available to service providers. It is the responsibility of service providers to access the portion of profile data needed for the services they are delivering. User profile data can be made available to a new service provider by simply allowing access to the user profile manager. This model, by storing and managing profile data at the source, also avoids consistency problems upon updates of profile attributes (consider e.g., spatio-temporal or social setting information). Upon each user request the service provider profile manager is responsible for querying the necessary profile managers and aggregating profile data. This task includes solving conflicts due to different values provided by different entities for the same attribute. The introduction of profile managers also implies the adoption of a standard formalism for the representation of profile data, enabling the interoperability among the various entities.

In order to achieve enhanced personalization, our framework also allows users and service providers to augment the profile attributes with policies; that is, rules that set or change certain profile attributes based on the current values of other profile attributes. Clearly, the introduction of policies makes

it possible to have, once more, conflicting attribute values, even considering only policies from the same entity (service provider or user). For this reason, the policy evaluation mechanism defined by the framework includes a quite involved conflict resolution technique.

The main contribution of this paper is the presentation of the architecture of our framework, first from a logical point of view, and then from an implementation point of view, in terms of a software architecture. Finally, in this paper we present a test case with an *adaptive proximity marketing* application used to validate our prototype on a real domain. A theoretical and experimental study on the soundness and efficiency of our conflict resolution mechanism has also been performed that validates our approach in terms of performance and scalability, but details are beyond the scope of this paper. For lack of space, we cannot include in this paper the discussion of two relevant issues: 'intra-session' adaptation, and privacy. We just mention here that we devised a distributed trigger mechanism for the former, and adopt access control techniques [3, 16] for the latter.

The rest of the paper is structured as follows: In the following section we give an overview of the framework logical architecture illustrating the formalism used to represent profiles and policies, the role of the main modules and the techniques used for conflict resolution. In Section 3 we illustrate how each component of the logical architecture has been implemented in the corresponding software architecture. Section 4 presents a testbed application used to demonstrate the system prototype. Section 5 discusses related work and Section 6 presents future research directions.

2. ARCHITECTURE

In this section we describe the logical architecture of our framework, starting with a list of requirements that have driven the design process. We then present its main components as well as the issues related to profile and policy representation and management.

2.1 Requirements

Based on an analysis of a large spectrum of Internet services that would benefit from adaptation, of the data required for implementing highly adaptive services, of the infrastructure that is available now and will available in the near future, as well as of the issues of data privacy and accessibility, we have identified the following set of requirements. (*i*) A representation formalism is needed for the specification of a very broad set of profile data, which integrates device capabilities with spatio-temporal

context, device and network status, as well as user preferences and semantically rich context; (*ii*) A representation formalism is needed for the specification of policies, which can dynamically determine the value of some profile data or presentation directives based on other values, possibly provided by different entities; (*iii*) Vocabularies and/or ontologies should be defined in order for different entities to share terms for the specification of profile attributes; (*iv*) The architecture should support the distributed storage and management of profiles and policies, with information stored and managed close to its source; (*v*) The architecture should provide a mechanism to aggregate profile data and policies from different sources, supporting a flexible and fine-grained conflict resolution mechanism; (*vi*) The architecture should rely on an advanced system for privacy protection which allows the user to precisely control the partial sharing of his profile data; (*vii*) The architecture should provide a configurable mechanism for 'intra-session' adaptation based on real-time update of certain profile data (e.g., location).

Clearly, efficiency should be taken into account when evaluating different solutions, even if efficiency requirements may vary based on the considered service.

2.2 Architecture Overview

The specification and implementation of a full-fledged architecture satisfying all the requirements illustrated above is a long-term goal. The contribution illustrated in this paper is a first step in this direction. We present an architecture where three main entities are involved in the task of building an aggregated profile, namely: the user with his devices (called *user* in the rest of the paper), the network operator with its infrastructure (called *operator*), and the *service provider* with its own infrastructure. A Profile Manager devoted to manage profile data and policies is associated with each entity and will be called *UPM*, *OPM*, and *SPPM*, respectively. In particular, *(i)* The `UPM` stores information related to the user and his devices. These data include, among other things, personal information, interests, context information, and device capabilities. The `UPM` also manages policies defined by the user, which describe the content and the presentation he wants to receive under particular conditions; *(ii)* The `OPM` is responsible for managing attributes describing the current network context (e.g., location, connection profile, and network status); *(iii)* The `SPPM` is responsible for managing service provider proprietary data including information about users derived from previous service experiences. Clearly, the architecture, including conflict resolution mechanisms, has been designed to handle an arbitrary number of entities (e.g., profile managers owning context services).

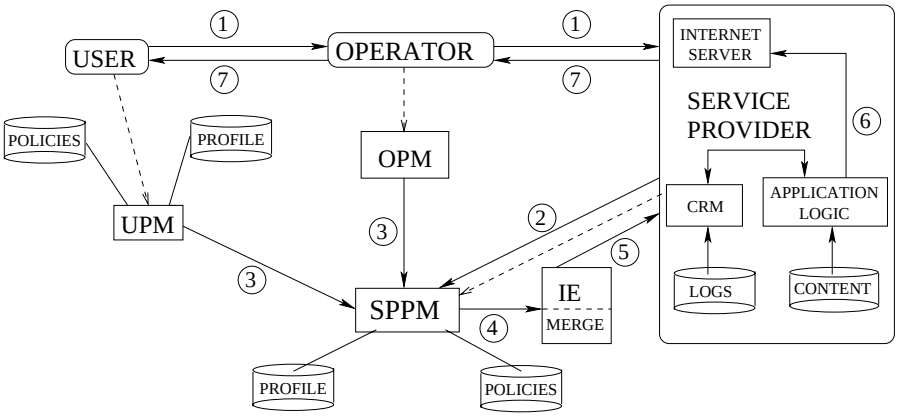

Figure 1. Architecture overview and data flow upon a user request

Figure 1 provides an overview of the proposed architecture. We illustrate the system behavior by describing the main steps involved in a service request: *(1)* A user issues a request to a service provider through his device and the connectivity offered by a network operator; *(2)* The service provider queries its Profile Manager (`SPPM`) to retrieve the profile information needed to perform adaptation; *(3)* The `SPPM` queries the `UPM` and the `OPM` to retrieve profile data and user's policies; *(4)* The `SPPM` then forwards collected and local profile data and policies to the Inference Engine (`IE`); *(5)* The `IE` first merges profile data; then, it evaluates service provider and user policies against the merged profile, resolving possible conflicts. The resulting profile attributes are then returned to the Service Provider; *(6)* These attribute values are used by the application logic to properly select content and customize its presentation; *(7)* Finally, the formatted content is sent to the user.

2.3 Profile Management and Aggregation

In the following we explain the mechanism of profile management, and address the issue of how to aggregate possibly conflicting data in a single profile.

2.3.1 Profile representation

In order to aggregate profile information, data retrieved from the different profile managers must be represented using a well defined schema,

providing a mean to understand the semantics of the data. For this reason, we chose to represent profile data using the Composite Capabilities/Preference Profiles (CC/PP) structure and vocabularies [19]. CC/PP uses the Resource Description Framework (RDF) to create profiles describing device capabilities and user preferences. In CC/PP, profiles are described using a 2-level hierarchy; attribute values can be either *simple* (string, integer or rational number) or *complex* (set or sequence of values, represented as `rdf:Bag` and `rdf:Seq` respectively). CC/PP attributes are declared in RDFS vocabularies. In addition to well known CC/PP-compliant vocabularies for device capabilities like UAProf [24] and its extensions, our framework assumes the existence of vocabularies describing information like user's interests, content and presentation preferences, and user's context in general. Clearly, there are several issues regarding the general acceptance of a vocabulary, the privateness of certain server-side attributes, and the uniqueness of attribute names. In this paper, we simply assume there exists a sufficiently rich set of profile attributes that is accessible by all entities in the framework. We also simplify the syntax used to refer to attributes avoiding to go into RDF and *namespace* details.

2.3.2 Profile aggregation and conflict resolution

Once the `SPPM` has obtained profile data from the other profile managers, this information is passed to the `IE` which is in charge of profile integration (Step 4 in Figure 1). Conflicts can arise when different values are given for the same attribute. For example, the `UPM` could assign to the *Coordinates* attribute a certain value *x* (obtained through the GPS of the user's device), while the `OPM` could provide for the same attribute a different value *y*, obtained through triangulation. In our architecture, resolution of this kind of conflicts is performed by the `Merge` submodule of the `IE`. In order to resolve this type of conflict, the Service Provider has to specify resolution rules at the attribute level in the form of priorities among entities. Priorities are defined by *profile resolution directives* which associate to every attribute an ordered list of profile managers, using the *setPriority* statement. This means that, for instance, a service provider willing to obtain the most accurate value for user's location can give preference to the value supplied by the `UPM` while keeping the value provided by the `OPM` just in case the value from the `UPM` is totally missing. Continuing the above example, the directive giving higher priority to the user for the *Coordinates* attribute is:

setPriority Coordinates=(*UPM*,*OPM*)

Profile resolution also depends on the type of attribute. With respect to attributes of type `Bag`, the values to be assigned are the ones retrieved from all entities present in the list. If some duplication occurs, only the first

occurrence of the value is taken into account (i.e., we apply the union operation among sets). Finally, if the type of the attribute is `Seq`, the values to be assigned to the attribute are the ones provided by the entities present in the list, ordered according to the occurrence of the entity in the list. If some duplication occurs, only the first occurrence of the value is taken into account.

2.4 Policies for Supporting Adaptation

As anticipated in the introduction, policies can be declared by both the service provider and the user. In particular, service providers can declare policies in order to dynamically personalize and adapt their services considering explicit profile data. For example, a service provider can choose the appropriate resolution for an image to be sent to the user, depending both on user preferences and on current available bandwidth. Similarly, users can declare policies in order to dynamically change their preferences regarding content and presentation depending on some parameters. For instance, a user may prefer to receive high-resolution media when working on his palm device, while choosing low-resolution media when using a WAP phone. Both service providers and users' policies determine new profile data by analyzing profile attribute values retrieved from the aggregated profile.

2.4.1 Policy Representation

Each policy rule can be interpreted as a set of conditions on profile data that determine a new value for a profile attribute when satisfied. A policy in our language is composed by a set of rules of the form:

If C_1 **And … And** C_n **Then Set** $A_k=V_j$

where A_k is an attribute, V_j is either a value or a variable, and C_i is either a condition like $A_i=V_l$ or *not* A_i with the meaning that no explicit nor derived value for A_i exists. For example, the informal user policy:

"When I am in the main conference room using my palm device, any communication should occur in textual form"

can be rendered by the following policy rule:

"**If** *Location*='MConfRoom' **And** *Device*='PDA' **Then Set** *PreferredMedia*='Text'"

2.4.2 Conflicts and resolution strategies

Since policies can dynamically change the value of an attribute that may have an explicit value in a profile, or that may be changed by some other policies, they introduce nontrivial conflicts. They can be determined by

policies and/or by explicit attribute values given by the same entity or by different entities. We have defined conflict resolution strategies specific for different conflict situations. While a complete description of possible conflicts and of the solutions implemented in our architecture is beyond the scope of this paper (see [4] for further details), here we just mention the basic technique. We implement conflict resolution strategies by transforming the logical program defined by the policy rules. Transformations basically consist in the assignment of a proper *weight* to each rule and in the introduction of negation as failure. In the resulting program, each rule with a generic head predicate *A* and weight *w* is evaluated only after the evaluation of the rule with the same head predicate and weight *w*+1. When a rule with weight *w* fires, rules with the same head predicate having a lower weight are discarded. The weight assignment algorithm ensures that the evaluation of the program satisfies the conflict resolution strategies, and a direct evaluation algorithm can be devised that is linear in the number of rules.

3. SOFTWARE ARCHITECTURE

An illustration of the software modules which have been developed is shown in Figure 2. There are two distinct data flows, which correspond to profile modifications and service requests, identified by Sequence I and II, respectively.

The local proxy (C) is an application running on the user device which adds custom fields to the HTTP request headers, thus enabling the SPPM to locate the user's ID and the URIs of his UPM and OPM. Currently, the local proxy is developed in C# (see Figure 3-A) and can be executed over the .NET (Compact) Framework. The UPM, OPM and SPPM consoles (B, P, Q) are browser-based web applications, which allow to modify profile attributes on the UPM, OPM and SPPM repositories. The Service Provider Application Logic module (E) is the component which delivers the profile- and context-dependent service to the user. The application logic implementation depends on the type of service to be delivered; the implementation of the application logic for the prototype web application we developed is briefly described in Section 4.

Besides managing local profiles and policies, the SPPM retrieves data from the remote profile managers and from its own repositories and feeds them to the Merge (I) and IE (J) modules. The integrated profile is returned via SOAP to the service provider application logic. The Merge module (I) receives from the Business Logic EJB (H) the profile resolution directives and the objects representing the remote profiles. Attribute values are retrieved from profiles using RDQL, a query language for RDF documents

implemented by the Jena Toolkit [17]. The integrated profile is built by applying the service provider profile resolution directives, as explained in Section 2. Finally, the object representing the integrated profile is forwarded to the Inference Engine module (J), together with the set of user and content provider policies, and profile resolution directives.

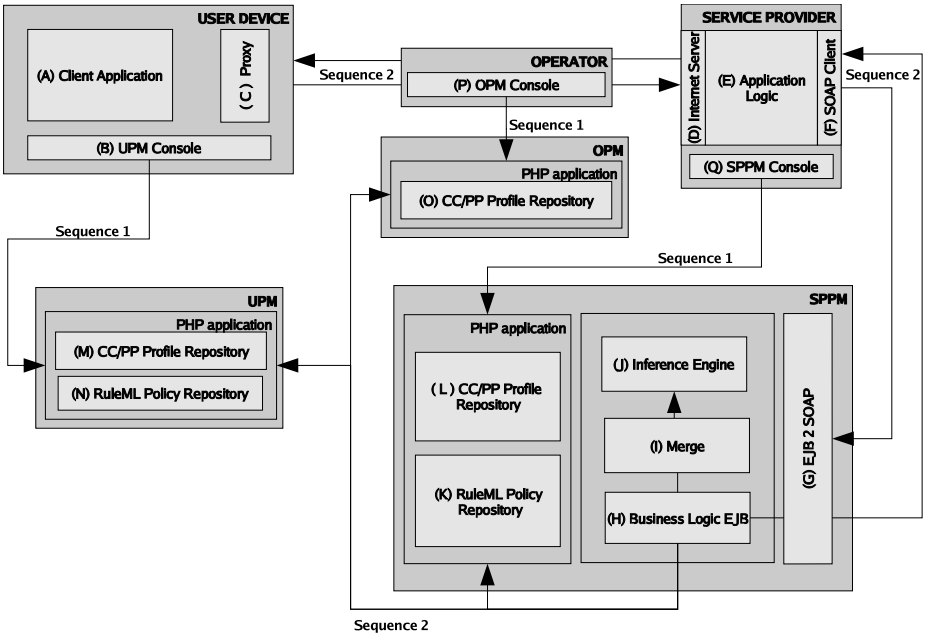

Figure 2. The developed software modules

Before starting the evaluation phase, the IE module modifies the logic program (composed by facts retrieved from the integrated profile, and policies) in order to apply the conflict resolution strategies described in Section 2. User and service provider policies are represented in RuleML [5]. The evaluation of the logic program is performed using Mandarax, an open source Java package for deductive rules. Mandarax is designed as a backward reasoning engine, and supports negation as failure, which is needed in our case to implement the conflict resolution mechanism. The output of the derivation process is a result-set in which every row contains a value of an attribute. These values are used to update the Java object representing the integrated profile, which is returned to the EJB (F).

Our planned technology for the Profile Managers includes the adoption of an RDF server such as Joseki [18]. However, at the time of writing, the profile repositories (L, M, O) are a collection of simple files in CC/PP

format. Policy repositories (K, N) are a collection of RuleML files which describe the user and service provider policies.

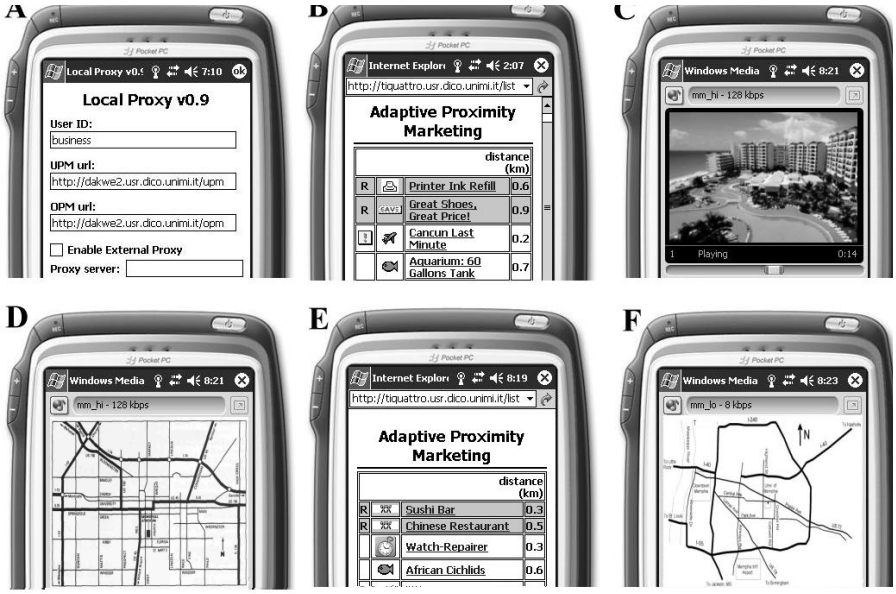

Figure 3. Some screen-shots of the web application prototype

4. AN ADAPTIVE PROXIMITY MARKETING SERVICE

In order to test our software architecture we developed a set of prototype services. In this section, we illustrate a web-based adaptive proximity marketing service. Its main goal is to provide targeted, location-aware *advertisements* about sales on items contained in a user's personal shopping list. For example, if the user's shopping list includes a specific camera model and the user is walking on a street where a shop has that camera on sale, the service will list an appropriate geolocalized ad on the user's device, possibly linked to multimedia content details. While we are not the first to consider such a service, our emphasis is on adaptation based on user and service provider policies. Advertisements are chosen and ranked by considering not only the personal shopping list, but other profile data such as the user's location, interests, and action context. Users can be either paying or non-paying service subscribers. Non-paying subscribers may also receive *unsolicited advertisements* regarding items which are not on their shopping

list. The choice of items for unsolicited advertisements can be driven by standard CRM software as well as from aggregated profile data. The service currently implemented is browser-based, and provided on a per-request basis (i.e., it is a pull service). The service is activated by accessing a specific web page, and the delivery of content is performed by the Cocoon programming framework [10]. Upon each request, the service returns a web page with the list of ads, which is automatically refreshed after a certain period of time. This time is dynamically set server-side based on aggregated profile data, and communicated to the (micro)browser using a `META` element.

Table 1. An excerpt of policies

Policy	**Owner**
(1) If *DeviceType* = 'PDA' Then Set *MediaQuality* = 'High'	user
(2) If *AvailableBandwidth* < 56kbps Then Set *MediaQuality* = 'Low'	service provider
(3) If *UserSpeed* = 'Slow' Then Set *RefreshTime* = '15min'	service provider
(4) If *UserSpeed* = 'Fast' Then Set *RefreshTime* = '3min'	service provider

Table 2. An excerpt of profile resolution directives

Profile Resolution Directive
(5) *setPriority AllowRecommandations = (SPPM, UPM)*
(6) *setPriority Coordinates = (UPM, OPM)*
(7) *setPriority MediaQuality = (SPPM, UPM)*
(8) *setPriority UserSpeed = (UPM, OPM, SPPM)*

In order to show some of the profile resolution directives and policies which determine service adaptation, we report one of the test cases we have considered: An hypothetical user is browsing around a hypothetical town with a PDA in his hands. We appropriately divided the town into bi-dimensional cells identified by a pair of coordinates, further assuming that some of the cells are covered by a GPRS connectivity service, while others by a more efficient WiFi HotSpot service. Movements of our user and context changes are simulated. The service needs to continuously adapt to user's changes of context. The screen-shots in Figure 3 show how different ads are displayed depending on the user's location and time of the day. In addition, the presentation is properly adapted to the user's device capabilities and available bandwidth. The adaptation parameters are set by the IE module, upon the evaluation of policies declared by the user and by the service provider. For instance, we suppose the user declared policy (1) in Table 1 to request high-quality multimedia content when using his PDA. Similarly, service providers can declare policies for determining content and presentation directives. A possibly conflicting policy (2) is declared by the

service provider, stating to deliver low-quality multimedia contents when the available bandwidth drops below a certain threshold. The refresh rate of the service is determined by policies (3) and (4). In particular, policy (3) determines a long refresh interval when the user is moving slowly, while policy (4) shortens the refresh interval when the user is moving fast.

The firing of policy rules may depend on the aggregated profile obtained by the Merge module, which in turn relies on profile resolution directives. We remind that this kind of directives can only be specified by the service provider. Some profile resolution directives are given in Table 2. For instance, directive (8) is intended to solve conflicts due to different estimations of the user's current speed given by different entities. The service provider gives higher confidence to the value provided by the UPM, since speed can be estimated precisely by user-side sensors (e.g., supplied by car appliances or GPS-enabled devices). If no value for speed is given by the user, the value provided by the operator (if present) is taken into account; otherwise, the value inferred by the service provider analyzing the history of the user's location is chosen.

5. RELATED WORK

Many research groups and companies have been working, at different levels, to provide effective solutions for service adaptation and personalization in a multi-device and mobile environment. In the following, we report on the efforts we consider closer to our work. Our approach is similar to the one underlying DELI [8] and Intel *CC/PP SDK* [6]. However, our framework provides a finer control on profile aggregation, and includes a policy mechanism. Various other architectures address the problem of service adaptation in mobile environments [2, 7, 9, 11, 14, 21]. The distinguishing feature of our architecture is that in our case the adaptation process is driven by the evaluation of distributed profile data and policies which are stored on and handled by modules in the trusted domain of their data source. For example, the Houdini framework [14] provides a mechanism of rule evaluation against user context information that is similar to ours. However, policy rules in [14] are specified by users only and stored on and handled by a single module. Since efficiency is a major concern in their applications this module is in the domain of the service provider. Moreover much less emphasis is given to conflict resolution issues.

We claim that our framework is able to support a wide range of context-aware applications, which can profitably exploit it for adapting and personalize their services to users. Even focusing on the domain of the application described in this paper, the number of related works is large

(e.g., [12, 15, 23]). In particular, the ViaVis' Proximity Marketing allows users to personalize the reception of advertisements in terms of their location, time and content. Again, a main difference in our service is that profile data and user preferences are not stored and managed at the service provider, but kept in the user trusted domain (at the UPM). This has several advantages especially when multiple services need to access overlapping portions of profile data (centralized updates, privacy control). Moreover, our solution provides users with a richer set of personalization parameters, which allow for a better definition of user contextual situations and a finer personalization of the service.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented a framework supporting adaptation and personalization of mobile Internet services. We illustrated the software architecture adopted for its implementation, and a prototype service used as a test-bed. Even if the main components of the framework are consolidated, various extensions and enhancements are possible and already foreseen. In particular, our profile technology can be meaningfully coupled with various content-based services and recommendation systems. Thanks to our framework, these systems can exploit both the explicit rules expressed as preferences by users, and the information regarding the context the users are immersed in. Moreover, various interesting works exist which are focused on gathering information about the user and its environment on the basis of sensors (e.g., [1, 22]). We believe that the integration of numerous sources of profile data (i.e., sensors) and related processing modules in our framework would be a natural and promising research direction.

REFERENCES

[1] AmbieSense. European project # IST-2001-34244. http://www.ambiesense.com/

[2] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli. Context-aware Middleware for Resource Management in the Wireless Internet. *IEEE Trans. on Software Engineering*, 29(12):1086–1099, IEEE, 2003.

[3] C. Bettini, S. Jajodia, X. Wang, and D. Wijesekera. Provisions and obligations in policy rule management. *Journal of Network and Systems Management*, 11(3):351–372, Kluwer, 2003.

[4] C. Bettini and D. Riboni. Profile Aggregation and Policy Evaluation for Adaptive Internet Services. In *Proc. of The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous)*, 2004.

[5] H. Boley, S. Tabet, and G. Wagner. Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In *Proc. of the first Semantic Web Working Symposium*, pages 381–402, 2001.
[6] M. Bowman, R. D. Chandler, and D. V. Keskar. Delivering Customized Content to Mobile Device Using CC/PP and the Intel CC/PP SDK. *Intel Technical Report*, Intel, 2002.
[7] K. H. Britton, R. Case, A. Citron, R. Floyed, Y. Li, C. Seekamp, B. Topol, and K. Tracey. Transcoding. Extending e-business to new environments. In *IBM Systems Journal*, 40(1):153–178, IBM, 2001.
[8] M. Butler. DELI: A DElivery context LIbrary for CC/PP and UAProf. *External Technical Report HPL-2001-260*, HP, 2002.
[9] H. Chen, T. Finin, and A. Joshi. Semantic Web in the Context Broker Architecture. In *Proc. of IEEE International Conference on Pervasive Computing and Communications (PerCom2004)*, pages 277-286, IEEE, 2004.
[10] The Apache Cocoon Project. Apache Software Foundation. http://cocoon.apache.org
[11] C. Efstratiou, K. Cheverst, N. Davies, and A. Friday. An Architecture for the Effective Support of Adaptive Context-Aware Applications. In *Proc. of the International Conference on Mobile Data Management*, pages 15–26, IEEE, 2001.
[12] ELBA: European Location Based Advertising. European project # IST-2001-36530. http://www.e-lba.com/
[13] B. Grosof. Prioritized Conflict Handling for Logic Programs. In *Proc. of Symposium on Logic Programming (ILPS)*, pages 197-211, 1997.
[14] R. Hull, B. Kumar, D. Lieuwen, P. Patel-Schneider, A. Sahuguet, S. Varadarajan, and A. Vyas. Enabling Context-Aware and Privacy-Conscius User Data Sharing. In *Proc. of the International Conference on Mobile Data Management*, pages 187–198, IEEE, 2004.
[15] IMAP: An innovative Interactive Mobile Advertising Platform. European project # IST-2001-33357. http://www.imapproject.org/imapproject/hmain.jsp
[16] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible Support for Multiple Access Control Policies. In *ACM Transactions on Database Systems*, 26(2):214–260, ACM press, 2001.
[17] Jena 2 - A Semantic Web Framework. http://jena.sourceforge.net/
[18] Joseki - The Jena RDF server. http://www.joseki.org
[19] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. Butler, and L. Tran, editors. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, W3C Recommendation, 15 January 2004. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/
[20] J. Krogstie, K. Lyytinen, A. L. Opdahl, B. Pernici, K. Siau, and K. Smolander. Mobile Information Systems - Research Challenges on the Conceptual and Logical Level. In *Proc. of ER'02/IFIP8.1 Workshop on Conceptual Modelling Approaches to Mobile Information Systems Development*, pages 1-13, Springer, 2002.
[21] S. Riché and G. Brebner. Storing and Accessing User Context. In *Proc. of the International Conference on Mobile Data Management*, pages 1-12, IEEE, 2003.
[22] D. Terdinam. Soon, Marketing Will Follow You. Wired News, 2003. http://www.wired.com/news/technology/0,1282,61597,00.html
[23] ViaVis Mobile Solutions Inc. http://www.viavis.com
[24] User Agent Profile Specification. WAP-248-UAProf. http://www.wapforum.org/