A Scheme for the Introduction of 3rd Party, Application-Specific Adaptation Features in Mobile Service Provision

Nikos Houssos, Nancy Alonistioti, and Lazaros Merakos

Communication Networks Laboratory, Department of Informatics & Telecommunications,
University of Athens, 15784, Athens, Greece
{nhoussos,nancy,merakos}@di.uoa.gr

Abstract. The long term vision of beyond 3G wireless communications describes an evolution towards beyond 3G systems. The ultimate goal of this evolution is a dynamic environment that enables the delivery of situation-aware, personalised multimedia services over heterogeneous, ubiquitous infrastructures. Under this perspective, the need is emerging for applying, in a systematic way, adaptability and reconfigurability concepts for service delivery in largely diverse contexts. Moreover, it is widely recognised that services will be increasingly developed by independent third parties. The present contribution complements previous work by the authors, related to mediating service provision platforms and advanced adaptability and profile management frameworks, by introducing mechanisms that allow third parties to dynamically enhance the service delivery and adaptation middleware in order to achieve application-specific customisations in various aspects of the mobile service provision process.

1 Introduction

The evolution of mobile networks and systems to 3rd generation and beyond is expected to bring about substantial changes to telecommunication service provision. In the envisioned beyond 3G era, a plethora of functionality-rich, profit-creating value-added services (VAS)[1] should be delivered to mobile users [1] over an unprecedented variety of infrastructures and contexts, which could not be predicted or catered for during service design and development. These extensively demanding requirements indicate the need for adaptability of applications as well as service provision and delivery procedures to highly diverse environments. Adaptability, an inherently challenging task, is further complicated by the fact that adaptation intelligence should be generic, portable and interoperable so that it can be flexibly applied to a diversity of entities in different circumstances.

[1] In the present paper the terms *service*, *value-added service (VAS)* and *application* are used interchangeably to refer to an information technology product which is directly accessible and perceptible to the end user and whose value resides mostly in functionality and content, rather than transport or connectivity.

J.-B. Stefani, I. Demeure, and D. Hagimont (Eds.): DAIS 2003, LNCS 2893, pp. 15–28, 2003.

In previous research efforts, the authors have developed schemes for addressing the above issue and incorporated them in a middleware platform for service provision [2]. In this paper, we complement this work by providing detailed mechanisms for enabling adaptation logic to be run-time injected into the service provision middleware by 3rd parties. This feature is important, since different services may require different algorithms for matching service requirements with context parameters. As a simple example one could consider a context parameter that is expressed in terms of dimension (e.g., terminal screen resolution). Two services could have the same value in their profile as a dimension requirement, but the algorithm for matching it with the corresponding context value could differ (e.g., one algorithm would require the currently supported screen resolution of the terminal to be just greater than or equal to the value in the service profile, while another one could additionally require that the width/height ratio would be equal to a specific quotient). Implementing such an algorithm is something trivial for the service developer; however, the lack of frameworks that would enable the dynamic loading of suitable service-specific algorithms does not currently allow this type of adaptation flexibility. In this contribution we demonstrate that this can be achieved through a procedure that does not incur significant overhead to the service creator and mainly involves specifying adaptation metadata in RDF/XML [3] and developing the algorithms according to certain simple, non-restrictive guidelines.

The rest of this document is organised as follows: At first, we present the environment where the proposed adaptation mechanism was integrated and applied, namely a software platform for provision of services over 3G mobile networks. We then discuss the platform support for 3rd party adaptation logic introduction. This discussion includes the features offered by the platform, a detailed view of the overall procedure from a third-party perspective and the supporting mechanisms that we have developed. The last sections of this paper are dedicated to summary, conclusions and acknowledgements.

2 Service Provision Platform

The present section introduces a distributed software platform for the flexible provision and management of advanced services in next generation mobile networks. The platform incorporates intelligent adaptation and reconfiguration mechanisms as well as advanced support for 3rd party adaptable service deployment and management. Before elaborating on the internal platform architecture, we provide a brief discussion of the business and service provision models supported by the platform.

Note that the detailed architecture and functionality of the platform, as well as details about the corresponding business models, has been presented by the authors in previous work [5] and is thus beyond the scope of the present paper. Only the basic aspects that are useful for presenting the 3rd party support mechanisms are included herein.

2.1 Business and Service Provision Model

The proposed framework is designed to support flexible, advanced business models that encourage among market players the establishment of strategic partnerships, which ultimately benefit end users by significantly enhancing service provision. The main business entities involved in such models are the following:

Mobile User: The mobile user is the actual consumer of services.

Mobile operator: This is the entity that operates the network infrastructure for mobile user access to services and will typically also provide third-party access to its network through a standardised API (e.g., OSA/Parlay [6], JAIN).

Platform operator: This is the entity that owns and administers the software platform for service provision.

Value-Added Service Provider (VASP): This is the provider (and typically also the developer) of the end-user application.

The platform operator acts as a single point of contact ("one-stop-shop") for:

- *VASPs* that are able to register their services with the platform and this way to have them delivered to a large number of mobile users over multiple networks.
- *Mobile users* that are given the ability to discover, select, download and execute registered value-added services in a flexible, adaptable and personalised manner.

To accomplish these tasks, the platform operator is engaged into business relationships with users, mobile operators and VASPs. A prior subscription with the VASP for using an application is not required for the user, since a dynamic service discovery, download and execution model is applied.

It is worth noting that it is possible for one single entity to undertake several of the roles described above and vice versa. For instance, a mobile or platform operator may also develop its own services and act as a VASP.

The service provision model supported by the RCSPP can be summarised as follows: Before a service becomes available to end-users, it is automatically deployed based on VASP-provided detailed service metadata. This procedure may include sophisticated actions like reconfiguration of the underlying networking infrastructure. From then on, the user is able to dynamically discover, download and execute the service, without the need for a prior subscription with the VASP.

2.2 Overview of Platform Architecture and Functionality

In this section we briefly present a distributed middleware platform that supports adaptability in mobile service provision. The functionality of the platform, which is called Reconfiguration Control and Service Provision Platform (RCSPP), comprises automated procedures for service deployment that include appropriate reconfiguration of the underlying network for optimal service delivery. In addition to that, an intelligent context-aware mobile portal is offered to the end-user, where procedures like service discovesry, downloading and adaptation are fully tailored to terminal capabilities, user preferences and network characteristics.

The architecture of the platform is depicted in Fig. 1. The main components of this architecture are the following:

– The *Reconfiguration Control and Service Provision Manager (RCSPM)* is the central platform component in that it co-ordinates the entire service provision and management process.
– The *Charging, Accounting and Billing (CAB)* system [8] is responsible for producing a single user bill for service access and apportioning the resulting revenue between the involved business players.

– The *End User Terminal Platform (EUT)* [9] resides in the mobile terminal (it is not depicted in Fig. 1) and includes functionality such as service downloading management, GUI clients for service discovery and selection, capturing of event notifications as well as service execution management.
– The *VASP* (or VASP Service Platform Client) component of the platform is located in the VASP domain (e.g., enterprise servers) and handles secure terminal access to a repository of application clients, while also providing web interfaces to RCSPM functionality, through which VASPs can carry out sophisticated service management operations (e.g., service deployment).

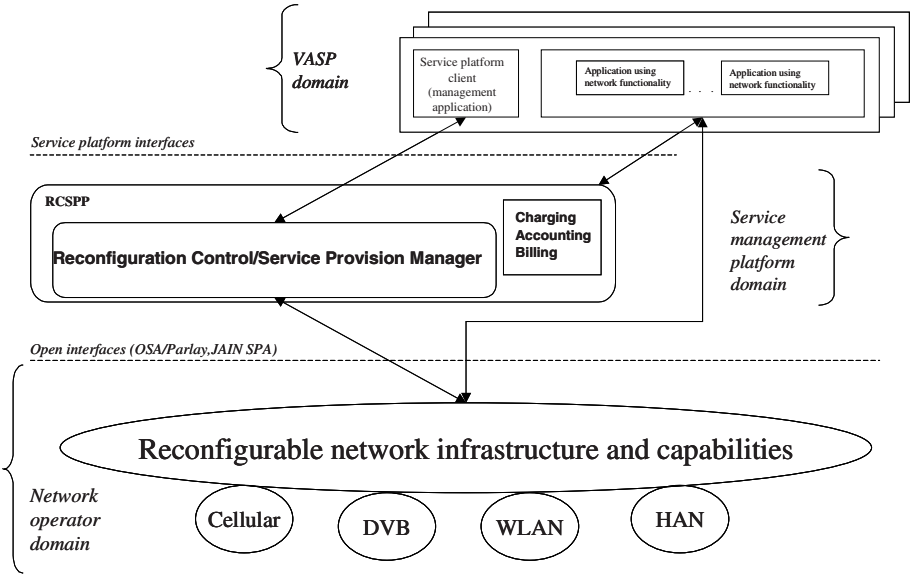

Fig. 1. Architecture for flexible service provision in 3G and beyond networks.

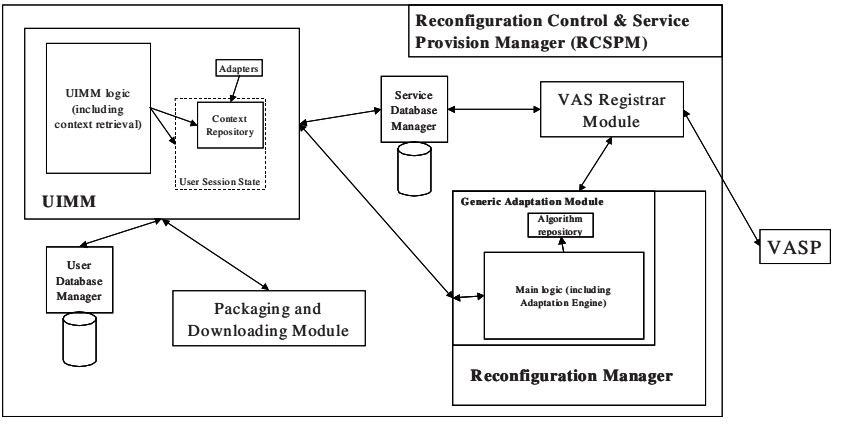

Fig. 2. Internal RCSPM architecture (not complete; only adaptability-relevant modules are depicted).

Adaptability management and 3rd party support in the RCSPP is based on the architectural components depicted in Fig. 2 and presented in the following:

- The VASP component of the platform, as described above.
- A subset of the components and interfaces of the RCSPM, which are described below:

  The *User Interaction Management Module (UIMM)* is responsible for providing the user with a highly personalisable, context-aware mobile portal. It manages user sessions with the RCSPP, maintains relevant contextual information and co-ordinates user-related operations like service discovery, selection, adaptation and downloading as well as user profile management.

  The *VAS Registrar Module (VASREGM*) is responsible for interacting with 3rd party service providers. Through the VASREGM the platform operator provides VASPs with a way to automatically deploy their services. The VASP compiles an formally specified profile of service attributes. Based on these attributes, the VASREGM co-ordinates service deployment, including various actions like reconfiguration of the underlying infrastructure and uploading of service components to the RCSPM. The service provider is able to manage (add/delete/update) its services via a convenient web interface.

  The *Reconfiguration Manager (RCM)* undertakes network, platform and service reconfigurability. The RCM is responsible for executing the appropriate reconfiguration actions on the underlying network during VAS management procedures (registration/de-registration/update), triggered by the VASP. The RCM also comprises a generic adaptation module [2] that is used for supporting adaptation through functions like intelligent profile matching. The adaptation module is able to dynamically load adaptation algorithms from a local repository or remote network locations. These algorithms can be developed by VASPs and inserted to the repository during the service registration operation with the mechanisms described in Section 3.

  The *Packaging and Downloading Module (PDM)* [7] is addressing an aspect of adaptable service delivery by being responsible for dynamically creating a single bundle that contains all the software components and other supporting resources (e.g., images, etc.) required for executing a service and for making it available for download to the mobile client. The single archive produced is dynamically tailored to the context of the particular VAS selection request.

The RCSPM also includes database managers that provide interoperable access to the persistent service, user and network profile repositories hosted by the platform.

3 3rd Party Support for Adaptable Service Provision

The current section presents in more detail the 3rd party support features of the RCSPP. At first, we describe the data and mechanisms that the platform provider makes available to VASPs for deploying their applications. Next, we present the detailed sequence of actions that are required from a VASP for the flexible and adaptive provision of a VAS and outline the general structure of the service management operations offered by the RCSPP. Subsequently, we elaborate on the implementation of certain mechanisms that are crucial for 3rd party adaptation support, namely the speci-

fication of service profiles, with a particular focus on metadata about adaptation algorithms, as well as the development and loading of the actual adaptation logic.

3.1 What Does the RCSPP Provide?

To offer third parties the capability of developing their own customised adaptation intelligence and injecting it into the system, the platform provider carries out the following actions:

- Supplies service developers with the VASP platform component that enables remote service deployment and data management, normally through an easy to use graphical user interface, typically implemented as a simple web HTML page (as is our demonstrator) or a Java application/applet.
- Publishes the formal specification of the service profile structure. The latter should be defined using a data representation approach that promotes interoperability and extensibility, such as XML or RDF. These approaches include constructs that are employed for describing profile structures, like XML Document Type Definition (DTD) and XML Schema for XML and RDF Schema for RDF. In our prototype, we have used XML and XML DTD for service profile representation and structure specification, respectively. This particular choice was made because it is characterised by simplicity, ease of implementation, although it is extensible and able to represent arbitrarily complex profiles.
- Provides a way for the VASP to bind service requirements in terms of the values of particular context parameters with the algorithms that shall undertake the matching/adapting task regarding these parameters. In our prototype we have devised ways to achieve that in RDF and XML, as described in Section 3.3.1.
- Provides all the necessary information that is required for a third party to develop a custom algorithm. This includes the detailed description of the platform-internal profile representation format and typical guidelines for code extension (e.g., which interfaces can be extended). Moreover, the platform enables the automatic loading of the algorithms into the adaptation module repository during the service registration operation. Notably in our prototype, custom algorithm development is supported only in Java, a language well-suited for extensibility.
- Makes public the type of context profile elements for which there is a default algorithm, as well as the algorithms themselves (including their implementation). The default algorithm is used for adaptation decision-making regarding a specific profile element in the case that the VASP has not explicitly identified an algorithm for this particular element.

3.2 What Does the VASP Have to Do?

Before making use of the platform, a third-party application provider should obtain the corresponding authorisation. This procedure, a part of which is performed through off-line business interaction between platform operator and VASP, is completed when the latter becomes the information (e.g., an SPKI certificate) that enables its authentication by the platform.

The deployment of an application and its delivery to end-users is performed with the support of the platform and requires the following actions from the side of registered VASPs:

1. Development of the service logic. Various versions and implementations may exist for a single service. No particular constraints in terms of programming languages, methodologies and tools is imposed by the platform on the service creator. However, in case a component-based development approach is followed, the on-demand composition of the optimal service configuration is possible, as described in Section 2.1.
2. Specification of service requirements in terms of context parameters. Environmental parameters are bound to all elements (e.g., service client, version, component, implementation) that model the downloadable part of a service offering. Contextual requirements have different "scope", depending on the type of element with which they are associated. That is, the service client software context requirements concern all versions and, therefore, if these requirements are not compatible with a user's current service provision environment no version of the application can be provided to this particular user. Likewise, the version requirements pertain to all components (core and optional) and the component requirements relate to all implementations.
3. Development of custom adaptation/matching algorithms for certain context parameters. This is necessary in case the platform does not provide a default algorithm for these parameters or the provided default algorithm is not appropriate for the application in question. The development of the algorithm should follow the relevant guidelines publicly announced by the platform operator. In general, a VASP is required to implement custom algorithms only for a small number of parameters that are particularly significant for the optimal provision of the service and for which special requirements exist.
4. Identifying and expressing the necessary metadata for service registration in the appropriate format. This metadata constitutes the service profile, which includes a variety of information, as elaborated in Section 3.3.1. A crucial part of the service profile relates to the contextual requirements of the application, for example terminal capabilities, network characteristics and user preferences and status, together with the identification of custom-made algorithms associated with individual parameters, if any. The service descriptor in our prototype is defined in XML and should be compatible with a particular XML DTD. However, parts of the profile, like the requirements and their associated algorithms can be specified in RDF/XML (see Section 3.3.1).
5. Performing the service registration operation through the platform's VASP component.

The platform makes available to VASPs service registration and the relevant service data management operations (service update and deletion), which all have the form of distributed transactions. Typical sub-tasks of these transactions are the following:

- Validation checks, applied on the VASP-originated service profile. The latter should comply (e.g., in terms of billing/pricing data) with the (prior) business agreement between VASP and platform operator. This task is performed by the VASREGM.
- Insertion/update/removal of information stored in the services database. This task is handled by the VASREGM, which makes use of the Service Database Manager interface.

- Insertion/update/removal of algorithm implementations that are stored in the corresponding repository, maintained by the adaptation module of the RCM. This task, which mainly comprises the uploading to the repository of the appropriate binary files/archives, is handled by the VASREGM and involves interactions with the RCM.
- Reconfiguration actions on the underlying infrastructure. The execution of these actions, which are determined by suitable interpretation of service metadata and context information (e.g., network capabilities/load), is co-ordinated by the RCM that receives high-level events (e.g., an update of the pricing policy and/or traffic flow information regarding service "X") and maps them to appropriate signaling (based on standardised or proprietary APIs/protocols) with network/system components/devices (e.g., routers, billing systems).

The co-ordination of the above mentioned transaction is handled by the VASREGM. For example, if during VAS update the reconfiguration of underlying network routers fails, any changes to the service database and algorithm repository should not be committed.

3.3 Supporting Mechanisms

3.3.1 Specification of Service Metadata

The service provision functions of the RCSPP are largely dependent on the availability of accurate metadata regarding the service, which is referred to as the *service profile*. The service profile is formulated by the VASP and communicated to the RCSPM during the service registration operation. The RCSPM maintains this information in an appropriate service database, whose data can be updated at any time by authorised third parties. The current section first describes the contents of the service profile, then elaborates on our choice of the format of its representation and finally presents how adaptation algorithm metadata can be included in service metadata, with the support of RDF.

The application profile encompasses a variety of information, such as:

- General data about the service, like name, version, description and VASP information.
- Data describing the service software architecture, including any optional components.
- Requirements from terminals.
- Requirements from network infrastructure. These include pure technical characteristics like network type (e.g., GSM/GPRS, UMTS, WLAN) and available bandwidth as well as requirements of a hybrid business/technical nature such as revenue sharing policy of network operator and availability of open interfaces (e.g., OSA/Parlay) to network functionality.
- User preferences that are supported by the application (e.g., available languages).
- VAS-specific information about tariffing/billing as well as revenue sharing between VASP and platform provider.
- Security data.

Since service metadata is subject to processing and exchange in different administrative domains, it should be represented in a storage-independent format that promotes

interoperability. Two current recommendations of the World-Wide-Web Consortium, XML [11] and RDF [12] can be thus considered as prime candidates for this task. XML is a ubiquitous, widely adopted by industry meta-language, which enables the representation of hierarchical data structures and incorporates capabilities for the definition of new vocabularies and schemata. RDF is a more complex framework that can be used for the encoding of arbitrarily complex information in the form of directed labeled graphs. Notably, XML is the most common format for serialising RDF data.

In general, XML is easier to use and manipulate, while RDF has greater capabilities for expressing semantically rich information. RDF results in more clarity, since there is an explicit format interpretation of any RDF-encoded data, based on the RDF Model Theory [13]. Consequently, a certain piece of information can be represented in RDF in exactly one, unique way, while in XML many different encodings with the same meaning are possible [16]. This advantage of RDF, however, comes at the cost of being more verbose and significantly more complex. The latter characteristic makes it less attractive for the majority of users and developers [14].

```
<!--DTD for VAS descriptor-->
<!--Author: Nikos Houssos, UoA-CNL-->
<!ELEMENT VAS (VASGEN, VASP,  SOFTWAREARCH, SECURITY)>
<!ELEMENT VASGEN (VASName, VASID?, VASVersion, VASDescription, SubscriptionType, Category, Keywords, Availability,
UpdateDescription?, OSA_Parlay_InfoGen?)>
.    .    .
<!ELEMENT VASP (VASPName, VASPType, VASPPublicKey, VASPReference)>
.    .    .
<!ELEMENT SOFTWAREARCH (ServerPart, ClientPart)>
.    .    .
<!ELEMENT ClientPart (ServiceClientVersion+, ContextReq)>
<!ELEMENT ContextReq (TermReq?, NetworkReq?, UserPref?, UserStatus?, OtherReq?)>
<!ELEMENT TermReq (#PCDATA)>
<!ELEMENT NetworkReq (#PCDATA)>
<!ELEMENT UserPref (#PCDATA)>
<!ELEMENT UserStatus (#PCDATA)>
<!ELEMENT OtherReq (#PCDATA)>
<!ELEMENT ServiceClientVersion (CorePart, OptionalPart?, ContextReq, PricingModel, TariffClass, CostDescription, FlowMonitoring,
QoSIndicator)>
.    .    .
<!ELEMENT CorePart (Component+)>
<!ELEMENT OptionalPart (Component+)>
<!ELEMENT Component (Description, Implementation+, ContextReq?, OptionalPart?)>
.    .    .
<!ELEMENT Implementation (Codebase, ContextReq?)>
.    .    .
<!ELEMENT SECURITY (IPRProtection, Confidentiality, VASConditionsOfUse, SecurityDomain, SPKICertificate)>
.    .    .
```

Fig. 3. XML DTD for service profile (Note: only part of it and simplified for readability)

In our approach, the service profile is encoded in XML; an XML Document Type Definition (DTD) is employed for defining the application metadata vocabulary. However, there is the possibility for the incorporation of RDF models as values of certain XML elements, as XML CDATA sections [15]. This has been considered necessary for certain elements, like contextual requirements, which can include information that is not a priori predictable and thus is not possible to include in an XML DTD or XML Schema that is universally adopted by all VASPs. This way, VASPs are allowed to insert in the service profile context requirements, while still producing XML documents that are valid and compatible with the service metadata DTD. The inherent greater extensibility capabilities of RDF [17] have been the main reason for this choice, although some researchers claim that they create challenging validation problems and can potentially create the risk for storing of incorrect data [18].

The above are exemplified by the corresponding DTD displayed in Fig. 3.

An important issue relates to how the VASP can specify custom algorithms that should be used for adaptation/matching of service profiles according to the current service provision context. We have identified a solution for this issue, based on the assumption that the context requirements metadata is encoded in RDF. We have defined a specific RDF property called *ComparatorAlgorithm*[2], whose subject can be any RDF resource and whose value is of type *ComparatorDescriptor*. The latter is a class of RDF resources, which represents adaptation/matching algorithms. *ComparatorDescriptor* is a sub-class of the more general *AlgorithmDescriptor* class. A *ComparatorDescriptor* object can have various properties, which provide data adequate for locating and loading the algorithm implementation (e.g., *FullyQualifiedName*, *ImplementationLocation*) or are used for providing general information about the algorithm (e.g., *AlgorithmDescription*, *DeveloperDescriptor*). The corresponding declarations are included in an RDF Schema that we have defined and made publicly available. The schema is shown in Fig. 4.

To illustrate the above, we provide in the following some examples, in which we use certain terminal capability attributes (as specified in the OpenMobileAlliance UAProf specification [19]) as cases of context parameters and assume an RDF/XML serialisation of RDF data.

If the value of the RDF property (context parameter) is compound, it is augmented with an `ComparatorAlgorithm` property element. For instance, the following definition for the `JVMVersion` property:

```
<prf:JVMVersion>
        <rdf:Bag><rdf:li>SunJRE/1.2</rdf:li></rdf:Bag>
</prf:JVMVersion>
```

becomes:

```
<prf:JVMVersion>
        <rdf:Bag><rdf:li>SunJRE/1.2</rdf:li></rdf:Bag>
        <alg:ComparatorAlgorithm>
                <alg:ComparatorType>Matcher</alg:ComparatorType>
        <alg:FullyQualifiedName>gr.uoa.di.cnl.adaptation.VersionMatcher<
/alg:FullyQualifiedName>
        <alg:ImplementationCodebase>http://www.cnl.di.uoa.gr/People/nhou
ssos/Impl/classes/</alg:ImplementationCodebase>
        </alg:ComparatorAlgorithm>
</prf:JVMVersion>
```

Lest the value of the RDF property (context parameter) is atomic (RDF Literal), we use the standard RDF technique for representing higher arity relations using binary relations [17]. Thus, the principal value of the property is included as an rdf:value property and a `ComparatorAlgorithm` property element is also added. Thus, the following definition for a `ScreenSize` property:

```
<prf:ScreenSize>1024x768</prf:ScreenSize>
```

becomes:

```
<prf:ScreenSize>
    <rdf:value>1024x768</rdf:value>
```

[2] In the reference to RDF resources in this paper we present them as "local" resources; we omit the globally qualified name for the sake of readability and simplicity of the text. Thus, for example, we write "ComparatorDescriptor" instead of "http://www.cnl.di.uoa.gr/People/nhoussos/Schemata/AlgorithmSchema-20030622#ComparatorDescriptor" or "alg:ComparatorDescriptor"

```
    <alg:ComparatorAlgorithm>
<alg:ComparatorType>Matcher</alg:ComparatorType>
<alg:FullyQualifiedName>gr.uoa.di.cnl.adaptation.ScreenSizeMatcher</alg:
FullyQualifiedName>
        <alg:ImplementationCodebase>http://www.cnl.di.uoa.gr/People/nhou
ssos/Impl/classes/</alg:ImplementationCodebase>
    </alg:ComparatorAlgorithm>
</prf:ScreenSize>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#" xml:base="http://www.cnl.di.uoa.gr/People/nhoussos/Schemata/AlgorithmSchema-20030622">
    <!-- Algorithm Descriptor Definition -->
    <rdf:Description rdf:ID="AlgorithmDescriptor">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:comment> The base class for resource classes that represent algorithms.  </rdfs:comment>
    </rdf:Description>
    <!-- Comparator Descriptor Definition -->
    <rdf:Description rdf:ID="ComparatorDescriptor">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#AlgorithmDescriptor"/>
        <rdfs:comment>
          A resource that represents an algorithm used for adaptation/matching.
      </rdfs:comment>
    </rdf:Description>
    <!-- Properties common for all AlgorithmDescriptor resources. -->
    <rdf:Description rdf:ID="FullyQualifiedName">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#AlgorithmDescriptor"/>
        <rdfs:comment>
      Provides the fully qualified name of the algorithm.
      Example: "gr.uoa.di.cnl.adaptation.LocationMatcher"
      </rdfs:comment>
    </rdf:Description>
    <rdf:Description rdf:ID="AlgorithmDescription">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#AlgorithmDescriptor"/>
        <rdfs:comment>Provides a textual description of the algorithm.</rdfs:comment>
    </rdf:Description>
    <rdf:Description rdf:ID="DeveloperDescriptor">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#AlgorithmDescriptor"/>
        <rdfs:comment>Provides a textual description of the entity that has developed the algorithm.</rdfs:comment>
    </rdf:Description>
    <rdf:Description rdf:ID="ImplementationLocation">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#AlgorithmDescriptor"/>
        <rdfs:comment>
      Indicates the network location (codebase) from where the algorithm implementation may be retrieved.
      Example: "http://www.cnl.di.uoa.gr/People/nhoussos/Algorithms/"
      </rdfs:comment>
    </rdf:Description>
    <rdf:Description rdf:ID="TargetResourceInstance">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#AlgorithmDescriptor"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
        <rdfs:comment> Indicates the resource instance on which the algorithm will be applied. This property has
meaning only when the algorithm metadata is specified separately from the context requirements metadata.
</rdfs:comment>
    </rdf:Description>
    <!-- Properties specific to ComparatorDescriptor resources. -->
    <rdf:Description rdf:ID="ComparatorType">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#ComparatorDescriptor"/>
        <rdfs:comment>Indicates the type of the Comparator algorithm (Adaptor or Matcher)</rdfs:comment>
    </rdf:Description>
</rdf:RDF>
```

Fig. 4. Algorithm Description RDF Schema

Note that in the above, the alg: prefix refers to the publicly accessible Internet location http://www.cnl.di.uoa.gr/People/nhoussos/Schemata/AlgorithmSchema-20030622#.

As is obvious from the above, the additional property element increases the size of the service profile. However, this additional size should not be a problem, mainly for two reasons:

- Typically a VASP would provide customised algorithms only for a small set of context parameters. The default algorithms (see Section 3.3.2) should suffice for the large majority of cases.
- Service registration as well as the other procedures utilising the service profile are management plane procedures that are not time-critical and do not involve communication over the costly and resource-constrained wireless link to the mobile terminal. Thus, lack of resources is not a principal issue in this case.

3.3.2 Development and Loading of Adaptation Algorithms

Adaptation algorithms, as stated in previous sections, can be developed by VASPs. In this section we elaborate on the framework that enables the easy development and dynamic loading of these algorithms.

Every adaptation procedure includes a phase during which crucial decisions are being made regarding what is the optimal adaptation action that should take place [2]. This procedure is performed based on the comparison of current context parameters and the contextual requirements of the adaptable entity. In the RCSPP this procedure is accomplished by a generic adaptation decision engine [2] that accepts as input two profiles, called the *adaptor* (representing context information) and the *adaptee* (representing the adaptable entity).

All types of profiles in our implementation are represented in a common, internal, object-oriented format, depicted in Fig. 5 [2]. Notably, the adaptee and adaptor profiles are instances of the same class (Profile). Profiles are instances of profile elements and consist of single profile attributes and sub-profiles (this is an application of the Composite design pattern [10]).

The adaptation algorithm is typically encapsulated in an object that implements the Adapter or Matcher interface, is aggregated in a ProfileElement instance and is dynamically loaded. This way, new adaptation algorithms for specific attributes can be introduced without the need to reprogram the code of the ProfileElement classes that represent those attributes (this is an application of the Strategy design pattern [10]).

The RCSPM is able to construct profiles according to the above representation, containing the service information stored in the application database after service registration. Based on the RDF-based algorithm metadata that has been provided by the VASP with the mechanism explained in the previous section, the constructed service profile hierarchies aggregate the appropriate algorithms specified by the VASP. Notably, the adaptation module has a default algorithm per each context parameter, which is used when no customised version is introduced by the VASP.

The implementation of an algorithm in Java (this is the only language supported in our prototype), is quite straightforward. The developer creates a class that implements one of the *Adapter* or *Matcher* interfaces. The algorithm logic typically retrieves the value of certain situational parameters from the context (adaptor) profile and, based on them, reaches a decision according to service-specific criteria. An example of a very simple algorithm regarding matching screen sizes is depicted in Fig. 6.

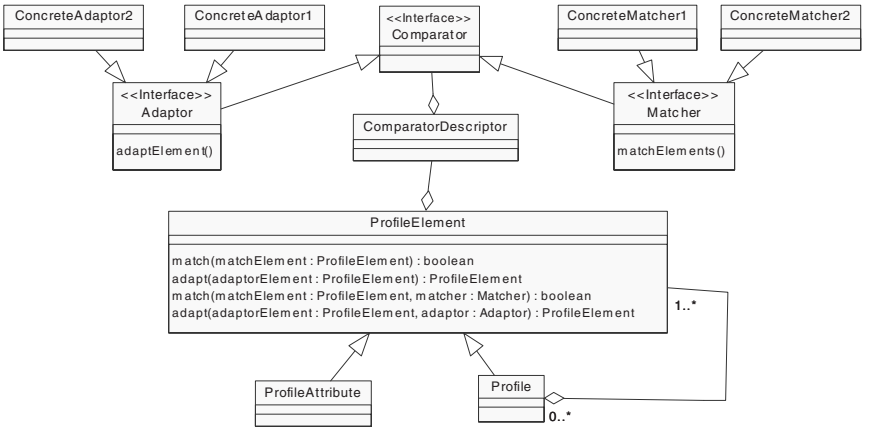

Fig. 5. Generic profile representation.

```
public class ScreenSizeMatcher implements Matcher          {
        public boolean matchElements( ProfileElement element1, ProfileElement element2 )       {
                boolean returnValue = false;
                int w1, w2, h1, h2;
                double ratio1, ratio2;
                if ( ( element1.getType().equals( "gr.uoa.di.cnl.ScreenSize" ) )
                        && ( element2.getType().equals( "gr.uoa.di.cnl.ScreenSize" ))    {
                        w1 = ((Integer) ((ProfileAttribute) element1).getValueAt( 0 )).intValue();
                        w2 = ((Integer) ((ProfileAttribute) element2).getValueAt( 0 )).intValue();
                        h1 = ((Integer) ((ProfileAttribute) element1).getValueAt( 1 )).intValue();
                        h2 = ((Integer) ((ProfileAttribute) element2).getValueAt( 1 )).intValue();
                        ratio1 = w1/h1;
                        ratio2 = w2/h2;
                        if ( ( ratio1 == ratio2  ) && ( w2 < 1.2*w1 ) && ( w2 > 0.8*w1 )
                                                && ( h2 < 1.2*h1 ) && ( h2 > 0.8*h1 ) )
                                return true;
                }
                return returnValue;
        }
}
```

Fig. 6. Example Matcher implementation.

4 Summary – Conclusions

The significance of adaptation in next generation mobile systems and services is widely recognised. Adaptation capabilities form a principal enabler of ubiquitous, seamless service provision over highly diverse underlying infrastructures. Moreover, third-party VASPs are expected to play an increasingly significant role in the development and delivery of mobile services. The present paper has introduced a scheme for offering third parties advanced capabilities for performing their own adaptations and customisations to the process of the provision of their applications. The proposed scheme exploits knowledge representation and object-orientation techniques to achieve this goal without incurring excessive additional overhead to VASPs.

Acknowledgements

Part of the work included in this paper has been performed in the framework of the project "ANWIRE" (www.anwire.org), which is funded by the European Community under the contract IST-2001-38835.

References

1. UMTS Forum Report No. 9, "The UMTS third generation market - structuring the service revenues opportunities", available from http://www.umts-forum.org/.
2. Houssos, N. et al.: Advanced adaptability and profile management framework for the support of flexible mobile service provision. IEEE Wireless Communication Magazine, Vol. 10, No. 4, August 2003, pp. 52-61.
3. World Wide Web Consortium: RDF/XML Syntax Specification, available from http://www.w3c.org/TR/rdf-syntax-grammar/.
4. Dillinger M., Alonistioti N., Madani K., (Eds.): Software Defined Radio: Architectures, Systems and Functions. John Wiley & Sons, June 2003.
5. Alonistioti N., Houssos N., "The need for network reconfigurability", in [4].
6. Moerdijk A. J., Klostermann L.:Opening the Networks with Parlay/OSA: Standards and Aspects Behind the APIs. IEEE Network, May 2003.
7. Houssos N., Gazis V., Alonistioti A.: Application-Transparent Adaptation in Wireless Systems Beyond 3G. M-Business 2003, Vienna, Austria, 23-24 June 2003.
8. Koutsopoulou M., Kaloxylos A., Alonistioti A.: Charging, Accounting and Billing as a Sophisticated and Reconfigurable Discrete Service for next Generation Mobile Networks. Fall VTC2002, Vancouver, Canada, September 2002.
9. Fouial O., Fadel K. A., Demeure I.: Adaptive Service Provision in Mobile Computing Environments. IEEE MWCN 2002, Stockholm, Sweden, 9-11 September 2002.
10. Gamma E., Helm R., Johnson R., Vlissides J.: Design Patterns: Elements of Reusable Object Oriented Software. Addison Wesley Longman, Inc., 1994.
11. XML: Extensible Markup Language home page, http://www.w3.org/XML/.
12. RDF: Resource Description Framework home page, http://www.w3.org/RDF/.
13. RDF Semantics, http://www.w3.org/TR/rdf-mt/.
14. Butler M., "Barriers to the real world adoption of Semantic Web technologies", HP Technical Report, HPL-2002-333.
15. Extensible Markup Language (XML) 1.0, W3C Recommendation, 6 October 2000.
16. Decker S., et al., "The Semantic Web: the roles of XML and RDF", IEEE Internet Computing, September-October 2000.
17. RDF Model and Syntax Specification, W3C Recommendation, 22 February 1999.
18. Smith C., Butler M., "Validating CC/PP and UAProf Profiles", HP Technical Report, HPL-2002-268.
19. Open Mobile Alliance (OMA): User Agent Profile (UAProf) specification, available from http://www.openmobilealliance.org.