Metadata Design for Introspection-Capable Reconfigurable Systems

Vangelis Gazis, Nancy Alonistioti, and Lazaros Merakos

Communication Networks Laboratory, Department of Informatics & Telecommunications, University of Athens, 157 84, Athens, Greece,
{gazis, nancy, merakos}@di.uoa.gr

Abstract. Global vision consensus on the next generation of wireless mobile communications, broadly termed 4G, sketches a hybrid infrastructure, comprising different wireless access systems in a complementary manner and vested with reconfiguration capabilities that facilitate a flexible and dynamic adaptation of the wireless infrastructure to meet the ever-changing service requirements. We identify essential metadata classes to support the reconfiguration of communication systems, introducing a respective object-oriented UML model. We elaborate on the design rationale that underpins the UML model, describing its classes and associations and discussing the possible metadata representation technologies and encoding formats. We proceed to identify existing metadata standards that are candidate for the representation of reconfiguration metadata, discussing and evaluating their suitability. Ultimately, we present a developed reconfiguration metadata description vocabulary and illustrate its application with an example.

1 Introduction

Over the last decade, the mobile industry has developed into a breeding ground for innovative wireless access technologies. In addition to second (2G) and third (3G) generation mobile communication systems, broadband WLAN type systems such as HIPERLAN/2, IEEE 802.11 and broadcast systems like DAB and DVB-T are becoming available and short range connectivity systems like Bluetooth are being developed rapidly. Considering that the observed proliferation of wireless access technologies is likely to persist and that future mobile devices will need to support multiple dissimilar wireless access standards, the mobile communication industry has been focusing on the reconfigurability concept as a technological enabler of future (multi-standard) mobile systems and radio resource management across different wireless standards. The now widely accepted vision for reconfigurable systems and networks sketches a seamless ubiquitous computing and communication infrastructure where mobile and immobile devices may proactively and/or reactively adapt their own communication capabilities by dynamically discovering, selecting, downloading and activating software implementations for the communication personalities they wish to assume in any given time instance.

The rest of the paper proceeds as follows: The next section highlights the fundamental concepts of reconfiguration, introducing key definitions and providing an overview of current approaches as well as the related standardization status. Next we focus in the realm of each individual reconfigurable system and introduce a generic object-oriented UML model that facilitates discovery of reconfiguration options (i.e., the reconfiguration space) to support the application of reconfiguration within and across communication standards. We go on to elaborate on the design rationale of the UML model, followed by a discussion and brief evaluation of instrumentation options. Finally, we conclude the paper and highlight directions for future work.

2 Reconfiguration – Basic Definitions and Standardization Issues

In general, the term reconfiguration refers to the (dynamic) instantiation, parameterization and inter-connection of protocols (i.e., communication-related functional entities) within the user, control and management planes of a collection of operating communication systems in a manageable, consistency-preserving and – preferably – transparent fashion. For the rest of the paper, the term reconfiguration will refer to the dynamic adaptation of implementation mappings of internal (communication) equipment components [1] that does not compromise their consistency or their ability to provide their services. Leveraging the work of from early software radio projects in the military domain [2], SDR Forum has pioneered in exploring reconfiguration in the domain of wireless communications. However, being the vanguard of reconfiguration developments and the first to define a software radio architecture [3], seems to have come at the expense of a rather restricted view on reconfiguration that focuses primarily on the radio domain (e.g., RF processing, down-conversion, IF processing, A/D conversion, etc) [4]. Soon it was realized that, restricting the concept of reconfiguration solely to radio-dependent communication functionality under control of mobile network operators or equipment manufacturers only, would severely limit its application domain and undercut its beneficial impact on the long run. Support grew on the viewpoint that the full potential of reconfiguration would best be served by opening up reconfigurable device capabilities to the wider service provision process and leveraging technical expertise in third-parties (e.g., software developers) [5], creating an open market for (software) implementations of reconfigurable equipment components that will propel the development of universally reconfigurable mobile systems.

The (now joint) Parlay/OSA standardization initiative has been a major step forward towards the openness of the mobile value chain and the participation of multiple players in mobile service provision. However, it did not anticipate the case of reconfigurable systems; Parlay/OSA consider the network infrastructure as immutable and specify logical interfaces for invoking the particular functionality it supports. Although not precluded by their logical architecture, the case of reconfigurable wireless networks and mobile systems capable of dynamically adapting their internal instrumentation is beyond the scope of the current standard which does not include reconfiguration-supporting interfaces. This shortcoming has been identified in [6] along with the need for appropriate Parlay/OSA extensions to support third-party driven reconfiguration actions upon the mobile network infrastructure.

Another paramount issue not yet identified in the literature that should be addressed by standardization, and which is explicitly identified here, concerns the specification of an appropriate object model for reconfigurable communication systems. Architectures supporting reconfiguration will require a suitable object-oriented information model to capture and express the internal organization and structure of reconfigurable equipment in an abstract, implementation neutral way that effectively provides the unified view necessary to start specifying a generic reconfiguration capability. Through object orientation and inheritance, common parts can be factored out and reused as an abstract foundation model from which wholly different instrumentation inherit, thereby allowing a fine-grain mix of standardized behavior with innovative, performance-focused, proprietary instrumentations. Object orientation does not necessarily restrict the granularity of structural analysis to individual classes and objects; use of more coarse-grain analysis modules (i.e., components) is also possible (and to a large extent desirable).

3 Designing for a Generic Reconfiguration Capability

Reconfigurable systems must be adaptable at two different levels: the base level that includes the (software-based) instrumentations of communication-related functionality and the so-called meta-level comprising the (abstract) specifications of that functionality. That will allow development of architectures supporting adaptation between different (software-based) instrumentations of communication personality and across disparate communication personalities (e.g., ad-hoc, cellular, broadcast, etc) in a uniform way. From that viewpoint, generic support structures like architectural frameworks for flexibly expressing and circulating reconfiguration-related metadata become of paramount importance. The next section introduces a UML model designed to provide suitable metadata abstractions for the development of manageable reconfigurable communication systems in beyond 3G mobile networks.

4 Modeling Reconfiguration Metadata

4.1 Metadata Classes

Product, the root abstract class in our model, specifies a ‘marketable’ item (i.e., a resource that may constitute the subject of an exchange in an economic system), which can be identified through a textually represented name. It includes a single (URI-convertible) URL attribute that provides a unique identifier of each individual product instance as a Web-identifiable resource, thereby streamlining it to the Semantic Web model and its Resource Description Framework (RDF) [7].

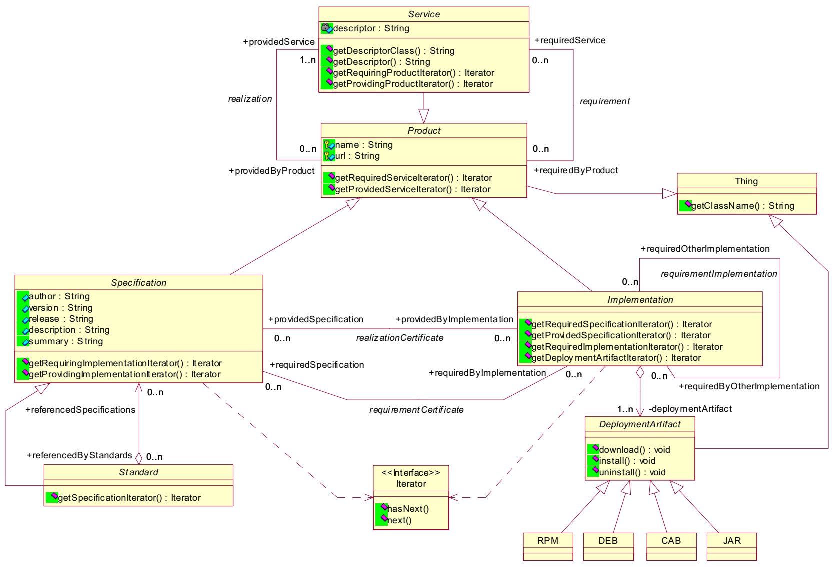
Service
descriptor : String
getDescriptorClass() : String
getDescriptor() : String
getRequiringProductIterator() : Iterator
getProvidingProductIterator() : Iterator
+providedService
+requiredService
realization
requirement
Product
name : String
url : String
getRequiredServiceIterator() : Iterator
getProvidedServiceIterator() : Iterator
+providedByProduct
+requiredByProduct
Thing
getClassName() : String
Specification
author : String
version : String
release : String
description : String
summary : String
getRequiringImplementationIterator() : Iterator
getProvidingImplementationIterator() : Iterator
+providedSpecification
+providedByImplementation
realizationCertificate
+requiredSpecification
+requiredByImplementation
requirementCertificate
Implementation
getRequiredSpecificationIterator() : Iterator
getProvidedSpecificationIterator() : Iterator
getRequiredImplementationIterator() : Iterator
getDeploymentArtifactIterator() : Iterator
+requiredOtherImplementation
requirementImplementation
+requiredByOtherImplementation
-deploymentArtifact
DeploymentArtifact
download() : void
install() : void
uninstall() : void
+referencedSpecifications
+referencedByStandards
Standard
getSpecificationIterator() : Iterator
<<Interface>>
Iterator
hasNext()
next()
RPM
DEB
CAB
JAR

Fig. 1. The object-oriented information model for reconfiguration metadata.

Service is a subclass of Product that refers to some precisely defined functionality and has a textual description property. It is meant to provide an abstract yet unambiguous placeholder for a service's definition accompanied by a textual descriptor that might be associated with arbitrary formal semantics, provided those semantics support a textual representation. It is not particularly important whether a unique formal format is employed for the service descriptor, since generic adaptation mechanisms may be used to identify the appropriate handler for each available format. However, it is of paramount importance that the service descriptor identifies the service unambiguously, an overlooked issue that is further elaborated on in the subsection entitled "Metadata encoding".

Specification is a subclass of Product with additional (textual) attributes, namely author, version, release, description and summary. Specification provides an abstract class for commonly representing behavioral and/or functional specifications (e.g., the specification of a authentication protocol). It is meant to provide a first-class abstraction for standards developed and published by authoritative bodies, such as the Universal Mobile Telecommunication System (UMTS) specifications developed and published by the 3rd Generation Partnership Project (3GPP). Currently, such specifications are recorded in a documentation system in various human-readable formats, such as the IETF Request For Comments (RFC) textual system. The lack of a common (machine-interpretable) format for specifications published by different authoritative bodies rules out the possibility of having those specifications parsed, understood and exploited by an intelligent agent in control of reconfigurable communication capabilities.

Standard is a subclass of specification designed to provide a generic container for related specification instances, in order to facilitate modeling of specifications that reference (as opposed to specialize) other specifications, possibly published by a different authoritative body (to the one that publishes the standard). The 3GPP specification of the IP Multimedia Subsystem (IMS) in UMTS is an example of a standard that leverages specifications developed by a different authoritative body (i.e., the IETF SIP specification). We stress that, through the Specification and Standard classes, inheritance-based as well as composition-based modeling of actual communication standards is supported, thereby rendering the full spectrum of modeling options available to the designer [8].

Implementation is a subclass of Product that refers to a real-life (software) artifact, which may realize multiple specifications. It is meant to model the real-life software instrumentation of a specification but may also be used to represent software-based functionality that is not associated to a particular specification (e.g., utility functionality). Given that an implementation may be developed in different programming languages and supporting technologies (e.g., C, C++, Java, .NET) and packaged in various deployment formats (e.g., Microsoft CAB, RedHat Linux RPM), modeling of implementations should provide unified support for different deployment artifacts through a common base class, such as the DeploymentArtifact abstract class included in Fig. 1.

4.2 Metadata Associations

A particular specification may depend on the availability of multiple services much as it may render multiple services. Similarly, a particular implementation, in addition to the set of services that its associated specifications collectively require and realize, may depend on the availability of additional services to function properly and may realize additional services during operation. Because they apply to Specification and Implementation instances alike, these concerns are expressed through a pair of associations between the Product and Service classes named requirement and realization, respectively.

Access to the aforementioned associations is supported based on an application of the Iterator design pattern [8] that abstracts the implementation details of the association from client entities. An agent may navigate these associations through an Iterator instance returned by any of the (getRequiredServiceIterator, getProvidedServiceIterator) and (getRequiringProductIterator, getProvidingPro-ductIterator) method pairs of the Product and Service classes, respectively, rendering client implementations dependent solely on the Iterator interface, while the supporting implementation of the navigation facility may vary arbitrarily from a local database row set to an hyperlinked knowledgebase distributed over the Internet or any suitable combination. Finally, an implementation may de dependent upon the availability of other implementations to function properly (e.g., object libraries), a concern expressed through the requirementImplementation association.

Regarding the relation between Specification and Implementation instances, we should note that it is not mandatory that an Implementation instance be associated to a

Specification instance; it might as well be an implementation of utility functionality not subject to standardization yet required by other implementations. Thus, the case of an Implementation unassociated to a Specification instance is considered valid. In the typical case, however, the association between a Specification and an Implementation is expressed via the realizationCertificate and requirementCertificate named (multilateral) associations. The former signifies that the Implementation instance realizes the behavior of the set of Specification instances, while the latter marks the dependence of the Implementation instance upon a set of Specification instances. Agents may navigate the realizationCertificate and requirementCertificate associations through an Iterator instance returned by any of the (getRequiredSpecificationIterator, getRequiringImplementationIterator) and (getProvidedSpecificationIterator, getProvidingImplementationIterator) method of the Implementation and Specification classes, respectively.

4.3 Metadata Encoding

The aforementioned UML model provides a common information model for expressing reconfiguration metadata that may be exploited by a reconfiguration management process. Considering that reconfiguration metadata may be subject to processing and exchange in different administrative domains, it should be represented in an instrumentation-independent format that ensures interoperability. Two recommendations of the World-Wide-Web Consortium, XML [9] and RDF [10] are considered as prime candidates for this task. In general, XML is easier to use and manipulate, while RDF has greater capabilities for expressing semantically rich information. However, only RDF is capable of unambiguous semantic representation, since there is an explicit unique interpretation of any RDF data, based on the RDF Model Theory [11]. Consequently, a certain piece of information can be represented in RDF in exactly one unique way, while in XML many different representations with the same meaning are possible [12]. This advantage of RDF comes at the cost of being more verbose and significantly more complex, making it less attractive for the vast majority of users and developers [13].

In our approach, all reconfiguration metadata are represented in RDF, while the vocabulary employed by the RDF representation is a combination of W3C-standard RDF vocabulary, industry-used vocabularies and an extension vocabulary defined in an RDF Schema document, all using XML as a serialization format. An extension vocabulary named RCM that is derived from an isomorphic mapping [14] of the aforementioned UML model to an RDF Schema document has been developed and used to describe the UML model classes and associations. To ease prototype implementation, we chose the widely used the Red Hat Package Manager (RPM) vocabulary [15], a superset of the Linux Standard Base Specification [16], for representing the metadata of the DeploymentArtifact class. Reconfiguration metadata are represented using the RCM extension vocabulary, which, thanks to the namespace extensibility mechanism of RDF, provides also for integration to the standard RDF and RPM vocabularies. The text below serves as an illustrative example of our RDF Schema applied for the case of the 3GPP GTP specification, which is dependent upon an ITU service identified via its RDF URI.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:RDF="http://www.w3.org/TR/WD-rdf-syntax#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:RPM="http://www.rpm.org/"
         xmlns:rcm="http://cnl.di.uoa.gr/People/Gazis/RCM/1.0/"
  <rcm:Specification rdf:about="GTP610">
            <rcm:Name>GTP</rcm:Name>
            <rcm:URL>
http://www.3gpp.org/ftp/Specs/archive/29_series/29.060/29060-610.zip"
            </rcm:URL>
            <rcm:Author>3GPP</rcm:Author>
            <rcm:Version>6.1.0</rcm:Version>
            <rcm:Release>6</rcm:Release>
            <rcm:Description>
GTP provides user data tunnelling within and across GPRS domains
            </rcm:Description>
            <rcm:Summary>
            GPRS Tunnelling Protocol (GTP) across the Gn & Gp interfaces specification
            </rcm:Summary>
            <rcm:Provides rdf:parseType="Collection">
              <rdf:Description
rdf:about="http://www.itu.ch/specifications/services/transport/reliable.rdf"/>
              <rdf:Description
rdf:about="http://www.itu.ch/specifications/services/general/tunnelling.rdf"/>
            </rcm:Provides>
            <rcm:Requires rdf:parseType="Collection">
              <rdf:Description
rdf:about="http://www.itu.ch/specifications/services/transport/unreliable.rdf"/>
            </rcm:Requires>
  </rcm:Specification>
</rdf:RDF>
```

The primary reason for preferring RDF over XML for metadata representation is that RDF has been specifically designed for unambiguous representation. Considering that RDF models can be serialized in XML, RDF provides an ideal instrument for unambiguously representing reconfiguration metadata whilst supporting their serialization into an interoperable, machine-interpretable textual format that can be widely circulated across different administrative domains without alteration of semantics. Naturally, the higher complexity associated with RDF is the price to pay for *semantic univocality* – although we feel that other significant benefits, such as seamless plug-in to the Semantic Web infrastructure and laying the foundation for a reconfiguration knowledge base upon which to build self-aware, cognitive communication systems, offset the cost in the long run.

5 Conclusions

In the forthcoming future, mobile communication devices will be vested with a cognitive introspective intelligence that monitors its operational context as well as its own instrumentation, adapting it whenever and wherever it deems necessary and in any way it sees fit through the dynamic download and assembly of software components into standard-compliant operating instrumentations. Availability of appropriate reconfiguration metadata is a prerequisite to the advent of introspective cognition capabilities and a facilitator of efficient reconfigurations, an issue that has not been at the focus of mobile communication research. Similarly, efficiency concerns dealing with the optimality of different metadata representation standards for the representation of reconfiguration metadata have met little attention in the literature. On the Parlay/OSA initiative front, reconfiguration is yet to be included in the standardization agenda and the issue of object-oriented models as reconfiguration enabling frameworks remains in research twilight. We address these issues by introducing a generic object-oriented model to express reconfiguration metadata that will enable future systems to evolve not just the instrumentation of their behavior but the behavior itself, thus facilitating reconfigurations across disparate network architectures (e.g., ad-hoc, cellular) and deployment topologies. In addition to design issues, we have discussed and evaluated the potential of existing technologies and related standards for representing and encoding reconfiguration metadata in a machine-interpretable format that can be circulated across different administrative domains without semantic losses. Future extensions of our work will focus on the development of appropriate algorithms to support service-driven reconfiguration of mobile communication devices, both for inter- and intra- standard scenarios in order to assess algorithm complexity and to conduct comparative performance evaluations.

References

1. Tang, Z.: Dynamic reconfiguration of component-based applications in Java, M.Sc. thesis, MIT, September 2000.
2. Cox, M. C.: Joint tactical radio system (JTRS), presentation available from http://www.jtrs.sarda.army.mil/.
3. Bickle, J.: Software radio architecture (SRA) 2.0 overview, OMG TC, December 11, 2000, Orlando, Florida.
4. Blust, S. M.: SDR definitions, SDR Forum Plenary & Technical Committee, September 1, 2000.
5. Pereira, J.: Beyond software radio, VTC Fall 1999, Amsterdam, Netherlands, September 22, 1999.
6. Alonistioti, A., Houssos, N., Panagiotakis, S.: A framework for reconfigurable provisioning of services in mobile networks, International Symposium on Communications Theory & Applications (ISCTA), Ambleside Cumbria UK (2001).
7. Manola, F., Miller, E.: RDF Primer, see http://www.w3.org/TR/rdf-primer/.
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object Oriented Software, Addison Wesley Longman (1995).
9. XML: Extensible Markup Language home page, see http://www.w3.org/XML/.
10. RDF: Resource Description Framework home page, see http://www.w3.org/RDF/.
11. Hayes, P.: RDF Semantics, see http://www.w3.org/TR/rdf-mt/.
12. Berners Lee, T.: Why RDF model is different from the XML model, W3C discussion note, see http://www.w3.org/DesignIssues/RDF-XML.html.
13. Butler, M.: Barriers to the real world adoption of Semantic Web technologies, HP Labs Technical Report, HPL-2002-333, see http://www.hp.com/.
14. Chang, W.: A discussion of the relationship between RDF-Schema and UML, W3C discussion note, see http://www.w3.org/TR/1998/NOTE-rdf-uml-19980804.
15. Red Hat Package Management format, see http://www.rpm.org/.
16. Linux Standard Base Specification, see http://www.linuxbase.org/.