Running head: UML Application Profiling

Introducing a UML Profile for Application Profiling in the e-Learning Domain

Abstract

In the e-Learning field open XML Schema standard metadata specifications are routinely established to promote interoperability. However they need to be specialized for facing the different user-communities specific exigencies. Thus the concept of an Application Profile, which specifies the modifications and extensions to one or more reference metadata schema, is being formalized. Several authors have recently proposed to represent XML metadata by means of the widely used Unified Modeling Language (UML), which provides a more intuitive and expressive description, and can then be automatically mapped to XML Schema. Following this trend, in this paper a first proposal is made to also adopt the UML for application profiling. Thus a UML profile for specifying Application Profiles is presented which can be automatically mapped to an XML Schema representation.

Introducing a UML Profile for Application Profiling in the e-Learning Domain

Interoperability, literally “the ability of a system or a product to work with other systems or products” (Miller, 2000), is today an all-pervasive buzzword in information technology, encompassing many different “flavours”. Beyond political, social and human interoperability issues (which are highly important, but behind the scope of the present paper), the technical dimensions of interoperability involve the capability of independently developed software components to exchange information, to interact –generally via published standard interfaces- in the accomplishment of a service, and, in the broadest sense, to be used together. On the end user’s side, interoperability implies the ability — *anywhere, anytime* is the current slogan— to discover, query and access resources drawn from different sources.

In an effort to boost (technical) interoperability, information professionals from different application domains have been actively defining and using *metadata* to document, capture and preserve information resources, and especially to share them on the net. The notion of metadata, or “data about data”, originally stemmed from the database domain, but since the mid 90’s the term has broadened to include any kind of standardised information about resources (including non-digital ones). Notably, more mature and “interoperability-aware” communities are undertaking extensive standardization activities to agree on common terminologies and formats for metadata, so to prevent the proliferation of incompatible notations. In fact, to ensure exchange of information across applications, and common interpretations between users, the solution is to impose a degree of control upon the terminology and encoding used by all involved parties. This includes in particular the need to agree on the *format and vocabulary of data exchanged*.

Indeed, open published standard specifications for metadata, developed in an “open and fair process” (Duval, 2004), bring in a wealth of beneficial effects in terms of interoperability and flexibility, on both the developer’ and the user‘s sides (Duval, 2004). One of the leading

domains where attention for interoperability is topical and lot of standardization initiatives have been ongoing for years is *e-Learning* (Hatala, Richards, Eap & Willms, 2004). Different kinds of organizations are involved in learning technology standardization. Notably, aside with accredited organizations like IEEE LTSC (LTCS, 2005), very active are a number of well-established consortia, like W3C (W3C, 2005), IMS (IMS, 2005) and ADL (ADL, 2005), collecting members representing any stakeholder of the global e-learning community. These consortia publish technical specifications (not only for storing and exchanging metadata, but also to represent contents structuring, learning activities, learners profiles, etc), produced through a self-ruled process to which all members can contribute. The intent is that these specifications be sufficiently general to meet the requirements of the largest number of users.

However, exactly because they need to fulfill the need of a wide range of users, the standard specifications often remain at a very abstract level, that may not adequately satisfy the needs of a set of users. Communities sharing objects within a specific application domain face then some contrasting needs: while remaining compliant with the standard specification, they need to restrict somehow their range to fit specific needs. Moreover, they often need to combine elements from more different specifications. Experience in the real world adoption of those published specifications has soon made evident a recurring process of *localization*, i.e., the specialization of one or more metadata specifications to fulfill the needs of a community of shared interests.

Indeed, since the very inception of standard schemas for metadata, developers from different sectors (for example, Cultural Heritage, Geographical information, Audio-visual, etc) have always been “mixing and matching” (Heery and Patel, 2000) metadata elements sets, adding elements to existing sets and redefining the semantics of existing elements in the context of specific applications. What is new is that recently this process has been explicitly recognized and formalized into the notion of an *Application Profile (AP)*. In short, APs are an assemblage

of elements from multiple domains combined in a compound schema to satisfy application-specific requirements. In more technical terms, Heery and Patel (2000) define APs as schemas which consist of data elements drawn from one or more *namespaces*, combined together by developers, and optimised for a particular application. A short introduction to APs is provided in the *Background* section.

The notion of an AP supposes, anyhow, that these communities recognize the advantage to re-use the effort and results collectively produced and have the aim to increase the potential interoperability between the metadata collection they describe and other communities' collections. In other words, an AP is an explicit declaration of how existing schemas, forming the so-called *base specification*, are being used: the base specification can consist of standards, specifications, as well as of other APs.

This paper focuses on how APs can be defined and coded. According to Hunter and Lagoze (2001), two alternative schema languages are today prevalently used for expressing APs: RDF (Resource Description Framework) Schema (Klyne and Carroll, 2004) and XML (eXtensible Markup Language) Schema (Fallside and Walmsley, 2004), (Thompson, Beech, Maloney & Mendelsohn, 2004), (Biron and Malhotra 2004). Each approach offers its own advantages, in particular RDF Schemas support rich semantic descriptions, while XML Schemas provide for explicit structural, cardinality and data typing constraints. XML (Bray, Paoli, Maler & Yergeau, 2004) is a notation produced by the World Wide Web Consortium (W3C) enabling the open interchange of data. Then, an XML Schema , also referred to as XML Schema Definition (XSD), is a language to describe the structure of an XML document, by defining its legal building blocks. After its elevation to official W3C recommendation in May 2001, XSD is rapidly becoming the prevailing format.

However, an XML Schema description already defines managed resources at a quite low level of abstraction. It is meant mainly for developers and already exposes implementation

specific details, by explicitly defining the rules that map the profile elements onto lower-level objects, with their types, attributes and relations. Therefore often, as is for instance done for IMS specifications (IMS, 2005), the XML binding comes accompanied with an *Information Model*, which provides a more abstract specification, also referred to as a "conceptual data schema". An Information Model can be defined in an informal way, for instance in tabular form and using natural language for a brief description of the elements' semantic.

With the growing adoption of XML Schema, the need is being recognized to develop more effective tools for designing structured XML data. In this direction, Routledge, Goodichild and Bird (2002) (Routledge, 2002) have pioneered work towards allowing developers of metadata to use the more intuitive and expressive UML notation, and then providing them with an automated translation to the XML schema. The Unified Modeling Language (UML, 2005) is in fact a general and widely used notation, with many supporting tools available for editing and analysis. IMS has recognized the popularity and power of the UML, and in fact it has recently moved to expressing its Information Model specifications in UML. The idea then is that the XSD binding can be automatically generated from a UML Information Model.

Following this trend, the contribution of this paper is the *first attempt to provide a UML based approach to application profiling*. Thus this paper provides the UML profile and guidelines for defining the Information Model of an AP in UML, from which a XML Schema binding can be automatically derived.

In the next section background notions of APs and of UML profiles are provided, along with a discussion of their relationships. Then the UML proposed profile for Application Profiling, which is the core contribution of this paper, is presented. Its application to a case study is illustrated. Conclusions are finally drawn.

Background

This section provides some necessary notions behind the profile proposed in the following of this paper for defining APs. To limit paper size, a basic knowledge of XML Schema concepts and elements, as well as of UML elements, is assumed (the web is a rich source of introductory material, for example, (Fallside and Walmsley, 2004) and (UML, 2005) respectively).

Application Profile

In recent years, strictly connected with the widespread exigencies of adapting diverse metadata schemas to the specific needs of a community, and also of unifying them into a single document, the concept of an Application Profile has emerged. Different definitions exist for this concept (Baker, Dekkers, Heery, Patel & Salokhe 2001), (Federal Geographic Data Committee, 1998), (DESIRE, 2000), (Duval, Hodgins, Sutton & Weibel, 2002), but all refer mainly to the adaptation, constraining and/or augmentation of one or more reference metadata schemas, so that they can better satisfy a community exigencies.

In fact, the process for defining an AP generally starts inside a community, who shares a common interest in modifying and/or refining available specifications, as well as in facilitating conformance testing for interoperability. More specifically the generally adopted procedure for defining APs includes the following steps (Application Profiles Workshop, 2004):

- Declare the referred Information Models, i.e., the set of specifications that will be used inside the AP, and the basic namespaces.
- Identify which elements from the namespaces will be used inside the community.
- Identify the default definitions or labels that must be overridden, so to reflect community needs. This may involve: the substitution of a vocabulary with a new one, or its extension, which is possible thanks to the extensions points[1] (Thompson et al., 2004) included in many available metadata schema.

- Include additional elements and/or fields in the referred specification. In particular, the generic structure of an AP should include: *modifications* (for example, cardinality, value type), *conditions* (logical expressions which determine if a modification can be applied), and *constraints* (dependencies between elements).
- Specify the language and its binding. As said in the Introduction, the technology generally adopted to this purpose (also inside the e-Learning domain) is XML.

The process proceeds then with the AP publication, ideally in a public registry, so that anyone from any organization can locate and refer to an authoritative version of it. Finally APs must be maintained and updated each time new exigencies arise inside the community who created them (Smythe, 2004).

The usage of APs brings along many benefits (Application Profiles Workshop, 2004), and most notably an increased level of interoperability because changes and extensions that can be made to the reference metadata schemas are established in advance and made explicit. This facilitates and supports the selection and reuse of elements inside existing profiles, rather than their development from scratch. Moreover by the adoption of specified bindings, APs make domain data exchange and services development for multiple communities easier, because it is only necessary to reconfigure specific fields.

Different global structures of APs can be defined. Without loose of generality, this paper explicitly considers that used inside the TELCERT project[2] (TELCERT, 2005) and currently under consideration by IMS, which represents a large community inside the e-Learning domain.

For IMS released specifications (IMS, 2005), an XML binding of the Information Model of the base specifications generally already exists[3] (also called a base schema). Therefore, this paper focuses expressly on the XML binding for an AP.

An AP can modify the base schema by (Dann, 2005):

- changing the cardinality of elements in the base schema, i.e., changing the *maxOccurs* and *minOccurs* attributes;
- modifying attributes occurrence/multiplicity and cardinality required, optional or prohibited, or even supplying a default value;
- substituting simple value types of attributes or elements with newly defined types or replacing them with fixed values;
- adding elements or attributes at locations where the base schema provides appropriate extension points.

Referring (Dann, 2005) for a more detailed description of technical details, an AP is encoded in a XML document with a root element, called <<schema_mod>>, which contains: modifications in a <<modifications>> element, definitions in a <<definitions>> element and the base schema in the baseSchema attribute of the <<schema_mod>> element.

It is important to notice that the base schema modifications can be either unconditional, or can depend on conditions which must be dynamically evaluated when a concrete instance document is considered.

UML Profile

The Unified Modeling Language (UML) (UML, 2005) is a multipurpose widespread notation suitable for a large number of concepts; yet, in some circumstances it can still be necessary to specialize or extend some basic elements of UML for expressing specific requirements or notions. Hence, since version 1.4 of UML, standard mechanisms for extending and specializing the UML metamodel with notions relative to specific domains or environments have been introduced, giving rise to the concept of a *UML Profile*. Nowadays UML Profiles receive growing attention (for example, UML Profile for CORBA (2005), (Ambler, 2005)) and the UML 2.0 (UML, 2005) defines the Profiling mechanisms as core UML Constructs.

In detail, a profile is defined as a collection of extensions mechanism, such as constraints, additional stereotypes and tagged values, for describing specific modelling constructs (UML, 2005). A profile is described in terms of UML standard metamodel elements, which refine, limit or extend the semantic of UML, but without changing the metamodel itself.

As described in detail in (EDOC 2005), the definition of a UML profile follows one or more of the steps reported below:

- *Identify (a subset of) the UML metamodel*: the subset includes only the model elements of UML standard whose definition and notation should be modified (i.e., refined, extended, etc). The semantics of all the other terms is intended to remain the standard one.
- *Specify "well-formedness rules"* for the identified subset of the UML metamodel. These are set of constraints which refine the definition of the metamodel elements, and are written in the UML's Object Constraint Language (OCL) (Warmer and Kleppe, 2003).
- *Specify "standard elements",* which are the (subset of) UML metamodel specification for describing the standard instances of UML stereotypes, tagged values or constraints.
- *Specify semantics:* it is possible to define "specialized semantics", formally through OCL well-formedness rules or informally through natural languages (for example text or tables), for refining or augmenting in a consistent way the constraints already included into the UML metamodel.
- *Specify common model elements* (i.e., instances of UML constructs), expressed in terms of the profile.

Thus UML Profiles add semantics, constraints and information to the existing metamodel in order to:

- provide a terminology adapted to a particular domain or platform;
- give a syntax to constructs used in the target domain for specific needs. In this case it is necessary to make explicit the classes which they depend from and define their UML syntax.

Relationship between Application Profiles and UML Profiles.

Even though UML Profiles and APs originate in two different contexts (as said, the former are used to extend or specialize a base UML meta-model, the latter to extend and combine base metadata schemas), they share the objective of adapting some already existing model to a specific purpose or domain.

In particular the following features can be matched:

1. the selection of a core subset of elements: from the base schema for AP and from UML metamodel for UML profile;
2. the addition of elements and/or fields to the source schema for AP vs. the creation of stereotypes from standard UML elements for UML Profiles;
3. the substitution of a vocabulary with new or extended terms for APs vs. the usage of new stereotypes, constraints and tagged values for UML profiles;
4. the description of semantics of derived schema for APs and the description of OCL constraints and extended semantics for UML Profiles.

Therefore, in consideration of the strong relationships and similarities between APs and UML Profiles, the latter concept is here adopted for deriving a UML representation of the former.

UML Mapping of XML Schema

Over the last years, with the wide industrial diffusion of UML and the growing adoption of XML Schema for data exchange, software developers have been facing a new technical issue: *representing XML Schema in UML*. Motivations for doing this mapping may be various: UML is a standard notation, UML diagrams can be read and understood even by not XML expert people, and they can be automatically exported into XML Metadata Interchange (XMI) or translated into code or in XML Schema. But the prevailing fact is that UML augments considerably the readability and modifiability of documents with respect to XML Schema representations.

Diverse solutions have been proposed, each one focused in specific transformation patterns, that could or could not fulfill completely all the XML Schema concepts. Scrupulous attention has been put in providing semantically equivalent solutions which could support a bijective mapping between XML Schema and UML design. To this purpose a good reference is the work of Bernauer, Kappel, and Kramler, (2003), (2004), which offers a comprehensive comparison of the existing UML transformation patterns.

The pioneering work of Booch, Christerson, Fuchs, and Koistinen (1999) provided the first UML modeling extensions for elements, attributes, model groups, and enumerations with respect to a predecessor of XML Schema. Based on this work, Carlson (2001) provided a UML profile using XML Metadata Interchange (XMI) (XMI, 2005) rules for XML Schema transformation, which has been implemented in the commercial tool “hypermodel” (HyperModel tool, n.d.). However the profile did not include the mapping of simple content complex types, global elements and attributes, and identity constraints. Carlson’s work has been widely referenced, and several authors tried to overcome some of the intrinsic limitations of the profile, sometimes even introducing some constructs not UML compliant. Among them, Provost (2002) focused mainly on representation of enumerations and other restriction constraints, and Eckstein and Eckstein (2004) improved the modeling of simple types and notations.

Routledge et al. (2002) put the attention on a different problem: separating the conceptual level of the model from the logical level, mainly for improving the implementation process of the XML Schema. They were the first to propose a three-level design in which a UML profile is defined for the logical schema, although they used some representations not fully UML compliant (Routledge et al., 2002), (Routledge, 2002). This work still represents a fundamental basis for new proposals to represent XML schemas by UML profile. Among them, it is important to mention: the work of Bernauer et al. (2003, 2004) which improved and extended the UML profiles for XML Schema, with the representation of model groups, global elements

and attributes, identity constraints and simple types in a more concise way and in compliance with UML semantics, the works of Jeckle (2001) and Krumbein and Kudrass, (2003), and the series of articles of Marchal (2004, March, May, June, August) which define a UML metamodel for the XML Schema and present a technique, based on the eXtensible Style-Sheet Language (XSLT, 2005), for engineering the mapping between UML and XML Schema.

UML-based Application Profiling

The above presented works deal with how the UML can be used to represent XML schemas. This paper for the first time addresses the related problem of representing in UML an AP, i.e., a schema derived from one or more basic XML Schemas. In this section the developed UML profile, i.e., the UML elements and extension mechanisms identified as necessary to represent all the aspects of a hypothetical AP, is presented. An example of how this UML profile can then be used to specify a given AP is illustrated in the next section.

As already mentioned, it is important to ensure the isomorphic capability to go back and forth between the UML and the XML Schema representations. In particular, a prototype tool which automatically translates the UML specification into an XML Schema has been developed for evaluation purposes and is later outlined.

Trivially an AP which requires no modification/extension to the base specification represents the specification itself. Therefore, the UML profile here developed for representing APs can be used straightforwardly to also represent Information Models (Base Schemas).

The UML Profile for Application Profiling

The proposed profile builds on the works of Carlson (2001) and Routledge (2002) previously outlined, and extends their UML model by including those aspects of XML Schema specifically needed for representing APs. For simplicity this section only contains the description of the

stereotypes specific to the AP. All the stereotypes used for representing the XML Schema constructs can be found in Bertolino (2004).

Taking as a reference the AP description available at the *Application Profiles Documentation and XML files* (2004) web site, the Information Model is identified with the <<baseSchema>> stereotype, while the XML document structure of the AP is identified with a <<schema_mod>> stereotype. Note that these stereotypes, which represent packages, do not add semantic information to the standard UML package model, but they only identify the set of classes which define the base schema or the structure of the AP, respectively.

Directly related to the <<schema_mod>> and <<baseSchema>> stereotyped packages, the <<namespace>> class is then introduced, to identify the original schema (or schemas) from which modifications are applied. In particular (see Figure 1), the <<namespace>> classes are used to indicate the origin (*importNamespace*) and target (*targetNamespace*) of the elements and data types used in the schema. The <<namespace>> class has two fixed attributes: *name* and *schemaLocation* representing, respectively, the name of the namespace (target namespace or import namespace) and its location[4].

'schema_mod' is also the name of the <<root element>> stereotyped class which is a mandatory element in each UML representation of AP since it is the starting point of the UML description (Figure 2). Once the root element of the schema is defined, the link between the Information Model and the XML Structure of AP is highlighted in the *schema_modType* as reported in Figure 2.

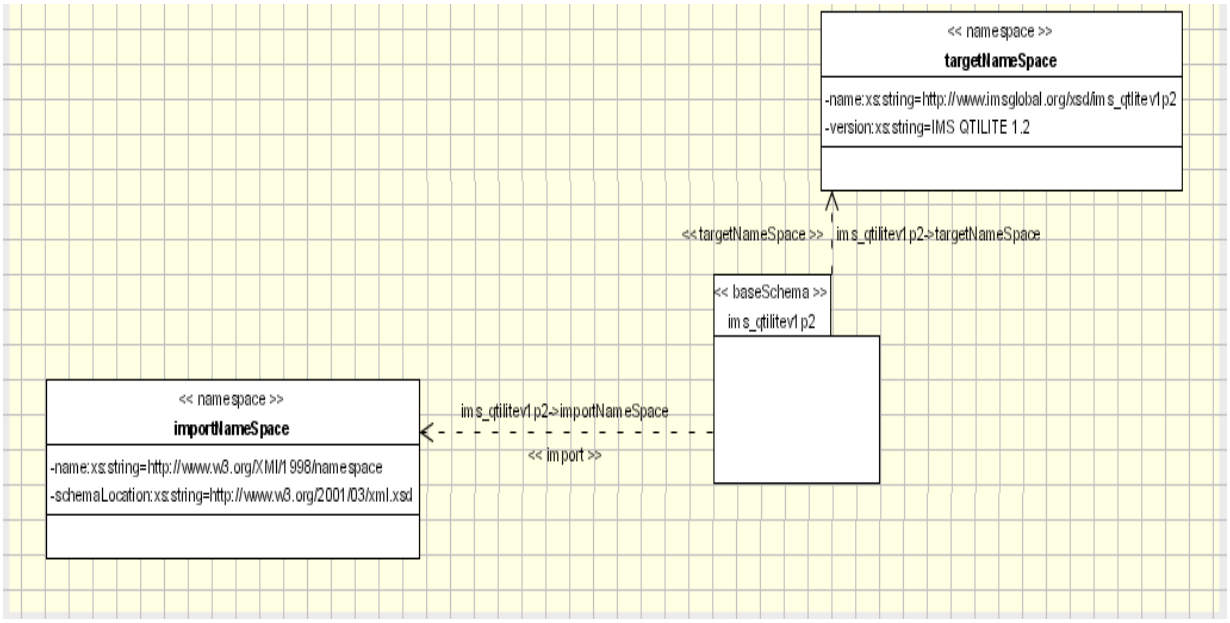

Figure 1 Application Profiling Schema Information Class Diagram

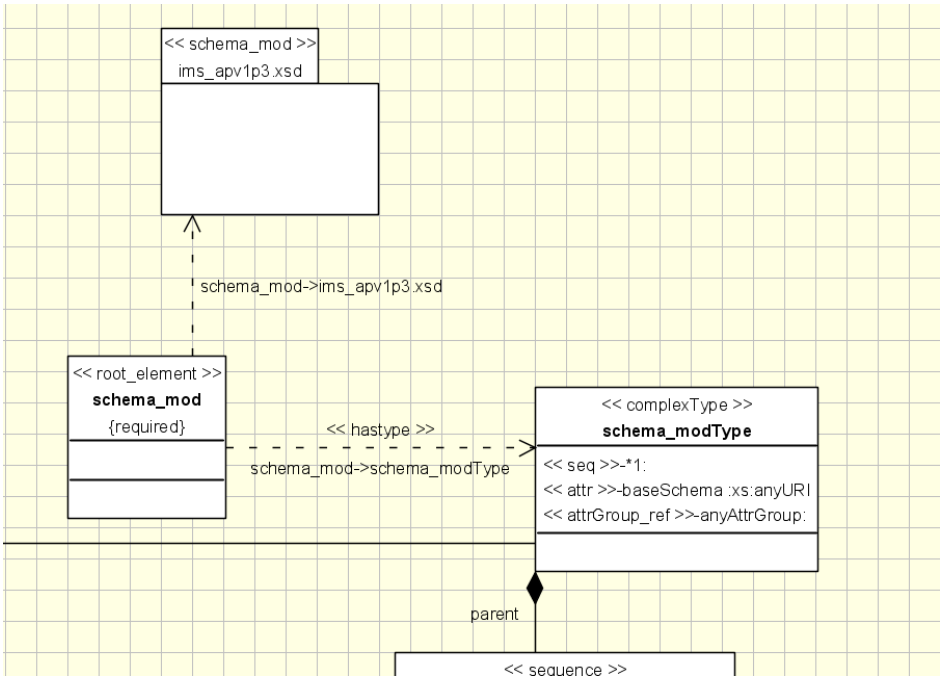

Figure 2 <<schema_mod>> class

Figure 3 highlights some specific features of the UML profile for XML Schema which are novel with respect to the work of Carlson (2001) and Routledge (2002). These aspects involve in particular the definition of:

- *Default Content Model:* the default content model for a complexType is 'sequence'. Therefore, when a complexType is mapped from UML to XML Schema the "«elt»"

attributes within the ‘complexType’ class are automatically mapped to a sequence of elements within the physical complexType reflecting the order of UML attributes. The adoption of a sequence as a default content model for complexType allows for simply representing the ordering of elements by just using a stereotyped <<complexType>> UML class. This definition is motivated by the fact that a sequence indicator inside an XML Schema specifies that its elements must appear in the exact order in which they are listed. If elements are included in other types of complex types (such as, for example, a choice content model), this needs to be explicitly modeled.

- *Nesting XML Schema Content Models* (e.g., a ‘choice’ nested inside a ‘sequence’): the concept of nesting is represented at the logical level by introducing separate stereotyped classes, for each nesting level, linked by a ‘composition’ association named “parent”. The direction of the composition association indicates the direction of the nesting, and the ordering of the attributes within each class indicates the ordering of the content models.
- *Global and local elements:* XML Schema elements are represented in two different ways according to their visibility in the document: global elements are represented as stereotyped classes, and have a stereotyped dependency relationship (“has type”) with their respective base types; local elements are instead included as UML attributes within the class representing the parent complex type or group. In both cases, by default elements are included in a sequence content model.
- *Anonymous types and nested content models description:* in the XML Schema, these are represented in the same way as nesting described above, with an additional naming scheme which ensures uniqueness and the preservation of ordering. In particular, anonymous types and content models are named by appending a sequential number (indicating order) to an asterisk (indicating an anonymous reference). On the other hand, when types are explicit

they are inserted in the UML model element and translated in the XML Schema with type = "typename".

Thus each element of the general schema of an AP, i.e., modifications, definitions and mappings (see the *Application Profile* section), is represented with a specific complex type containing all the information and attributes necessary for its XML binding. Figure 3 shows the UML profile representation of the main application profiling structure information together with two non-mandatory elements (*annotation* and *anyGroup*):

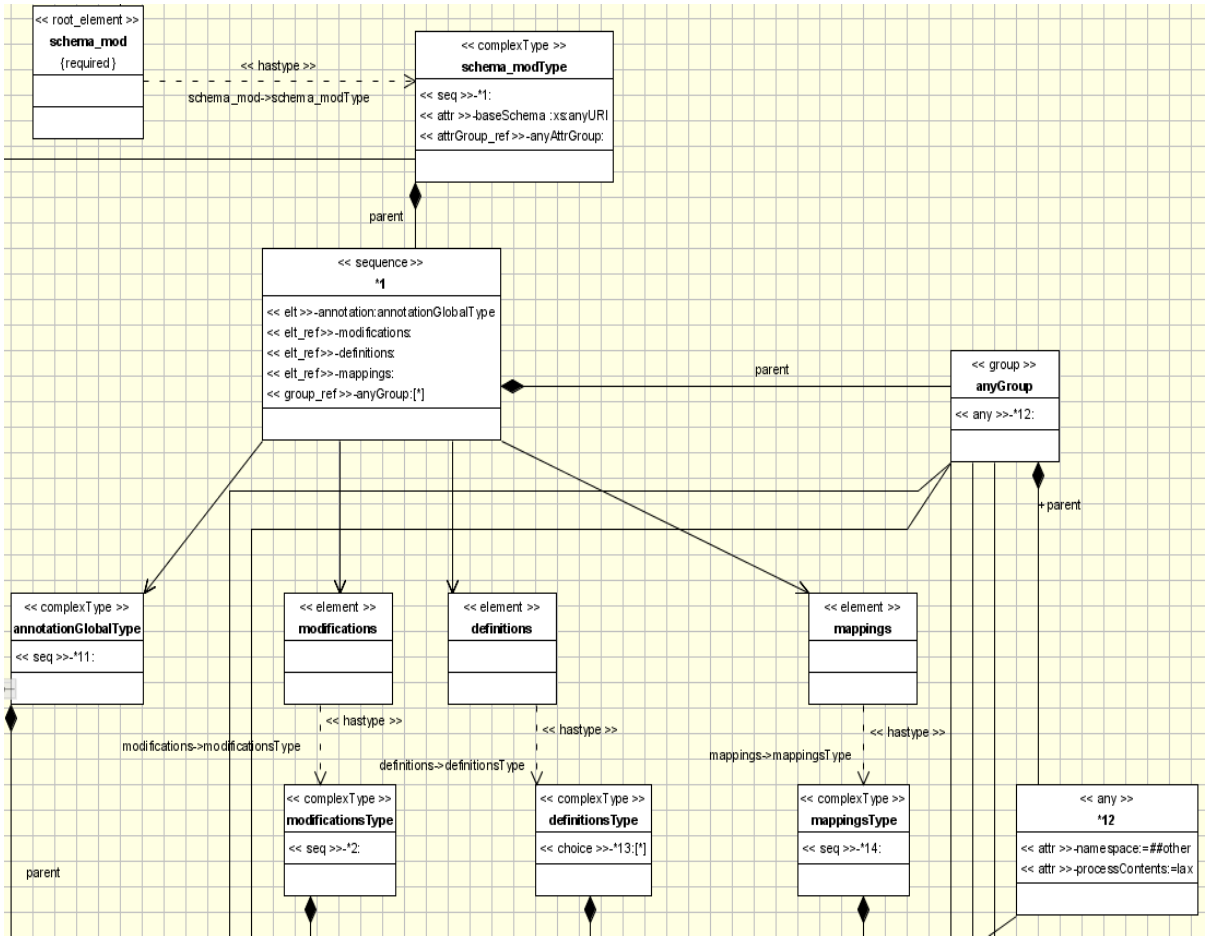

Figure 3 Global structure of Schema_mod

For aim of completeness, Figure 4 and Figure 5 report, once and for all, the corresponding XML Schema structure and XML Schema code.

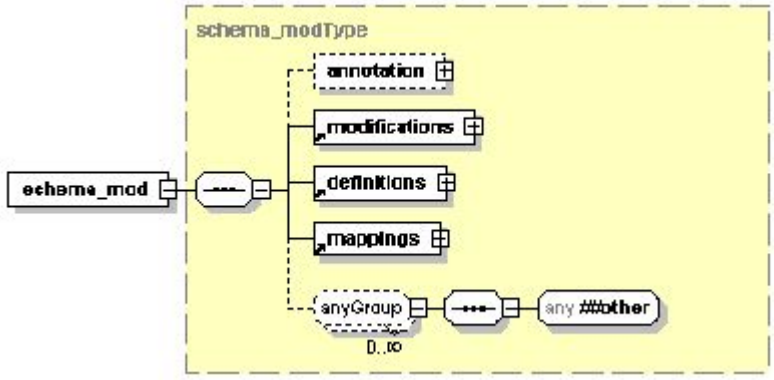

Figure 4 XML Schema structure of schema_mod

```
<!-- filename=ims_apv1p3.xsd -->
<xs:schema targetNamespace="http://iwm.uni-koblenz.de/xsd/ims_apv1p3" xmlns="http://iwm.uni-
koblenz.de/xsd/ims_apv1p3" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" version="IMS APPLICATION PROFILING 1.3">
<!-- ************************** -->
<!-- ** Inclusions and Imports ** -->
<!-- ************************** -->
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
<!-- ***************** -->
<!-- ** Root Element ** -->
<!-- ***************** -->
<xs:element name="schema_mod" type="schema_modType"/>
      <!-- **************** -->
<!-- ** schema_mod ** -->
          <!-- **************** -->
          <xs:complexType name="schema_modType">
             <xs:sequence>
                <xs:element name="annotation" type="annotationGlobalType"  minOccurs="0"/>
                <xs:element ref="modifications"/>
                <xs:element ref="definitions"/>
                <xs:element ref="mappings"/>
                <xs:group ref="anyGroup" minOccurs="0" maxOccurs="unbounded"/>
             </xs:sequence>
             <xs:attribute name="baseSchema" type="xs:anyURI" use="required"/>
             <xs:attributeGroup ref="anyAttrGroup"/>
          </xs:complexType>
</xs:schema>
```

Figure 5 XML code for schema_mod structure[5]

The *ModificationType* class, relative to the modeling of 'modifications', is the core element of UML Application Profiling metamodel. It collects all the generic information used within a modification and, following the guidelines of Dann (2005), (*Application profiles Documentation and XML files,* 2004), it includes all the types of modifications applicable to an AP. ModificationType is defined as a *UML abstract class* since it includes all attributes common to

modifications (such as *annotation*, the *base schema* it derives and the *element* to be modified), but does not represent itself a modification. A UML abstract class describes common attributes (and behaviors) for a set of classes without having any own objects.

An ad hoc representation, reported in Figure 6, has been created for representing the different types of modifications that the derived class can assume and to simply translate these possible cases to an XML representation. Note that ModificationType is the only class of the presented UML profile in which the transformation rules for sequence are not valid. New stereotypes, <<seq_ann>>, <<seq_choice>>, <<seq_group>>, are instead introduced to represent, respectively, <<annotation>>, <<choice>> and <<group>> elements inside a same sequence (see Bertolino (2004)). In particular, the 'seq_choice' generalization association represents the XML choice of all the possible types of modifications (i.e. *cardinality*, *attributes_properties*, *attribute_extension*, *element_extension* and *modification*); the 2.2 associated number is used to represent the order in which the modification should appear in the XML Schema representation, while the prefix *seq* is used (only in this case) to represent all the elements of the same sequence showing modifications.

To be able to maintain the compliance between AP and the associated Information Model, only some modifications are allowed (Dann, 2005), i.e., attribute properties modifications, cardinality modifications, modification restriction, attribute extensions and element extensions. For simplicity only the *attributes properties modification* is analyzed here; it defines the attributes which in turn refer the attribute or text node that must be changed in the base schema:

- a *use* attribute is used for referring the attribute to be changed;
- the *type* attribute is used for referring the datatype of the attribute;
- *fixed, default* and *mapping* attributes are used for defining all the modified attribute element characteristics.

Moreover attributes properties modification type extends, as all the other modifications type, the basic type of modification named baseModification. Left site of Figure 7 shows the UML description of Attributes Properties Modification type.

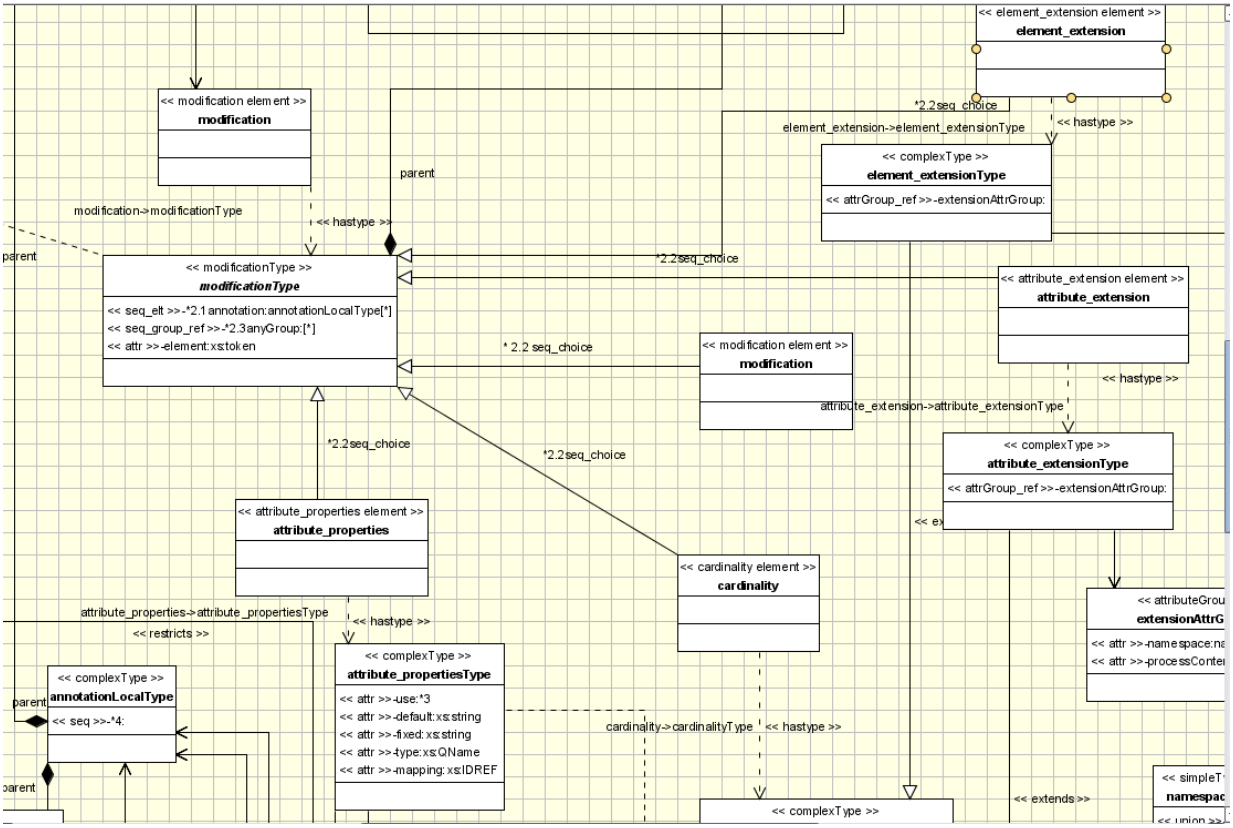

Figure 6 Example of the modificationType class UML modeling

The baseModification type represents the basic type of modification from which all the other modifications derive. It includes, in its attributes, the reference to the subelement, *subelement*, the modification referred to and, if necessary, the condition, *cnd* attribute, under which the modification is applied. The baseModification stereotyped class of Figure 7 is an example of the usage of sequence complex type as default content model. Right site of Figure 7 shows other types of modifications as cardinality and attribute extension modification.

Other relevant rules for the UML profile include:

- Attributes default values: when attributes have default (fixed, etc) values, the UML mechanism of tagged values is used to introduce additional information, (for example: {use= “required”}[6]).

- MinOccur and MaxOccur: Attribute multiplicities reflect the ‘minOccurs’ and ‘maxOccurs’ tags when mapped to the schema.

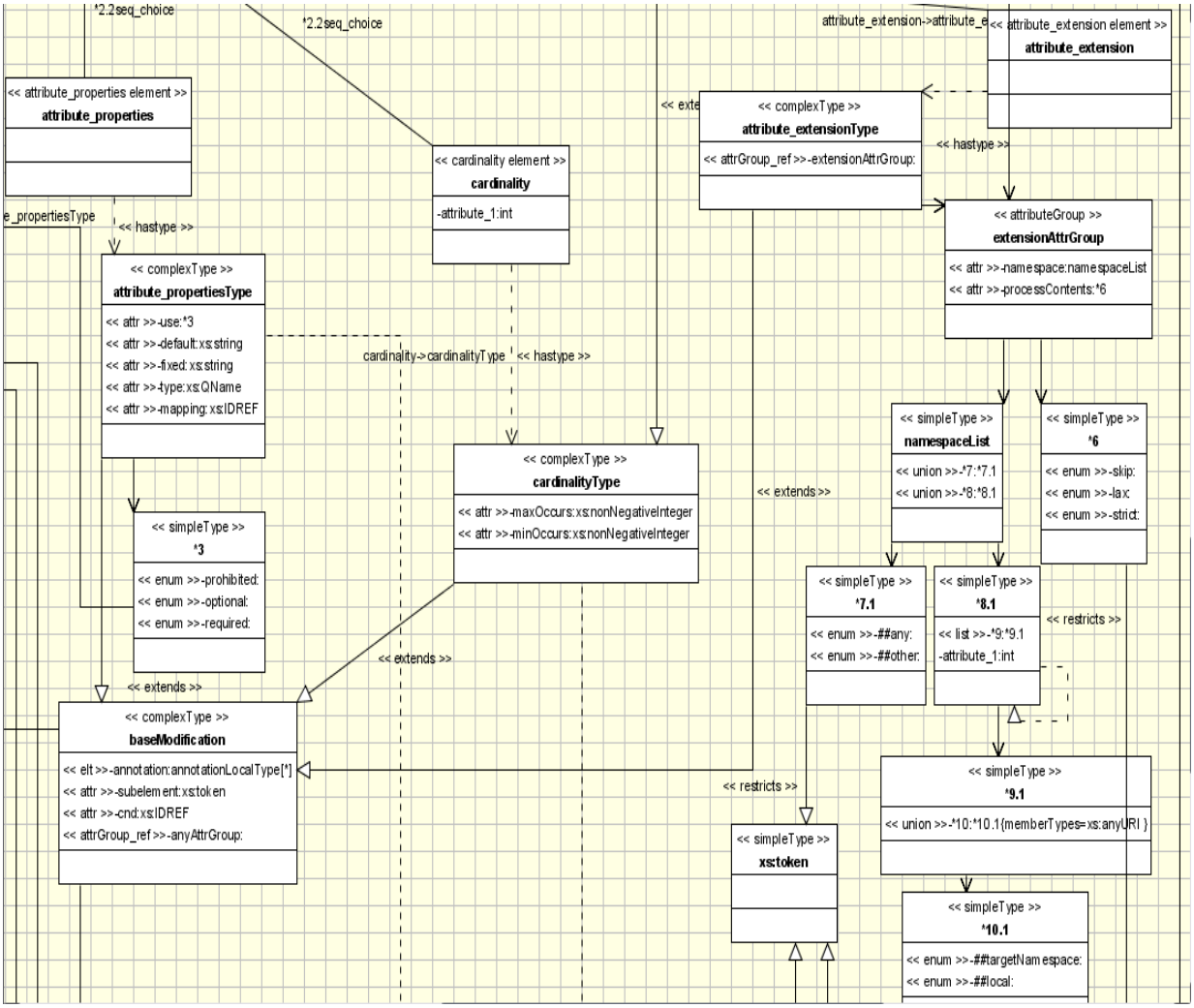

Figure 7 Attributes properties, cardinality, and attribute extension Modification

The remaining elements of a schema modification for APs include (see Bertolino, (2004) for details) mandatory elements, such as Definitions and Mappings, and non-mandatory elements, such as Annotations, for which the UML description of anyGroup element is included by default. In particular:

Definitions element represents definitions bound by some condition(s) and requires in particular the definition of two classes: *anyAttrGroup* and *any*.

Mappings element provides means to relate values of a simple type in the Information Model to values in another simple type in the profile. This can be used, for example, to translate a standard vocabulary of the original schema into the vocabulary of the community

Annotation element is a piece of human-readable information in a profile. Annotation type can be useful to clarify the meaning of an item and to provide further information.

In conclusion, an overview of the UML profile proposed is shown in Figure 8 below.

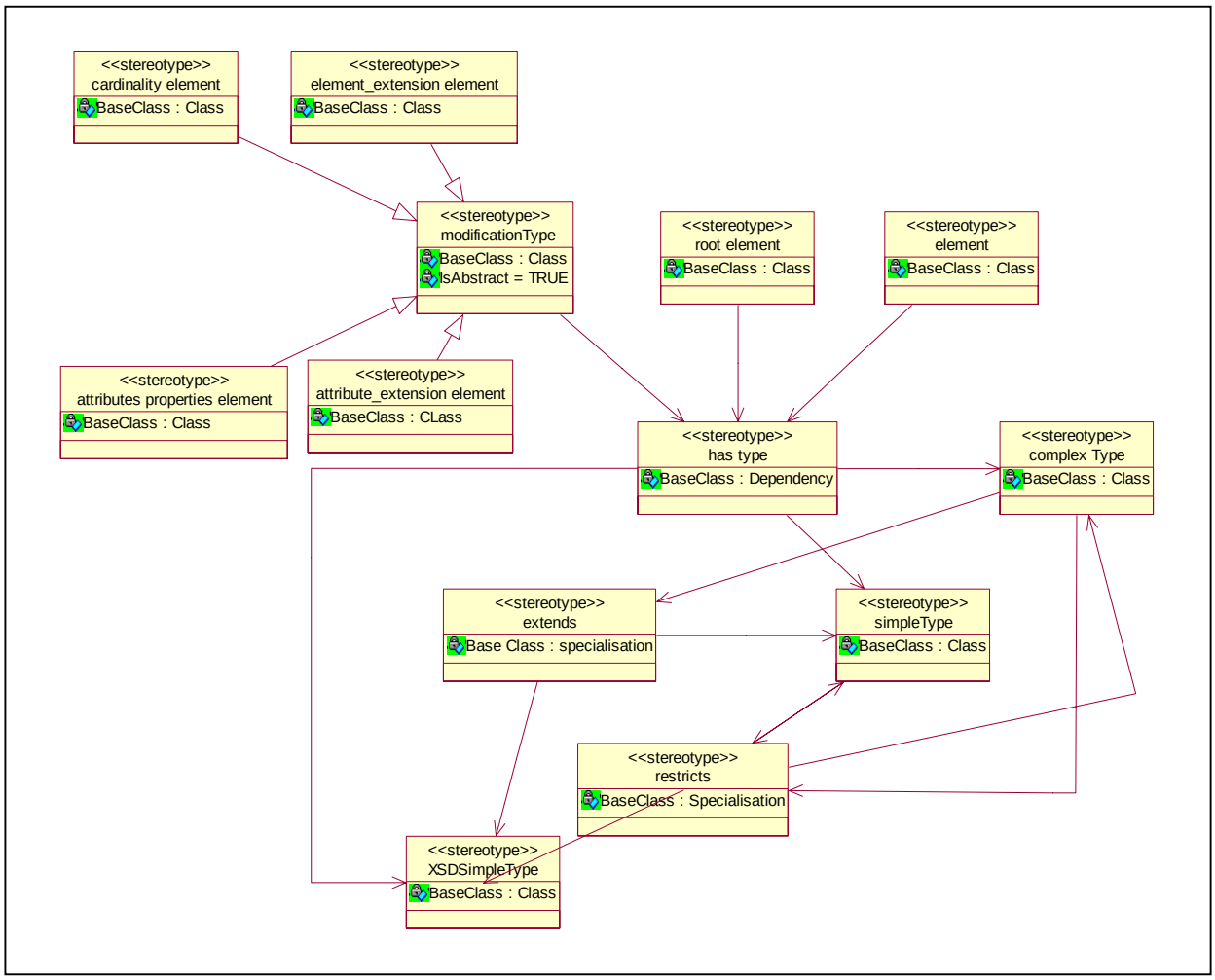

Figure 8 UML Profile for Telcert UML Application Profiling

Deriving XML Schema from UML models

Using as a reference model the Information Model class and applying the required modifications as indicated by the introduced UML profile, a specified AP can be derived. In practice the AP mainly corresponds to the same Class Diagram of the Information model, where

some classes or attributes are changed according to the modifications rules and the application customization adopted. It is possible to model an instance of AP by instantiating the UML Application Profile Class Diagram into an Object Diagram.

The proposed profile has been conceived so that from the Class Diagrams of Information Model and Application Profile an automatic binding to XML Schema can be obtained. The adoption of an automatic mapping mechanism represents an interesting facility for translating the UML model into a XML Schema, which can be generalized and specialized for any purpose and for different environments. Any possible modification in fact can be introduced and implemented at UML level, avoiding the complexity of managing huge number of XML lines of code.

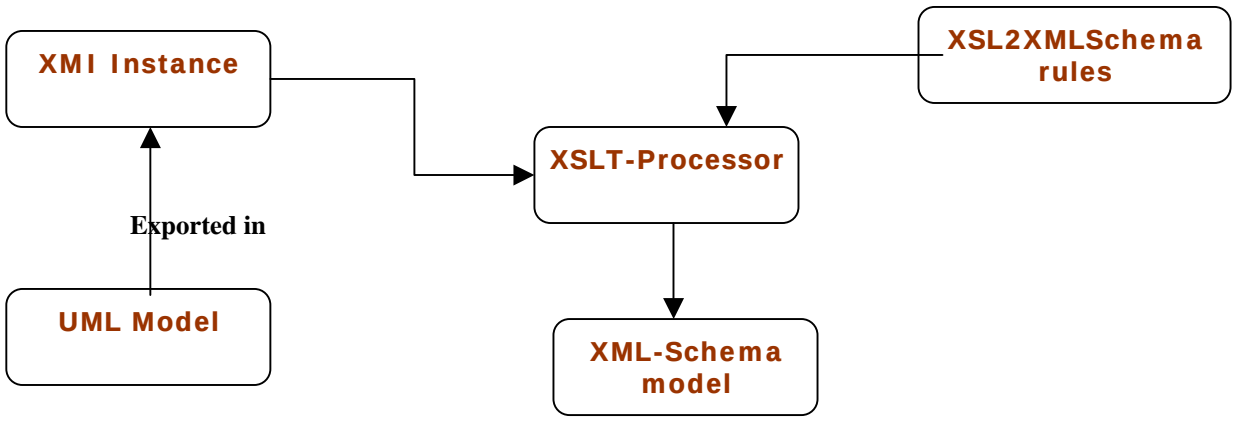

Figure 9 U2XAptly tool structure

A prototype tool, called U2XAptly (**U**ML **to** **X**ML-Schema **A**pplication **P**rofile **T**rans**L**ation with st**Y**le sheet) (Bertolino, 2004), has been developed for demonstration purposes. As shown in Figure 9, this tool manipulates the XMI exported model by means of Extensible Stylesheet Language (XSLT) (XSLT,2005) for deriving the XML Schema model. More precisely, taking in input the UML-based AP, exported into the XMI 1.2 format by the Poseidon UML case tool, U2XAptly maps the XMI model into the XML Schema model via a XSLT processor. Briefly the XML Schema is realized by transforming the UML class into a <*xs:element*>, and each attribute into a xs:element's *sequence*, *union* or *group*. The obtained XML Schema model is then

visualized. Figure 10 resumes the main details of this mapping process, further information are in Bertolino (2004).

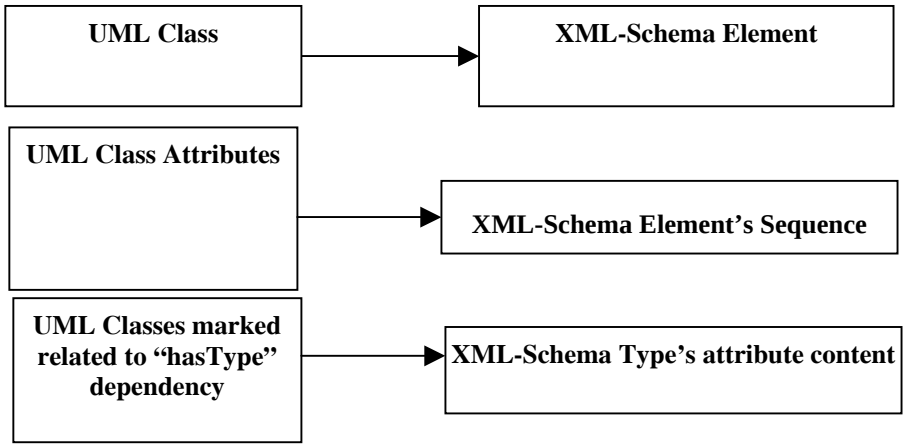

Figure 10 Simple mapping mechanism

Case Study

The UML profile for APs described in the previous section has been used for deriving an example AP in which a set of simple modifications are specified with respect to the IMS Question & Test Interoperability QTILite (QTILite 2002) Information Model.

Very briefly, IMS Question & Test Interoperability (QTI, 2004) specification has been designed for facilitating interoperability between different e-learning systems, and describes a datamodel for representing questions and test data and their corresponding results. QTILite is a subset of the QTI, which does not support all the features of the full QTI. However it provides a well-documented content format and a valid development support with enable also the report of test results. From a practical point of view the QTILite (as the QTI) is a datamodel for representing both the question and test data so information exchange between content providers, content management tools, assessment delivery systems and learning system is enabled (QTI, 2004).

The IMS does not provide a UML specification of the Base Schema, but only its XML Schema description. Thus, on the basis of the developed UML Profile, two UML Class diagrams

representing the QTILite Information Model (Base Schema) and the AP, respectively, were derived. In particular the UML representation of the QTILite Base Schema, provided a detailed Class Diagram that could be used as a reference basis for developing APs. Referring to this Class Diagram, the derivation of the UML AP representation can be done quite easily. If a UML Base Schema specification already existed, AP specification would only require to identify the specific changes/extensions.

Henceforth, this section focuses only on the description of the hypothetical AP, on those parts in which some illustrative modifications to the Base Schema are required. Simple modifications were introduced into the original QTILite Base Schema, as an example of a possible AP. In particular these involved three simple amendments related with attribute properties cardinality and definition modification. Each of these elements was represented with a specific complex type containing all the information and attributes necessary for application profiling.

The starting class of the Class Diagram representing the considered AP is labelled by the sterotype <<baseSchema >>, as described in previous sections (Figure 11).

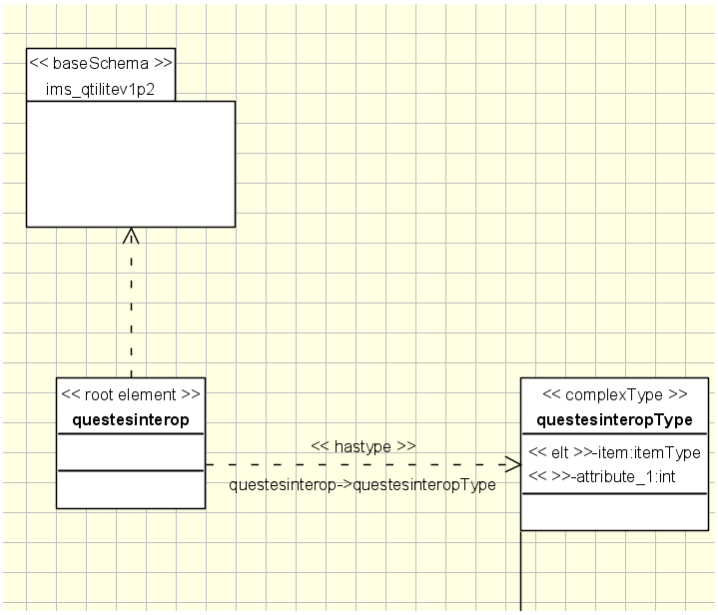

Figure 11 Starting classes of Application Profile

For clarity the figures below report the part of the AP involved in the modifications. In this figures the marked classes are those in which the modifications have been implemented.

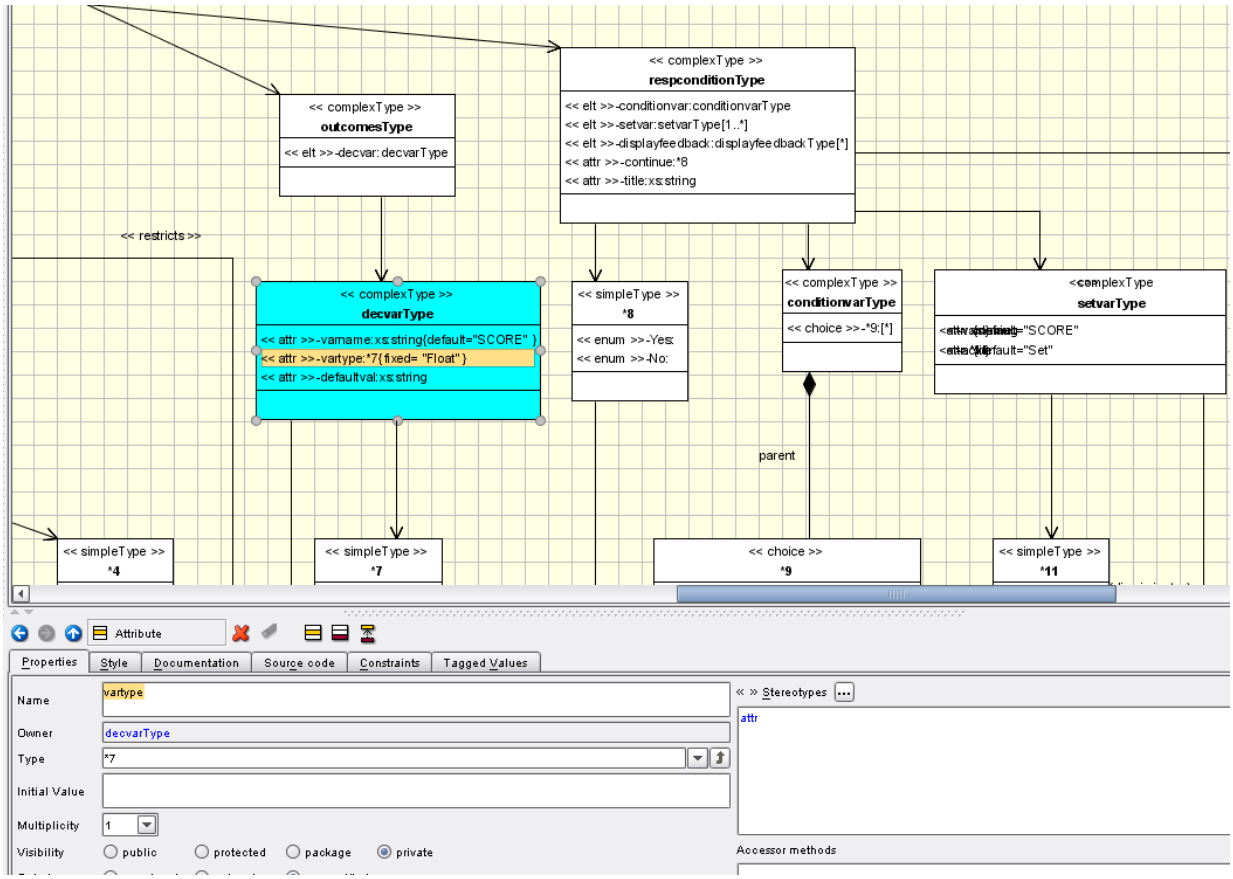

Figure 12 Attribute properties modification

In detail they show the usage of:

- Attribute properties modification>: the attribute "vartype" of the complexType "decVarType" has been considered and its default type "Integer" chanced into to the fixed type "Float (see Figure 12).

- Cardinality modification: the element *objectives* in the complex type *itemType* have been considered and its cardinality changed from minOccur=0, maxOccur=unbounded to minOccur=1 and maxOccur=1 (see Figure 13). Moreover, for the *objectiveType* of the *objectives* element, the *material* cardinality has been changed from maxOccur= unbounded to minOccur=1 and MaxOccurr=1 (see Figure 14).

- Definition modifications: a new type with respect to base schema has been introduced in the definition of a *modifiedVarType* (see Figure 15).

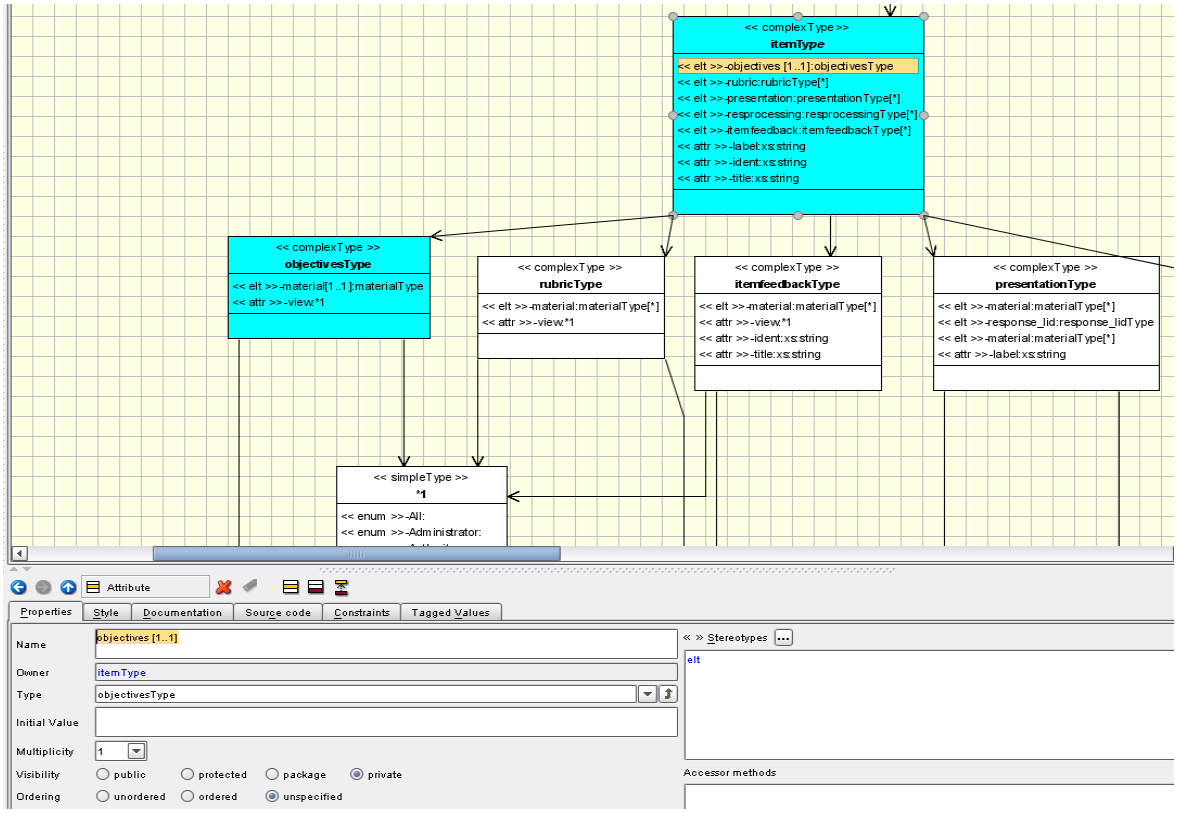

Figure 13 Cardinality modification element objectives

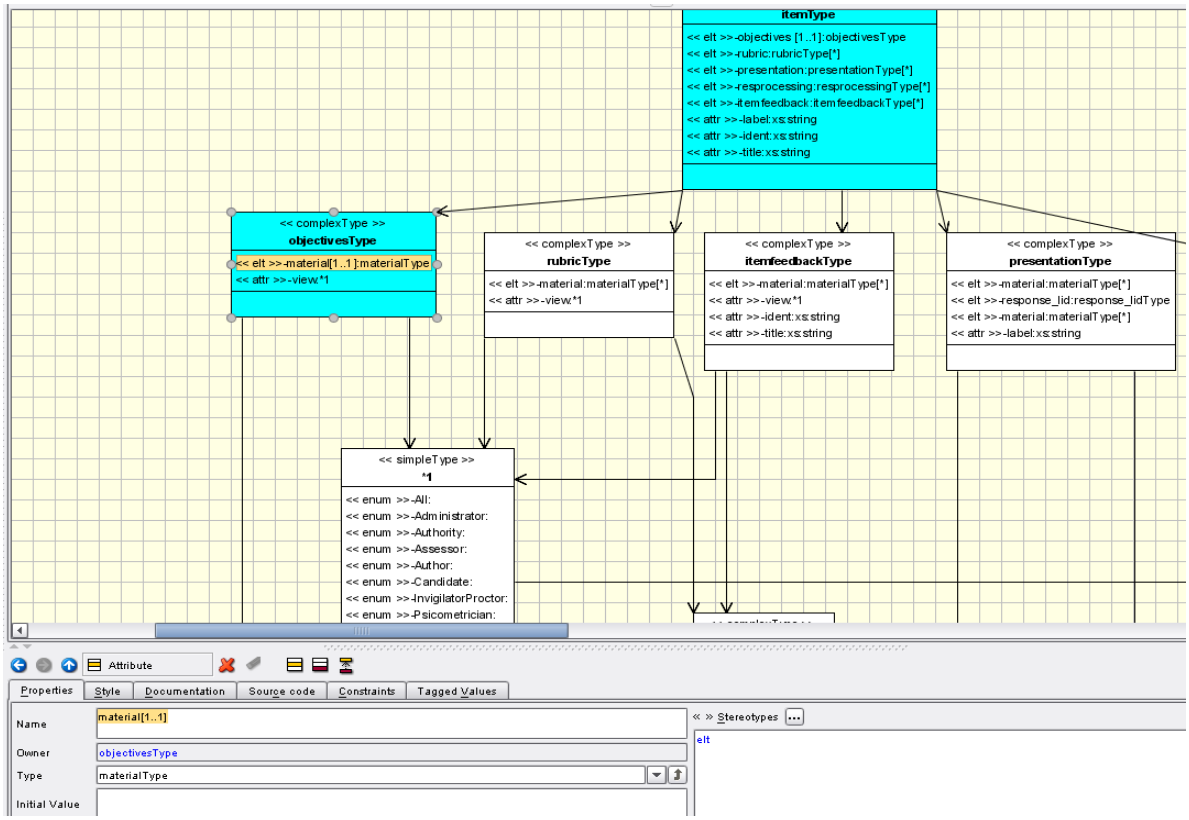

Figure 14 Cardinality modification material cardinality

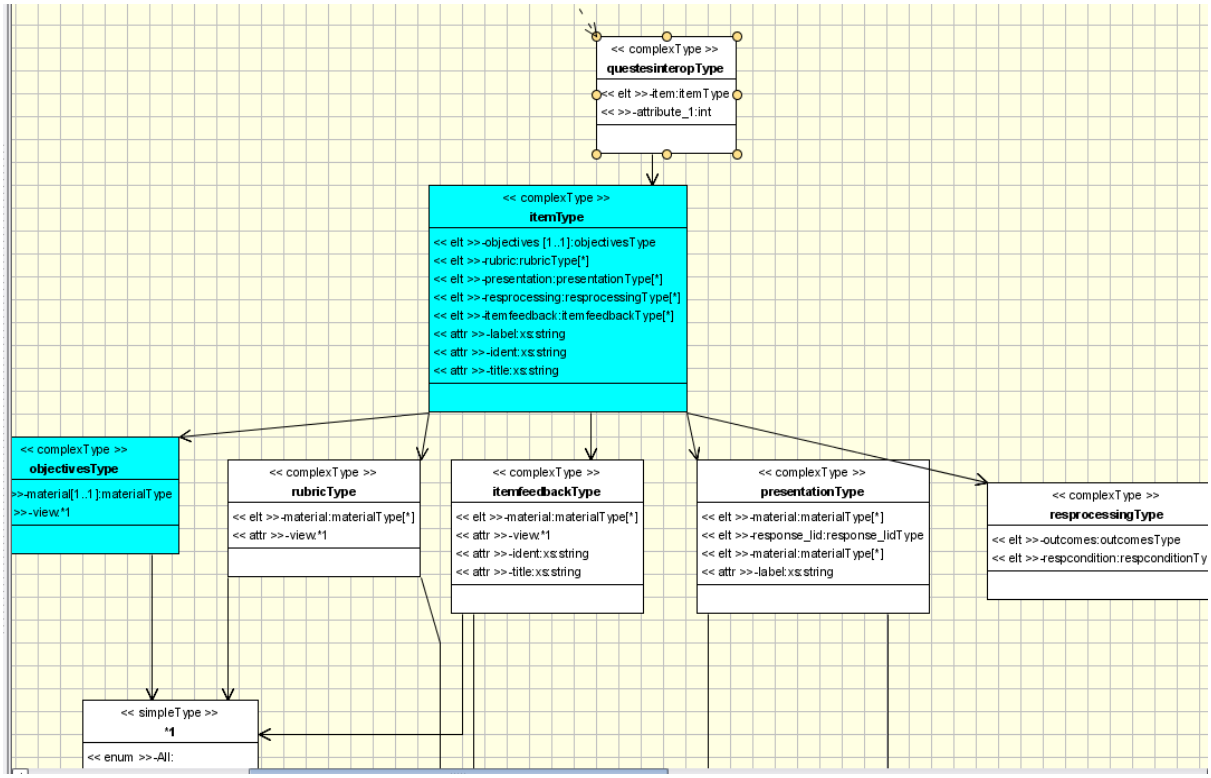

Figure 15 Definition modifications

Conclusions and future work

In many application domains where information resources are shared and routinely exchanged in the form of standard metadata, Application Profile is emerging as an important means to adapt the generic metadata schemas to the specific exigencies of a community, as amply discussed in the introduction. The XML Schema definition language is becoming the universally adopted notation for specifying the metadata, but it does it at a very detailed level, hence the exigency of expressing above it the metadata Information Model in a more readable and manageable format arises. The Unified Modeling Language is currently proposed by many as a convenient notation for this purpose, due to its graphical flavour and its high expressiveness power, and several authors have already proposed mappings from XML Schemas to UML models, as surveyed in the *UML Mapping of XML Schema* Section. In this context, the contribution of this paper is a UML profile for describing directly in the UML an Application Profile. In the *UML-based Application Profiling* Section the elements and the profiling rules, such as Base Schema, Schema_Mod, and so on, have been introduced. A prototype tool called U2Xaptly has been developed for evaluation purposes. The usage of this profile has then been illustrated in a simple case study of profiling the QTILite (QTILite 2002) Information Model.

The field on Application Profile is still in its early stages. In particular, many technical issues ask for further research, but ‘social’ and ‘political’ issues as well, for instance fostering the agreement on common formats and notations, and the establishment of common registries where Application Profiles can be published, need special attention. As said, the very notion of an AP supposes that the user-community recognizes the advantage to re-use the effort and results collectively produced and has the aim to increase the potential interoperability between the metadata collection they describe and other communities’ collections.

With regard to the specific issue dealt here of using UML for expressing APs, the proposed UML profile only covers the static description of metadata. Work is ongoing within the already

mentioned TELCERT project to also adopt UML for profiling the intended behaviour and standard protocols, following the emerging paradigm of Service Oriented Architectures.

This work in fact addresses only part of the scientific and technical challenges tackled within the European STREP TELCERT, in which new development technologies and techniques (such as UML) are pursued to accommodate the localisation of the learning technology specifications into an Application Profile to meet cultural, pedagogical and organisational needs within a user-community. Particular attention is devoted to the definition of appropriate Information Models that describe the core data objects and their properties and can be automatically mapped to a XML representation underneath. Another major concern of TELCERT, which has not been tackled here, is the development and exploitation of advanced conformance testing strategies, as a key element for enhancing interoperability. Indeed, the interoperability buzzword, from which this paper took its start, remains the final objective of future work and most likely the focus of next years efforts within the e-Learning domain.

Acknowledgements

This work has been financially supported in part by the European project TELCERT (FP6 STREP 507128) Technology Enhanced Learning Conformance - European Requirements and Testing. The authors would like to thank the partners of the above project for their useful hints and feedback, and in particular Prof. Ingo Dahn for his very fruitful suggestions. Federica Ciotti and Sara Santoponte contributed to the UML profile definition and to the U2XAptly tool development, respectively. Andrea Polini has been a valid and irreplaceable help throughout the process.

References

Bertolino, A. (2004, November) *Initial Recommendations on Advantage Testing Technologies. D09*. Retrieved April 30, 2005, from http://www.imsglobal.org/telcert.html

ADL (2005) *Advanced Distributed Learning*. Retrieved April 30, 2005, from http://www.adlnet.org/index.cfm

Ambler S. W. (2005) *UML Profile for Data Modeling*, Retrieved April 30, 2005 from www.agiledata.org/essays/umlDataModelingProfile.html

Application profiles Documentation and XML files. (2004, July) Retrieved April 30, 2005, from http://www.imsglobal.org/telcert.html

Application Profiles Workshop (2004, February). Zurich. Retrieved April 30, 2005, from http://www.imsglobal.org/D28_Dissemination/D28_workshop1.zip

Baker, T., Dekkers M., Heery R., Patel M. & Salokhe G. (2001, November) What Terms Does Your Metadata Use? Application Profiles as Machine-Understandable Narratives. *Journal of Digital Information*.

Bernauer, M., Kappel, G., & Kramler, G. (2003, November), *Representing XML Schema in UML - A UML Profile for XML Schema* . Technical Report. Retrieved April 30, 2005, from http://www.big.tuwien.ac.at/research/publications/papers03.html

Bernauer, M., Kappel, G., & Kramler, G. (2004, July), Representing XML Schema in UML - A Comparison of Approaches. *Proceedings of the 4th International Conference on Web Engineering (ICWE2004)*, LNCS 3140, Munich, Germany, 440-444.

Biron P.V., & Malhotra A. (2004, October) *XML Schema Part 2: Datatypes Second Edition.* Retrieved 28 October 2004 from http://www.w3.org/TR/xmlschema-2/

Booch, G., Christerson, M., Fuchs, M., & Koistinen J. (1999 December).*UML for XML Schema Mapping specification*. Rational White Paper.

Bray T., Paoli J., Maler E., & Yergeau F. (2004, February) Extensible Markup Language (XML) 1.0 (Third Edition) Retrieved April 30, 2005, from http://www.w3.org/TR/REC-xml/

Carlson D. (2001) Modeling XML Applications with UML: Practical e-Business Applications. Addison Wesley

Dann I. (2005) IMS Application Profile Guidelines WhitePaper. Part 2 – Technical Manual to appear on http://www.imsglobal.org/

DESIRE (2000, March) *Metadata Registry Framework*. Retrieved April 30, 2005, from http://www.desire.org/html/research/deliverables/D3.5/d35.html

Duval E. (2004, February). Learning Technology Standardization: Making Sense of it All, *Int. Journal on Computer Science and Information Systems.* 1(1).33-43.

Duval E., Hodgins W., Sutton S. & Weibel R. (2002, April). Metadata Principles and Practicalities, *D-Lib Magazine*, 8(4). Retrieved April 30, 2005, from http://www.dlib.org/dlib/april02/weibel/04weibel.html

Eckstein R. & Eckstein S.(2004). *XML und Datenmodellierung*. dpunkt.verlag, 2004.

EDOC (2005) *UML Profile for enterprise distributed Object Computing (EDOC)* Retrieved April 30, 2005 from http://www.omg.org/technology/documents/formal/edoc.htm

Fallside D.C, & Walmsley P. (2004, October) *XML Schema Part 0: Primer Second Edition*, Retrieved 28 October 2004 from http://www.w3.org/TR/xmlschema-0/

Federal Geographic Data Committee (1998) *Guidelines for Creating a Profile for the Content Standard for Digital Geospatial Metadata.* FGDC-STD-001-1998. Retrieved April 30, 2005, from http://www.fgdc.gov/metadata/csdgm/profile.html

Hatala M., Richards G. Eap T. & Willms J. (2004, May). The Interoperability of Learning Object Repositories and Services: Standards, Implementations and Lessons Learned. *ACM Proceeding of WWW 2004*, New York, USA, 19-27.

Heery R., & Patel M. (2000, September) Application Profiles: Mixing and Matching Metadata Schemas. Ariadne 25. Retrieved April 30, 2005, from http://www.ariadne.ac.uk/issue25/app-profiles/

HyperModel tool (n.d.) Retrieved April 30, 2005, from http://www.xmlmodeling.com/hyperModel/

Hunter J., & Lagoze C. (2001, May) Combining RDF and XML Schema s to Enhance Interoperability Between Metadata Application Profiles. *Proceedings of International WWW Conference*. (10), Hong-Kong.

Jeckle, M. (2001, August). Practical usage of W3C's XML-Schema and a process for generating schema structures from UML models. *Proceedings of International on*

Advances in Infrastructure for E-Business, Science, and Education on the Internet (SSGRR 2001). 6 (12). L'Aquila Italy.

Klyne G., & Carroll J.J. (2004, February) *Resource Description Framework (RDF):Concepts and Abstract Syntax* Retrieved 10 February 2004 from http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

Krumbein T. & Kudrass, T. (2003) Rule-Based Generation of XML Schema s from UML Class Diagrams. *Proceedings of Web Databases (WebDB).* Berlin 213–227.

IMS (2005) IMS Global Learning Consortium Retrieved April 30, 2005, from http://www.imsglobal.org/

LTCS (2005) IEEE Learning Technology Standards Committee. Retrieved April 30, 2005, from http://ltsc.ieee.org/

Marchal, B. (2004, March) *UML, XMI, and code generation* Retrieved April 30, 2005, http://www-106.ibm.com/developerworks/xml/library/x-wxxmcol.html

Marchal, B. (2004, May) *UML, XMI, and code generation, Part 2* Retrieved April 30, 2005 http://www-106.ibm.com/developerworks/xml/library/x-wxxmcol.html

Marchal, B. (2004, June) *UML, XMI, and code generation, Part 3* Retrieved April 30, 2005 http://www-106.ibm.com/developerworks/xml/library/x-wxxmcol.html

Marchal, B. (2004, August*) UML, XMI, and code generation, Part 4* August 2004 a Retrieved April 30, 2005 http://www-106.ibm.com/developerworks/xml/library/x-wxxmcol.html

Miller, P. (2000 June). Interoperability, What is it and Why should I want it? Ariadne 24. Retrieved April 30, 2005, from http://www.ariadne.ac.uk/issue24/interoperability/

Poseidon (2005) Retrieved April 30, 2005, from http://www.gentleware.com/

Provost W.(2002) *UML For W3C XML Schema Design*. Retrieved April 30, 2005, from http://www.xml.com/lpt/a/2002/08/07/wxs_uml.html

QTI (2004, June*) IMS Question & Test Interoperability*. Retrieved April 30, 2005, from http://www.imsglobal.org/question/index.cfm

QTILite (2002, February*) IMS Question & Test Interoperability QTILite Specification Final Specification Version 1.2.* Retrieved April 30, 2005, from http://www.imsglobal.org/question/qtiv1p2/imsqti_litev1p2.html

Routledge N., Goodichild A.,& Bird L. (2002, February). UML and XML Schema . *Proceedings of The Thirteenth Australasian Database Conference (5).* Melbourne, Australia. 157-166.

Routledge N. (2002.) *XML-Schema Profile Definition for UML Honours thesis extract*. Retreived April 30, 2005, from http://titanium.dstc.edu.au/papers/xml-schema-profile.pdf

Smythe C. (2004, July) *State-of-the-Art Report on Technologies and Techniques for Testing* Retrieved August 1, 2004 from http://www.imsglobal.org/telcert/D03_v1.4.pdf

TELCERT Technology Enhanced Learning Conformance - European Requirements and Testing Retrieved July 15, 2005 from http://www.opengroup.org/telcert/

Thompson H. S., Beech D., Maloney M., & Mendelsohn N., (2004, October). *XML Schema Part 1: Structures Second Edition*. Retrieved 28 October 2004 from http://www.w3.org/TR/xmlschema-1/

UML (2005), *Unified Modeling Language (UML).* Retrieved April 30, 2005 from http://www.uml.org/.

UML Profile for CORBA, v 1.0 (2005) Retrieved April 30, 2005 from http://www.omg.org/technology/documents/formal/profile_corba.htm.

XSLT (2005) *XSLT The Extensible Stylesheet Language* Retrieved April 30, 2005 from www.w3.org/TR/xslt

XMI (2005) *XML Metadata Interchange (XMI).* Retrieved April 30, 2005 from http://www.omg.org/technology/documents/formal/xmi.htm

Warmer, J., & Kleppe, A. (2003) *OCL The Object Constraint Language* (2nd ed.). Addison-Wesley.

W3C (2005) World Wide Web Consortium (W3C). Retrieved April 30, 2005, from http://www.w3.org/

Footnote

[1] Extension points can be localized in specification schemas by looking for the wildcards *xs:any* for places where new elements can be inserted or *xs:anyAttribute* to add new attributes.

[2] TELCERT is an ongoing Technology Enhanced Learning STREP project under the EU 6th Framework programme carried on by a consortium of eLearning providers, research and industry organisations. TELCERT has the purpose of developing innovative software testing and conformance systems to assure interoperability in eLearning content and technology and accelerate market take up of open technologies based on specifications and standards.

[3] As said, the Information Model binding could be represented with different formats such as RTF or even in a textual form. However the general trend is toward adoption of the XML schema.

[4] For readability, the Schema Information is sometime represented in a separated Class Diagram. In this case a *Schema Information Class Diagram* is used to define the namespace information of XML Schema , while a different Class Diagram includes all the elements of the <<schema_mod>> package. When this is the case, the latter Class Diagram is called the *Main Application Profile Class Diagram*.

[5] Comments are added to the XML Schema for readability purposes, of course they are not automatically generated

[6] UML identifies with braces the enclosed tagged values of an attribute; however, some UML tools (as, for example *Poseidon* (2005)) do not show explicitly the tagged values but have a property window which lists all the element properties.