ebXML Registry Tutorial:  
UML Class diagram to ebRIM mapping

Version 0.6

Draft Technical Note, January 19, 2006

Document identifier:  
regrep-UML-to-ebRIM-tutorial-06

Location:  
http://www.oasis-open.org/committees/regrep/documents/...

Editors:

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ivan Bedini</td>
<td>France Telecom</td>
</tr>
<tr>
<td>Farrukh Najmi</td>
<td>Sun Microsystems</td>
</tr>
<tr>
<td>Nikola Stojanovic</td>
<td>RosettaNet</td>
</tr>
</tbody>
</table>

Contributors:

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diego Ballve</td>
<td>Individual</td>
</tr>
</tbody>
</table>

Abstract:  
The document includes a standard methodology and guidelines for mapping a domain specific information model in UML format to the ebXML Registry Information Model.

Status:  
This document is an OASIS ebXML Registry Technical Committee Working Draft Technical Note. Committee members should send comments on this specification to the regrep@lists.oasis-open.org list. Others should subscribe to and send comments to the regrep-comment@lists.oasis-open.org list. To subscribe, send an email message to regrep-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the OASIS ebXML Registry TC web page (http://www.oasis-open.org/committees/regrep/).
# Table of Contents

1 Introduction ................................................................................................................................................................................. 6  
  1.1 Terminology ............................................................................................................................................................................. 6  
  1.2 Conventions ............................................................................................................................................................................. 6  
2 Overview ......................................................................................................................................................................................... 7  
  2.1 Overview of UML ....................................................................................................................................................................... 7  
  2.2 Overview of Person Information Model .................................................................................................................................. 7  
  2.3 Overview of ebXML Registry Information Model ................................................................................................................ 8  
  2.3.1 RegistryObject ........................................................................................................................................................................ 9  
  2.3.2 Object Identification .............................................................................................................................................................. 9  
  2.3.3 Object Naming and Description ........................................................................................................................................ 9  
  2.3.4 Object Attributes ................................................................................................................................................................. 10  
  2.3.4.1 Slot Attributes ............................................................................................................................................................... 10  
  2.3.5 Object Classification ............................................................................................................................................................ 10  
  2.3.6 Object Association .............................................................................................................................................................. 11  
  2.3.7 Object References To Web Content .................................................................................................................................. 11  
  2.3.8 Object Packaging .............................................................................................................................................................. 11  
  2.3.9 Service Description ........................................................................................................................................................... 12  
3 Mapping a Domain Specific UML Model to ebRIM .......................................................................................................................... 13  
  3.1 Class Mapping .......................................................................................................................................................................... 13  
  3.1.1 Defining a Sub-Class of ExtrinsicObject .............................................................................................................................. 13  
  3.2 Interface Mapping ..................................................................................................................................................................... 14  
  3.3 Inheritance Mapping ............................................................................................................................................................... 15  
  3.3.1 Mapping of Multiple Inheritance ..................................................................................................................................... 15  
  3.4 Method Mapping ....................................................................................................................................................................... 15  
  3.5 Association Mapping ............................................................................................................................................................... 15  
  3.5.1 Navigability / Direction Mapping .................................................................................................................................... 15  
  3.5.2 Role Name / Association Name Mapping ......................................................................................................................... 15  
  3.5.3 Defining a New Association Type ..................................................................................................................................... 15  
  3.5.4 Aggregation Mapping .......................................................................................................................................................... 17  
  3.5.5 Composition Mapping ....................................................................................................................................................... 17  
  3.5.6 N-ary Association Mapping ........................................................................................................................................... 18  
  3.5.7 XOR Associations ............................................................................................................................................................. 18  
  3.6 Attribute Mapping ................................................................................................................................................................... 18  
  3.6.1 Mapping to Identifier .......................................................................................................................................................... 19  
    3.6.1.1 Mapping to id Attribute .............................................................................................................................................. 19  
    3.6.1.2 Mapping to Logical Id (lid) Attribute .......................................................................................................................... 20  
    3.6.1.3 Mapping to ExternalIdentifier ................................................................................................................................ 20  
  3.6.2 Mapping to Name and Description .................................................................................................................................. 21  
  3.6.3 Mapping to Classification ................................................................................................................................................ 21  
  3.6.4 Mapping to ExternalLink ................................................................................................................................................... 22  
  3.6.5 Direct Mapping to ebRIM Attribute ................................................................................................................................... 22  
  3.6.6 Mapping to Slot ..................................................................................................................................................................... 22  
    3.6.6.1 Mapping to rim.Slot.slotName ...................................................................................................................................... 22  
    3.6.6.2 Mapping to rim.Slot.slotType ...................................................................................................................................... 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6.3</td>
<td>Mapping to rim.Slot.values</td>
<td>23</td>
</tr>
<tr>
<td>3.7</td>
<td>Enumerated Type Mapping</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>Using ClassificationSchemes</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>Use Cases for ClassificationSchemes</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>Canonical ClassificationSchemes</td>
<td>25</td>
</tr>
<tr>
<td>4.3</td>
<td>Extending ClassificationSchemes</td>
<td>26</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Use Cases for Extending ClassificationSchemes</td>
<td>26</td>
</tr>
<tr>
<td>4.4</td>
<td>Defining New ClassificationSchemes</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>PIM UML class diagram mapping to [ebRIM]</td>
<td>27</td>
</tr>
<tr>
<td>5.1.1</td>
<td>PIM Class Mapping</td>
<td>27</td>
</tr>
<tr>
<td>5.1.2</td>
<td>PIM attributes mapping</td>
<td>27</td>
</tr>
<tr>
<td>5.2</td>
<td>PIM Association mapping</td>
<td>30</td>
</tr>
<tr>
<td>5.2.1</td>
<td>PIM compositions mapping</td>
<td>31</td>
</tr>
<tr>
<td>5.3</td>
<td>PIM classifications mapping</td>
<td>31</td>
</tr>
</tbody>
</table>
Index of Tables

Table 1: [ebRIM ] object type definition and mapping for the PIM source UML model class diagram........ 27
Table 2: PIM classes’ attributes mapping.......................................................................................... 30
Table 3: PIM association mapping to [ebRIM]................................................................................... 30
Table 4: PIM composition mapping to [ebRIM]................................................................................... 31
Table 5: Classification mapping for the source model PIM concepts.................................................... 31
Illustration Index

Figure 1: Person Information Model: A Sample Domain Specific Model.................................................. 7
Figure 2: Person Information Model: Inheritance View........................................................................ 8
Figure 3: ebXML Registry Information Model, High Level Public View.................................................. 8
Figure 4: ebXML Registry Information Model, Inheritance View........................................................... 9
Figure 5: ObjectType ClassificationScheme: Before and After Extension for LifeEvent.......................... 14
Figure 6: ObjectType ClassificationScheme: Before and After Extension For Spouse............................ 16
Figure 7: Sample Association instance between a Husband and Wife pair............................................. 17
Figure 8: Attribute Mapping Algorithm Flowchart................................................................................ 19
Figure 9: Geography ClassificationScheme Example............................................................................. 25
1 Introduction

The document includes a standard methodology for mapping a domain specific information model to the ebXML Registry Information Model.

As more and more organization are adopting ebXML Registry standard they are faced with the recurring need to map between their domain specific information model to the ebXML Registry Information Model [ebRIM] in order to use the registry to manage their domain specific artifacts. Currently this mapping is being done in an ad hoc manner.

This technical note provides the necessary guidelines, design patterns and algorithms to customize an ebXML Registry for a specific domain. Specifically, it enables a consistent mapping from domain specific information UML models to ebXML Registry Information Model.

It is not the purpose of this document to educate the reader on ebXML Registry [ebRIM], [ebRS], information modeling or the Unified Modeling Language [UML]. The reader of this document should have a good understanding of the ebXML Registry specifications and the UML 1.5 specification.

1.1 Terminology

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in [RFC2119].

1.2 Conventions

Throughout the document the following conventions are employed to define the data structures used. The following text formatting conventions are used to aide readability:

- UML Diagrams
  UML diagrams are used as a way to concisely describe information models in a standard way. They are not intended to convey any specific Implementation or methodology requirements.

- Identifier Placeholders
  Listings may contain values that reference ebXML Registry objects by their id attribute. These id values uniquely identify the objects within the ebXML Registry. For convenience and better readability, these key values are replaced by meaningful textual variables to represent such id values.
  For example, the following placeholder refers to the unique id defined for the canonical ClassificationNode that defines the Organization ObjectType defined in [ebRIM]:

  <id="${CANONICAL_OBJECT_TYPE_ID _ORGANIZATION}"/>

- Constants
  Constant values are printed in the Courier New font always, regardless of whether they are defined by this document or a referenced document. In addition, constant values defined by this document are printed using bold face. The following example shows the canonical id and lid for the canonical ObjectType ClassificationScheme defined by [ebRIM]:

  <rim:ClassificationScheme
     lid="urn:oasis:names:tc:ebxml-regrep:classificationScheme:ObjectType"
     id="urn:uuid:3188a449-18ac-41fb-be9f-99a1adca02cb">

1 Example Values

These values are represented in italic font. In the following, an example of a RegistryObject's name "ACME Inc." is shown:

  <rim:Name>
    <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
  </rim:Name>
2 Overview

This chapter provides an overview of ebXML Registry Information Model [ebRIM] and the sample domain specific Person Information Model (PIM). The PIM is the source information model for the mapping patterns defined by this document. The [ebRIM] is the target for the mapping patterns defined by this document.

The information presented is informative and is not intended to replace the normative information defined by ebXML Registry and UML specifications.

2.1 Overview of UML

This document will not provide an overview of UML. The reader SHOULD review UML tutorials [TUT] to get a rapid understanding of [UML]. The reader MAY refer to [UML] if a deeper understanding is needed. Although UML defines many different types of diagrams the focus of this document is the UML Class diagram. The reader SHOULD familiarize themselves with the UML Class Diagram notation using [TUT] and [UML].

2.2 Overview of Person Information Model

Throughout this document we use a sample domain specific information model called Person Information Model (PIM). This document will demonstrate the mapping principals described using the PIM as source model and [ebRIM] as the target model for the mapping.

Figure 1: Person Information Model: A Sample Domain Specific Model

Figure 1 shows the UML Class diagram for the Person Information Model. The model shows that:
1. A Person has several LifeEvents:
   - BirthEvent: Marks the birth of the associated Person
   - MarriageEvent: Marks a marriage of the associated Person
   - BirthingEvent: Marks a delivery of one or more babies where the associated person is a parent.
   - DeathEvent: Marks the death of the associated Person

2. A Person has a PhysicalTraits which is a collection of various physical traits that describe the Person.

3. A Person has a birth mother and birth father which are also Person

4. A Person has children which are also Person

5. Each class MAY define various attributes as shown within the box for each class.

---

**Figure 2: Person Information Model: Inheritance View**

Figure 2 above shows another class diagram for the model that shows the inheritance view of the model. Here we see that the various Event classes inherit from the same LifeEvent base class and further specialize it for that specific event.

---

**2.3 Overview of ebXML Registry Information Model**

This section summarizes the ebXML Registry Information Model [ebRIM]. This model is the target of the mapping defined in this document. The reader SHOULD read [CMRR] for a more detailed overview of ebXML Registry as a whole.

**Figure 3: ebXML Registry Information Model, High Level Public View**

The ebXML registry defines a Registry Information Model [ebRIM] that specifies the standard metadata that may be submitted to the registry. Figure 3 presents the UML class diagram representing the Registry Information Model. Figure 4, shows the inheritance relationships in among the classes of the ebXML Registry Information Model.
The next few sections describe the main features of the information model.

### 2.3.1 RegistryObject

This is an abstract base class used by most classes in the model. It provides minimal metadata for registry objects. The following sections use the Organization sub-class of RegistryObject as an example to illustrate features of the model.

### 2.3.2 Object Identification

A RegistryObject has a globally unique id which is a UUID based URN:

```xml
<rim:Organization id="urn:uuid:dafa4da3-1d92-4757-8fd8-ff2b8ce7a1bf"/>
```

Listing 1: Example of id attribute

Since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which is unique for different logical objects. However the lid attribute value MUST be the same for all versions of the same logical object. The lid attribute value is a URN that MAY potentially be human friendly:

```xml
<rim:Organization id=${ACME_ORG_ID}
    lid="urn:acme:ACMEOrganization">
</rim:Organization>
```

Listing 2: Example of lid Attribute

A RegistryObject MAY also have any number of ExternalIdentifiers which may be any string value within an identified ClassificationScheme.

```xml
<rim:Organization id=${ACME_ORG_ID}
    lid="urn:acme:ACMEOrganization">
    <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
        identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
        value="ACME"/>
</rim:Organization>
```

Listing 3: Example of ExternalIdentifier

### 2.3.3 Object Naming and Description

A RegistryObject MAY have a name and a description which consists of one or more strings in one or more local languages. Name and description need not be unique across RegistryObjects.

```xml
<rim:Organization id=${ACME_ORG_ID}
    lid="urn:acme:ACMEOrganization">
    <rim:Name>
        <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
    </rim:Name>
    <rim:Description>
        <rim:LocalizedString value="ACME is a provider of Java software." xml:lang="en-US"/>
    </rim:Description>
    <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
        identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}>
</rim:Organization>
```

---

Figure 4: ebXML Registry Information Model, Inheritance View
2.3.4 Object Attributes

For each class in the model, [ebRIM] defines specific attributes. Examples of several of these attributes such as id, lid, name and description have already been introduced.

2.3.4.1 Slot Attributes

In addition the model provides a way to add custom attributes to any RegistryObject instance using instances of the Slot class. The Slot instance has a Slot name which holds the attribute name and MUST be unique within the set of Slot names in that RegistryObject. The Slot instance also has a ValueList that is a collection of one or more string values.

The following example shows how a custom attribute named “urn:acme:slot:NASDAQSymbol” and value “ACME” MAY be added to a RegistryObject using a Slot instance.

```
<rim:Organization id=${ACME_ORG_ID}
    lid="urn:acme:ACMEOrganization">
    <rim:Slot name="urn:acme:slot:NASDAQSymbol">
        <rim:ValueList>
            <rim:Value>ACME</rim:Value>
        </rim:ValueList>
    </rim:Slot>
    <rim:Name>
        <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
    </rim:Name>
    <rim:Description>
        <rim:LocalizedString value="ACME makes Java. Provider of free Java software." xml:lang="en-US"/>
    </rim:Description>
    <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
        identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
        value="ACME"/>
</rim:Organization>
```

Listing 5: Example of a Dynamic Attribute Using Slot

2.3.5 Object Classification

Any RegistryObject may be classified using any number of Classification instance. A Classification instance references an instance of a ClassificationNode as defined by [ebRIM]. The ClassificationNode represents a value within the ClassificationScheme. The ClassificationScheme represents the classification taxonomy.

```
<rim:Organization id=${ACME_ORG_ID}
    lid="urn:acme:ACMEOrganization">
    <rim:Slot name="urn:acme:slot:NASDAQSymbol">
        <rim:ValueList>
            <rim:Value>ACME</rim:Value>
        </rim:ValueList>
    </rim:Slot>
    <rim:Name>
        <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
    </rim:Name>
    <rim:Description>
        <rim:LocalizedString value="ACME makes Java. Provider of free Java software."/>
    </rim:Description>
    <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
        identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
        value="ACME"/>
</rim:Organization>
```
2.3.6 Object Association

Any RegistryObject MAY be associated with any other RegistryObject using an Association instance where one object is the sourceObject and the other is the targetObject of the Association instance. An Association instance MAY have an associationType which defines the nature of the association. There are a number of predefined Association Types that a registry must support to be [ebRIM] compliant as shown in Table 1. [ebRIM] allows this list to be extensible.

The following example shows an Association between the ACME Organization instance and a Service instance with the associationType of “OffersService”. This indicates that ACME Organization offers the specified service (Service instance is not shown).

```xml
<rim:Association
  id=${ASSOCIATION_ID}
  associationType=${CANONICAL_ASSOCIATION_TYPE_OFFERS_SERVICE_ID}
  sourceObject=${ACME_ORG_ID}
  targetObject=${ACME_SERVICE1_ID}/>
```

Listing 7: Example of Object Association

2.3.7 Object References To Web Content

Any RegistryObject MAY reference web content that are maintained outside the registry using association to an ExternalLink instance that contains the URL to the external web content. The following example shows the ACME Organization with an Association to an ExternalLink instance which contains the URL to ACME’s web site. The associationType of the Association MUST be of type “ExternallyLinks” as defined by [ebRIM].

```xml
<rim:ExternalLink externalURI="http://www.acme.com"
  id=${ACME_WEBSITE_EXTERNAL_ID}>
  <rim:Association
    id=${EXTERNALLYLINKS_ASSOCIATION_ID}
    associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}
    sourceObject=${ACME_WEBSITE_EXTERNAL_ID}
    targetObject=${ACME_ORG_ID}/>
```

Listing 8: Example of Reference to Web Content Using ExternalLink

2.3.8 Object Packaging

RegistryObjects may be packaged or organized in a hierarchical structure using a familiar file and folder metaphor. RegistryPackage instances serve as folders while RegistryObject instances serve as files in this metaphor. A RegistryPackage instances groups logically related RegistryObject instances together as
members of that RegistryPackage.

The following example creates a RegistryPackage for Services offered by ACME Organization organized in RegistryPackages according to the nature of the Service. Each Service is referenced using the ObjectRef type defined by [ebRIM].

```
<rim:RegistryPackage
  id=${ACME_SERVICES_PACKAGE_ID}>
  <rim:RegistryObjectList>
    <rim:ObjectRef id=${ACME_SERVICE1_ID}>
      <rim:RegistryPackage
        id=${ACME_PURCHASING_SERVICES_PACKAGE_ID}>
        <rim:ObjectRef id=${ACME_PURCHASING_SERVICE1_ID}>
        <rim:ObjectRef id=${ACME_PURCHASING_SERVICE2_ID}>
      </rim:RegistryPackage>
    </rim:ObjectRef>
    <rim:ObjectRef id=${ACME_HR_SERVICES_PACKAGE_ID}>
      <rim:ObjectRef id=${ACME_HR_SERVICE1_ID}>
      <rim:ObjectRef id=${ACME_HR_SERVICE2_ID}>
    </rim:ObjectPackage>
  </rim:RegistryObjectList>
</rim:RegistryPackage>
```

Listing 9: Example of Object Packaging Using RegistryPackages

### 2.3.9 Service Description

Service description MAY be defined within the registry using the Service, ServiceBinding and SpecificationLink classes defined by [ebRIM]. This MAY be used to Publish service descriptions such as WSDL and ebXML CPP/A.
3 Mapping a Domain Specific UML Model to ebRIM

This chapter identifies several common mapping patterns that are encountered when a domain specific information model is mapped to [ebRIM]. For each such pattern we define a consistent heuristic or algorithm to perform the mapping. The goal is to make it easier for domain experts to utilize the ebXML Registry for their domain and to have consistency across all domain-specific uses of ebXML Registry.

A source model may be in many different formats such as Java, XML, SQL and so on. [UML] is a standard for information model description and therefore this document assumes the source information model is described in UML. [UML] terminology and notation is consistently used throughout this chapter and this document.

It should be understood that the mappings produced by applying the heuristics and algorithms described in this document will be only as good as the input UML model (this is the old garbage-in, garbage-out principal). A person applying these mapping patterns (the mapper) MAY choose to deviate from these patterns to compensate for special situations in the input UML model. Any mapping pattern not covered by this document MAY be addressed in an ad hoc manner by the mapping. Suggestions for improvements to the mapping should be sent to the Editors listed on the title page of this document.

3.1 Class Mapping

This section defines how a class in the source model is mapped to a class in [ebRIM]. Mapping of attributes of the source class will be discussed in section 3.6.

A class in the source model is mapped to [ebRIM] using the following algorithm:

1. **Direct Class Mapping To Rim:** First determine if there is a class in ebRIM that closely matches the class in the source model. For example the Person class in PIM matches closely to the Person class in [ebRIM]. Thus it is preferred that the Person class in PIM is mapped to the Person class in [ebRIM].

2. **Mapping To ExtrinsicObject Sub-Class:** If no class in [ebRIM] is a good match then define a new sub-class of ExtrinsicObject class in [ebRIM] and map the source class to the new sub-class. See section 3.1.1 on how to define a new sub-class of ExtrinsicObject. For example the various LifeEvent classes in PIM SHOULD be mapped to sub-classes of ExtrinsicObject where the class names match the various LifeEvent class names.

3.1.1 Defining a Sub-Class of ExtrinsicObject

This section provides the steps to define a new sub-class of ExtrinsicObject class.

To define a sub-class of ExtrinsicObject you MUST extend the canonical ObjectType ClassificationScheme and add a new ClassificationNode as a child or descendent of the canonical ClassificationNode for ExtrinsicObject in the ObjectType ClassificationScheme.

For example to extend the ObjectType ClassificationScheme for the LifeEvent classes in PIM the following ClassificationNode hierarchy MUST be submitted to the ebXML Registry via a SubmitObjectsRequest.

Note that:

- The id attribute values SHOULD have actual id values. See 5.3 for generating unique id values.
- The parent attribute of the LifeEvent ClassificationNode is the id of the ExtrinsicObject ClassificationNode in the ObjectType ClassificationScheme.
- Figure 5 shows the structure of the ObjectType ClassificationScheme before and after the extension for mapping the LifeEvent classes from PIM.

```xml
<!-- Add LifeEvent classes to ObjectType ClassificationScheme -->
<rim:ClassificationNode code="LifeEvent" id="${LIFE_EVENT_NODE_ID}"
```

Copyright © OASIS Open 2005. All Rights Reserved.
Listing 10: Example of Adding LifeEvent Classes and sub-classes to ObjectType ClassificationScheme

Figure 5: ObjectType ClassificationScheme: Before and After Extension for LifeEvent

3.2 Interface Mapping

Interfaces are classes that only have methods and have no attributes (they may contain constant attributes). They should be mapped in a manner similar to Class mapping. The only difference is that Interface methods that follow the getter method design pattern MAY be mapped to corresponding attributes.

For example, if the Person class in PIM model was an interface that had a method called getAge(), then that method MAY be mapped to an age attribute in the corresponding [ebRIM] class.
### 3.3 Inheritance Mapping

A class in the source model may have a generalization or inheritance relationship with another class in the model. For example, the BirthEvent, MarriageEvent, BirthingEvent and DeathEvent classes have an inheritance relationship with the LifeEvent class in PIM.

Such inheritance relationships SHOULD be reflected in the mapping to [ebRIM] by defining a corresponding inheritance relationship among the ClassificationNodes defined when extending the ObjectType scheme. This has already been illustrated in section 3.1.1 and Figure 5.

#### 3.3.1 Mapping of Multiple Inheritance

A special case is "multiple inheritance" where the source model has multiple base classes for the same derived class. There is no direct support for multiple inheritance in [ebRIM]. In case the source model has a derived class with multiple base classes, the mapping SHOULD choose one base class to map as the base ClassificationNode in the ObjectType ClassificationScheme. The remaining base classes SHOULD be mapped as ClassificationNodes in the ObjectType ClassificationScheme and should be associated with the derived class using an Association whose associationType is the id for the canonical ClassificationNode “Extends” or “Implements” within the canonical AssociationType ClassificationScheme.

### 3.4 Method Mapping:

There is no support for mapping methods from a source model to [ebRIM]. Methods that follow a getter method MAY be mapped to an attribute as defined in section 3.3.

### 3.5 Association Mapping

A UML Association in the source model SHOULD be mapped to an [ebRIM] Association.

#### 3.5.1 Navigability / Direction Mapping

Associations in UML MAY be directed or undirected. Associations in [ebRIM] are always implicitly directed from the sourceObject to the targetObject of an Association.

Directed UML associations MUST map the Class at the arrowhead end as targetObject and the Class at the other as sourceObject. In case of Undirected UML associations the mapper MAY specify the mapping of the Classes at each end to sourceObject or targetObject using their best judgement.

#### 3.5.2 Role Name / Association Name Mapping

UML defines for an association, an association name as well as two role names (one for each end of the association).

The role name in the UML mapping at the targetObject end of the association, if present, SHOULD be mapped to the associationType. If the role name at the targetObject end (target role name) is not present then the association name SHOULD be mapped to the associationType.

In addition, the target role name (or UML association name) MAY also be mapped to the Association name in ebRIM.

#### 3.5.3 Defining a New Association Type

This section provides the steps to define a new Association Type.

To define a Association Type you MUST extend the canonical AssociationType ClassificationScheme and add a new ClassificationNode as a child or descendent of the AssociationType ClassificationScheme.

For example to extend the AssociationType ClassificationScheme for the “spouse”, “husband” and “wife” association in PIM the following ClassificationNode hierarchy SHOULD be submitted to the ebXML Registry via a SubmitObjectsRequest.

Note that:
Figure 5 shows the structure of the AssociationType ClassificationScheme before and after the extension for mapping the Spouse Association Types from PIM.

It is a good idea to organize AssociationTypes hierarchically even though the source model may not have those semantics defined. For example it makes good sense to define the “Husband” and “Wife” AssociationTypes as children of the “Spouse” AssociationType.

<!-- Add Spouse, Husband, Wife to AssociationType ClassificationScheme --
<rim:ClassificationNode code="Spouse" id="${SPOUSE_NODE_ID}"
  parent="urn:uuid:6902675f-2f18-44b8-888b-c91db8b96b4d">
  <rim:Name>
    <rim:LocalizedString charset="UTF-8" value="Spouse"/>
  </rim:Name>
</rim:ClassificationNode>
<rim:ClassificationNode code="Husband"
  id="${HUSBAND_NODE_ID}"
  parent="urn:uuid:6902675f-2f18-44b8-888b-c91db8b96b4d">
  <rim:Name>
    <rim:LocalizedString charset="UTF-8" value="Husband"/>
  </rim:Name>
</rim:ClassificationNode>
<rim:ClassificationNode code="Wife"
  id="${WIFE_NODE_ID}"
  parent="urn:uuid:6902675f-2f18-44b8-888b-c91db8b96b4d">
  <rim:Name>
    <rim:LocalizedString charset="UTF-8" value="Wife"/>
  </rim:Name>
</rim:ClassificationNode>

Listing 11: Example of Adding Spouse Association Types

Figure 6: ObjectType ClassificationScheme: Before and After Extension For Spouse

Figure 7 shows an example UML instance diagram to show two Associations between Person “PierreCurie” and Person “MarieCurie” in PIM. Note that the husbandToWife association has “PierreCurie” as the sourceObject and “MarieCurie” as the targetObject while the wifeToHusband associations has the two reversed.
### Aggregation Mapping

A UML Aggregation maps to multiple [ebRIM] Associations in a manner consistent with earlier sections.

### Composition Mapping

When a UML Class (Container) wholly contains another class (Contained) then the UML Association between the two is called a UML Composition. The Composition Association is denoted with a filled diamond at the source end of the Association.

An example of composition in PIM is where the Person class is the container while the PhysicalTraits class is the contained class.

A composition association in UML can be mapped as a [ebRIM] association or like a slot as follow:

1. The container class and the contained class map to [ebRIM] as defined by section 3.1.
2. The composition Association maps to a Slot instance that is defined for the container RegistryObject.
3. The composition Slot MUST have as the value of its “name” attribute,
   
   a. The target role name from the UML Association, or if that is not present
   
      b. The name of the UML Association

4. The composition Slot MUST have as the value of its “slotType” attribute, the logical lid of the canonical DataType “ObjectRef”. This value is:

   `urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef`

5. The composition Slot MUST have as the value of its “values” attribute, a list of String where each String MUST be the value of the id attribute of an object that is composed or contained by the container RegistryObject

Note that the ebXML Registry does not enforce the semantics of composition Associations. Specifically, deleting a container object does not automatically delete contained objects.
The following example shows how the composition association between a Person instance and a PhysicalTraits instance in PIM maps to [ebRIM].

```xml
--The ExtrinsicObject of objectType Person for Person PierreCurie -->
<rim:ExtrinsicObject id="${PIERRECURIE_PERSON_ID}" mimeType="text/xml"
    objectType="${OBJECT_TYPE_PERSON_ID}"
    <rim:Slot name="physicalTraits"
    slotType="urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef">
        <rim:ValueList>
            <rim:Value>${PIERRECURIE_PHYSICAL_TRAITS_ID}</rim:Value>
        </rim:ValueList>
    </rim:Slot>
</rim:ExtrinsicObject>

--The ExtrinsicObject of objectType PhysicalTraits for Person PierreCurie -->
<rim:ExtrinsicObject id="${PIERRECURIE_PHYS_TRAITS_ID}" mimeType="text/xml"
    objectType="${OBJECT_TYPE_PHYS_TRAITS_ID}"
    <rim:Slot name="" slotType="">
        <rim:ValueList>
            ...
        </rim:ValueList>
    </rim:Slot>
</rim:ExtrinsicObject>
```

Listing 12: Example of Composition of PhysicalTraits Instance Within Person Instance

3.5.6 **N-ary Association Mapping**

UML N-ary associations involving three or more Classes is not commonly used and is not covered by this document in detail. It is suggested that RegistryPackage may be considered as a mapping for such n-ary Associations.

3.5.7 **XOR Associations**

XOR Associations as defined by UML are not commonly used in source models. XOR Associations may be mapped to [ebRIM] Associations and it MUST be the responsibility of the mapping to enforce the XOR constraints in an application specific manner.

3.6 **Attribute Mapping**

This section defines how attributes of a class in the source model are mapped to [ebRIM]. Mapping of the source class to [ebRIM] has been discussed in section 3.1.

Figure 8 provides the flowchart for the algorithm that SHOULD be used to map attributes from the source model to [ebRIM]. Each box in right column maps to a section later in the document that describes the mapping in detail.
3.6.1 Mapping to Identifier

Section 2.3.2 describes the various ways that a RegistryObject may be identified in [ebRIM].

3.6.1.1 Mapping to id Attribute

If the identifier value in source model conforms to a UUID based URN as shown below,

```
urn:uuid:dafa4da3-1d92-4757-8fd8-ff2b8ce7a1bf
```

Listing 13: Example of id attribute
and if it provides a globally unique identifier for the source class then it MUST be mapped to the id 
attribute in the target [ebRIM] class. Note that if the identifier value in the source model MUST be the 
same across different versions of the same logical instance of the source class then it MUST not be 
mapped to the id attribute. Instead it SHOULD be mapped to the Logical id (lid) attribute as defined next.

For a detailed description of the versioning capabilities of ebXML Registry and the lid attribute please see 
[ebRS] and [ebRIM] respectively.

3.6.1.2 Mapping to Logical Id (lid) Attribute

If the identifier value in the source model may be the same across all versions of an instance of the class 
then it SHOULD be mapped to the lid attribute of the target class in [ebRIM]. The registry requires that the 
lid attribute value:

• SHOULD be a URN

• MUST be unique across all logical RegistryObjects in the registry

• MUST be the same across all versions of the same logical RegistryObject

The lid attribute is a good way to assign a meaningful identifier to a RegistryObject. If the source attribute 
is a human friendly identifier for the source class then it MAY be a good candidate to be mapped to the lid 
attribute. Note that the source attribute value need not be a URN. If it is not a URN, then the mapping 
SHOULD define a deterministic algorithm for mapping the non-URN value to a URN value that meets 
above constraints on lid attribute values.

For example, the name attribute of a Person instance in PIM MAY be mapped to the lid attribute on the 
Person class in [ebRIM] sing the following algorithm:

```java
lid = "urn:pim:" + Person.name
```

For example the rim.Person instance for "MarieCurie" would look like:

```xml
<rim:Person id=${MARIECURIE_PERSON_ID}
  lid="urn:pim:MarieCurie">
...
</rim:Person>
```

Note that above example is slightly flawed because use of a person’s name in the algorithm does not 
guarantee that the lid would be unique since another person could have the same exact name. Also note 
that the urn:pim namespace MUST be registered with IANA to truly guarantee that it is a unique name 
space.

3.6.1.3 Mapping to ExternalIdentifier

If the attribute in the source model is an identifier for the source class instances but does not map to an id 
or lid attribute then it SHOULD be mapped to an ExternalIdentifier in [ebRIM]. The mapping MUST specify 
a ClassificationScheme instance that MUST be used as identificationScheme for the ExternalIdentifier.

For example, the nationalId attribute of the Person class in PIM may be mapped to an ExternalIdentifier 
that uses a ClassificationScheme named “NationalIdentifierScheme” as its identificationScheme attribute 
value. The mapping is responsible for defining the “NationalIdentifierScheme” ClassificationScheme as 
described in section 4.2.
3.6.2 Mapping to Name and Description

If the source attribute provides a name or description for the source class instance then it SHOULD be
mapped to the name or description attribute of the RegistryObject class in [ebRIM]. The
rim.RegistryObject.name and rim.RegistryObject.description attributes are of type InternationalString
which can contain the name and description value is multiple locales as composed LocalizedString
instances. This means that the mapping SHOULD map the name and description to the appropriate
locale.

For example the pim.Person class has a name attribute of datatype String. The mapping SHOULD map it
to the rim.Person.name attribute as shown below:

```
<rim:Person id=${MARIECURIE_PERSON_ID} lid="urn:pim:MarieCurie">
  <rim:Name>
    <rim:LocalizedString value="Marie Curie" xml:lang="en-US" />
    <rim:LocalizedString value="Marie Curie" xml:lang="fr" />
  </rim:Name>
</rim:Person>
```

Listing 15: Example of Mapping to name Attribute

Note that the xml:lang attribute in above example SHOULD be omitted when the default locale is implied.
Since a person’s name does not change with locale the above example would be better off specifying a
single LocalizedString with no xml:lang attribute specified. It is showing multiple locales for illustration
purposes only.

3.6.3 Mapping to Classification

If the source attribute is somehow classifying or categorizing the class instance then it SHOULD be
mapped to a Classification in [ebRIM]. For an overview of Classification see section 2.3.6.

For example, the rim.Person.gender attribute is of datatype Gender which is an Enumeration class where
the enumerated set of values are “Male”, “Female” and “Other”. The mapping MAY map
pim.Person.gender to a Classification on a rim.Person instance. Since a Classification requires a
ClassificationScheme, the mapping MUST specify the ClassificationScheme.

```
<rim:Person id=${MARIECURIE_PERSON_ID} lid="urn:pim:MarieCurie">
  <!--Classify Person as a Female using the Gender Taxonomy-->
  <rim:Classification id=${GENDER_CLASSIFICATION_ID}
    classificationNode=${GENDER_FEMALE_NODE_ID}
    classifiedObject=${MARIECURIE_PERSON_ID}>
  ...
</rim:Person>
```

Listing 16: Example of Mapping to name Attribute

Note that in above example the Gender ClassificationScheme is indirectly referenced via the
ClassificationNode for “Female” within that taxonomy.
### 3.6.4 Mapping to ExternalLink

If the source attribute will always contain a URL (or a URN) then it SHOULD be mapped to an `ExternalLink`. For an overview of `ExternalLink` see section 2.3.7.

For example, the `rim.Person.homepage` attribute, if not null, always contain the URL for the Person’s homepage. It SHOULD therefore be mapped to an `ExternalLink` as shown below.

Note that an `ExternalLink` MUST be related to a `RegistryObject` using an `Association` instance in [ebRIM]. This allows the same `ExternalLink` to be shared by many `RegistryObject` instances.

```xml
<rim:Person id=${MARIECURIE_PERSON_ID}
  lid="urn:pim:MarieCurie"/>

<rim:ExternalLink externalURI="http://www.aip.org/history/curie/
  id=${MARIECURIE_WEBSITE_EXTERNAL_LINK_ID}>
<rim:Association
  id=${MARIECURIE_HOMEPAGE_EXTERNALLYLINKS_ASSOCIATION_ID}
  associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}
  sourceObject=${MARIECURIE_WEBSITE_EXTERNAL_LINK_ID}
  targetObject=${MARIECURIE_PERSON_ID}/>
```

Listing 17: Example of Mapping to ExternalLink

### 3.6.5 Direct Mapping to ebRIM Attribute

In some cases an attribute in the source model class may closely match an attribute in the [ebRIM] class. This is the most direct and preferred attribute mapping.

For example the `Person` class in PIM has an attribute “phone” (referred to as `pim.Person.phone`) whose semantics closely match the attribute “telephoneNumbers” in the `Person` class in [ebRIM] (referred to as `rim.Person.telephoneNumbers`). Thus it is preferred that the `pim.Person.phone` attribute is mapped to `rim.Person.telephoneNumbers`. Impedance mismatches between the source attribute data type and target attribute data type MAY be handled by the mapper using domain specific knowledge. For example the `pim.Person.phone` attribute is of datatype `String` while the `rim.Person.telephoneNumbers` attribute is of datatype `TelephoneNumber` where `TelephoneName` consists of several `String` attributes:

- “areaCode”
- “countryCode”
- “number”

Thus the mapper MUST choose which `rim.TelephoneNumber` attribute the `pim.Person.phone` attribute maps to. As an example they MAY chose to map it the `rim.TelephoneNumber.number` attribute. Alternatively, they may define a domain specific algorithm for splitting the `pim.Person.phone` attribute into one, two or three components that map to the various `TelephoneNumber` attributes in a deterministic manner.

### 3.6.6 Mapping to Slot

When all other options for mapping the source attribute are inadequate then the attribute MUST be mapped to a Slot.

#### 3.6.6.1 Mapping to rim.Slot.slotName

The source attribute name SHOULD be mapped to the `rim.Slot.slotName` attribute. To prevent name conflicts the mapping SHOULD define a mapping algorithm that generates a URN with the source
attribute name as its last component. It is also suggested that the source class name be the second last
component of the URN.

For example, the pim.Person.profession attribute SHOULD be mapped to a URN like:

```xml
<rim:Person id=${MARIECURIE_PERSON_ID}
  lid="urn:pim:MarieCurie">
  <rim:Slot name="urn:pim:Person:profession">
    ...
  </rim:Slot>
  ...
</rim:Person>
```

Listing 18: Example of Mapping pim.Person.Profession to slotName

### 3.6.6.2 Mapping to rim.Slot.slotType

The rim.Slot.slotType attribute value SHOULD be defined so it conveys the datatype semantics of the Slot
value. The value of the rim.Slot.slotType attribute MUST be the lid attribute value of a ClassificationNode
in the canonical DataType ClassificationScheme.

For example, the data type of the pim.Person.profession in PIM is String. It MUST therefore be mapped to
the rim.Slot.slotType value of:

```xml
<rim:Person id=${MARIECURIE_PERSON_ID}
  lid="urn:pim:MarieCurie">
  <rim:Slot name="urn:pim:Person:profession"
    slotType="urn:oasis:names:ebXML-regrep:DataType:String">
    ...
  </rim:Slot>
  ...
</rim:Person>
```

Listing 19: Example of Mapping DataType to slotType

Note that if the datatype happens to be a Collection then the slotType should reflect the data type of the
Collection elements. In case of a heterogeneous Collection the most specific data type from the DataType
ClassificationScheme MUST be used.

### 3.6.6.3 Mapping to rim.Slot.values

The rim.Slot.values (ValueList in XML Schema) SHOULD be defined as follows:

- If the value is a reference (datatype/slotType is urn:oasis:names:ebXML-regrep:DataType:ObjectRef) to another RegistryObject then the value MUST be the value
  of the id attribute of the RegistryObject being referenced.
- If the datatype of the source attribute is not a Collection then there should only be a single
  “rim:Value” within the ValueList.
- If the datatype of the source attribute is a Collection then there MAY be a multiple
  “rim:Value” within the ValueList.

The following example shows how the pim.Person.profession attribute is specified when mapping a
pim.Person instance to a rim.Person instance.

```xml
<rim:Person id=${MARIECURIE_PERSON_ID}
  lid="urn:pim:MarieCurie">
  <rim:Slot name="urn:pim:Person:profession"
    slotType="urn:oasis:names:ebXML-regrep:DataType:String">
    <rim:ValueList>
      <rim:Value>Scientist</rim:Value>
    </rim:ValueList>
  </rim:Slot>
  ...
</rim:Person>
```
3.7 Enumerated Type Mapping

A source attribute whose datatype is an Enumeration class SHOULD be mapped to a Classification on the target RegistryObject. An example of this has been provided with the mapping of the pim.Person.gender attribute in section 3.6.3.
4 Using ClassificationSchemes

The ebXML Registry provides a powerful, simple and flexible capability to create, extend and apply
taxonomies to address a wide set of use cases. A taxonomy in ebRIM is called a ClassificationScheme.
The allowed values in a ClassificationScheme are represented by ClassificationNode instances within
 ebRIM.

![Figure 9: Geography ClassificationScheme Example](image)

Figure 9 shows a geography ClassificationScheme. It is a hierarchical tree structure where the root of the
tree "iso-ch:3166:1999" is the name of the ClassificationScheme while the rest of the nodes in the tree are
ClassificationNodes.

Note that most ebXML Registry implementations [IMPL] provide a GUI tool to create and manage
ClassificationSchemes graphically.

4.1 Use Cases for ClassificationSchemes

The following are some of the many use cases for ClassificationSchemes in an ebXML Registry:

- Used to classify RegistryObjects to facilitate discovery based upon that classification. This is the primary role of ClassificationSchemes in ebXML Registry.
- Used to define all possible values of an Enumeration class. For example, the pim.Gender class is represented in ebRIM as a Gender ClassificationScheme.
- Used to define the datatypes supported by a registry (DataType scheme).
- Used to define the Classes supported by a registry (ObjectType scheme).
- Used to define the association types supported by the registry (AssociationType scheme).
- Used to define the security roles that may be defined for users of the registry (SubjectRole scheme).
- Used to define the security groups that may be defined for users of the registry (SubjectGroup scheme).

4.2 Canonical ClassificationSchemes

There are several ClassificationSchemes that are specified by ebRIM and required to be present in every
ebXML Registry. Such standard ClassificationSchemes are referred to as “canonical” ClassificationSchemes.

An ebXML Registry user MAY extend existing canonical ClassificationSchemes or add new domain
specific ClassificationSchemes. However, they cannot update/delete the existing canonical
4.3  Extending ClassificationSchemes

A registry user MAY extend an existing ClassificationScheme regardless of whether it is a canonical
scheme or a user defined scheme as long as the Access Control Policies for the scheme and its nodes
allow the user that privilege. The user may extend an existing scheme by submitting new
ClassificationNodes to the registry that reference existing ClassificationNodes or an existing
ClassificationScheme as the value of their “parent” attribute. The user SHOULD assign a logical id (lid) to
all user defined ClassificationNodes for ease of identification.

4.3.1  Use Cases for Extending ClassificationSchemes

The following are some of the most common use cases for extending ClassificationSchemes:

• Extending the ObjectType scheme to define new Classes supported by a registry. Listing 10 shows an
  example of extending the ObjectType scheme.

• Extending the AssociationType scheme to define the association types supported by the registry.
  Listing 11 shows an example of extending the AssociationType scheme.

• Extending the SubjectRole scheme to define the security roles that may be defined for users of the
  registry.

4.4  Defining New ClassificationSchemes

A user may submit an entirely new ClassificationScheme to the registry. Often the scheme is a domain
specific scheme for a specialized purpose. When mapping a domain specific model there are many
situations where a new ClassificationScheme needs to be defined.
5 PIM UML class diagram mapping to [ebRIM]

Following the defined rules, above in this document, in this section is detailed the whole mapping for the PIM UML model (see Figure 1) to [ebRIM].

5.1.1 PIM Class Mapping

In this section is defined the mapping of the PIM UML classes to [ebRIM] as showed in section 3.1. The table below summarizes the corresponding binding. It is possible to see that: only the PIM Person class is mapped to an already existing canonical registry object, the PIM Gender class is considered as a classification, so it doesn’t appear within the table; all other classes rise up a new sub-node of the [ebRIM] ExtrinsicObject canonical classification node and classes inheritance is respected within the registry using the registry classification tree as illustrated in section 3.3.

<table>
<thead>
<tr>
<th>Source Concept</th>
<th>ebRIM Object Type Name</th>
<th>ebRIM Parent Classification Node Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIM</td>
<td>PIM</td>
<td>ExtrinsicObject</td>
<td>This instance of ClassificationNode, sub-node of the ExtrinsicObject object type, groups all object type for the domain.</td>
</tr>
<tr>
<td>Person</td>
<td>Person</td>
<td>RegistryObject</td>
<td>This is the canonical [ebRIM] Person object type</td>
</tr>
<tr>
<td>LifeEvent</td>
<td>LifeEvent</td>
<td>PIM</td>
<td>New ClassificationNode object type, sub-node of PIM, represents the LifeEvent PIM UML class</td>
</tr>
<tr>
<td>BirthEvent</td>
<td>BirthEvent</td>
<td>LifeEvent</td>
<td>New ClassificationNode object type, sub-node of LifeEvent, represents the BirthEvent PIM UML class</td>
</tr>
<tr>
<td>MarriageEvent</td>
<td>MarriageEvent</td>
<td>LifeEvent</td>
<td>New ClassificationNode object type, sub-node of LifeEvent, represents the MarriageEvent PIM UML class</td>
</tr>
<tr>
<td>BirthingEvent</td>
<td>BirthingEvent</td>
<td>LifeEvent</td>
<td>New ClassificationNode object type, sub-node of LifeEvent, represents the BirthingEvent PIM UML class</td>
</tr>
<tr>
<td>DeathEvent</td>
<td>DeathEvent</td>
<td>LifeEvent</td>
<td>New ClassificationNode object type, sub-node of LifeEvent, represents the DeathEvent PIM UML class</td>
</tr>
<tr>
<td>Place</td>
<td>Place</td>
<td>PIM</td>
<td>New ClassificationNode object type, sub-node of PIM, represents the Place PIM UML class</td>
</tr>
<tr>
<td>PhysicalTraits</td>
<td>PhysicalTraits</td>
<td>PIM</td>
<td>New ClassificationNode object type, sub-node of LifeEvent, represents the PhysicalTraits PIM UML class</td>
</tr>
</tbody>
</table>

Table 1: [ebRIM ] object type definition and mapping for the PIM source UML model class diagram

5.1.2 PIM attributes mapping

In this section are applied the rules for attributes mapping defined in section 3.6 to obtain the whole mapping of PIM classes’ attributes. Within the PIM model there are some classes’ attributes that represent the corresponding association between classes (ex.: birth attribute of type BirthEvent for Person is a reference to the association between Person and BirthEvent classes). These attributes could be mapped to [ebRIM] as slots of type objectRef that reference the corresponding instance of the registry class as done in table 2. Equivalently these attributes can be mapped to [ebRIM] as associations. To maintain both, associations and attributes within the registry couldn’t be the better solution because it asks a double management of the same concept, the relationship within two registry instances. From the registry point of view using the associations should be better for different reasons. First, the query manager interface provides a set of dedicated association queries that could be useful for the registry discovery process that slot doesn’t have. Second, the integrity of registry object relationships can be better managed with associations then slots. In any case, implementer of registry applications can choose the better solution that they prefer, depending on the implementation issues.

The whole list of attributes that can be considered as slots at the same time that association is: birth, marriage, death, birthFather, birthMother, birthings, spouse and children attributes for PIM Person class;
location and babys attributes for BirthingEvent class; location, husband and wife for MarriageEvent class; location attribute for DeathEvent class and; location attribute for BirthEvent class.

Within the PIM UML class diagram the LifeEvent class generalizes BirthingEvent, MarriageEvent, BirthEvent and DeathEvent classes and for registry stored content purpose, it can be considered as an abstract class that never is used directly for storing registry content information. For this reason in the attributes mapping it isn't considered and its attributes and associations are directly mapped to sub-classes.

The table below summarizes the whole mapping for all PIM classes.

In this table slots attributes are defined as follow: Slot(name, type,'value').

- Name is the corresponding name for the slot (ex: urn:pim:Person:profession);
- Type is one of the admitted [ebRIM] types for a slotType as defined in [ebRIM];
- Value can be a list of admitted values for this attribute that SHOULD be verified by the registry content validation service at the submission or 'any value' if no constraints exists for the attribute. (for example an attribute can admit only 'Yes' or 'No' values)

<table>
<thead>
<tr>
<th>Source Attribute</th>
<th>[ebRIM] Attribute</th>
<th>Cardinality</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source class: Person</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>personName</td>
<td>1..1</td>
<td></td>
</tr>
<tr>
<td>homepage</td>
<td>externalLink</td>
<td>0..1</td>
<td></td>
</tr>
<tr>
<td>nationalId</td>
<td>ExternalIdentifier</td>
<td>0..1</td>
<td>&quot;NationalIdentifierScheme&quot; ClassificationScheme as identificationSchema</td>
</tr>
<tr>
<td>profession</td>
<td>Slot(urn:pim:Person:profession, urn:pim:Person:birth,urn:oasis:names:tc:ebxml-regrep:DataType:String, 'any value')</td>
<td>0..*</td>
<td>Profession attribute can be mapped as a slot or if an enumeration list is provided it can be mapped to the classification attribute.</td>
</tr>
<tr>
<td>gender</td>
<td>Classification</td>
<td>1..1</td>
<td>Referring to &quot;Gender&quot; ClassificationScheme</td>
</tr>
<tr>
<td>birth</td>
<td>Slot(urn:pim:Person:birth,urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID)</td>
<td>1..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>marriage</td>
<td>Slot(urn:pim:Person:marriage,urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID)</td>
<td>0..*</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>death</td>
<td>Slot(urn:pim:Person:death,urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID)</td>
<td>0..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>birthFather</td>
<td>Slot(urn:pim:Person:birthFather,urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID)</td>
<td>1..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>birthMother</td>
<td>Slot(urn:pim:Person:birthMother,urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID)</td>
<td>1..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>physicalTraits</td>
<td>Slot(urn:pim:Person:physicalTraits,urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID)</td>
<td>1..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>birthings</td>
<td>Slot(urn:pim:Person:birthings,urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID)</td>
<td>1..2</td>
<td>Alternatively this attribute can be...</td>
</tr>
<tr>
<td>Source Attribute</td>
<td>[ebRIM] Attribute</td>
<td>Cardinality</td>
<td>Comment</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>s:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID</td>
<td>spouse Slot(urn:pim:Person:spouse, urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID)</td>
<td>0..*</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>s:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID</td>
<td>children Slot(urn:pim:Person:children, urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, $birth_instance_ID)</td>
<td>0..*</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>Source class: BirthingEvent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>timeStamp Slot(urn:pim:BirthingEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:DateTime, 'any date time')</td>
<td>0..1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>location Slot(urn:pim:BirthingEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, '$location_id')</td>
<td>1..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td></td>
<td>babys Slot(urn:pim:BirthingEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, '$person_ids')</td>
<td>0..*</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>Source class: MarriageEvent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>timeStamp Slot(urn:pim:MarriageEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:DateTime, 'any date time')</td>
<td>0..1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>location Slot(urn:pim:MarriageEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, '$location_id')</td>
<td>1..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td></td>
<td>husband Slot(urn:pim:MarriageEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, '$person_id')</td>
<td>1..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td></td>
<td>wife Slot(urn:pim:MarriageEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, '$person_id')</td>
<td>1..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td>Source class: DeathEvent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>timeStamp Slot(urn:pim:DeathEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:DateTime, 'any date time')</td>
<td>0..1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>location Slot(urn:pim:DeathEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef, '$location_id')</td>
<td>1..1</td>
<td>Alternatively this attribute can be mapped to [ebRIM] as registry association.</td>
</tr>
<tr>
<td></td>
<td>causeOfDeath Slot(urn:pim:DeathEvent:causeOfDeath, urn:oasis:names:tc:ebxml-regrep:DataType:String, 'any string')</td>
<td>1..1</td>
<td></td>
</tr>
<tr>
<td>Source class: BirthEvent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>timeStamp Slot(urn:pim:BirthEvent:timeStamp, urn:oasis:names:tc:ebxml-regrep:DataType:DateTime, 'any date time')</td>
<td>0..1</td>
<td></td>
</tr>
</tbody>
</table>
When user submits content to the registry, the registry itself doesn't provide any feature for controlling that object structure, semantic, cardinalities and associations defined within the original model are respected. For that, implementers can develop in an application manner or, when possible, using the registry Content Management Service and define new registry content validation and cataloging services to supply this "lack". For further information on this stuff, readers can refer itself directly to the [ebRS] specification.

### 5.2 PIM Association mapping

In this section is defined the whole mapping of PIM associations to [ebRIM] applying the general rules illustrated in section 3.5.

In the mapping the *LifeEvent* super-class has been not considered because, as already explained in the previews section, it doesn't carry any registry content itself. It is always an instance of its sub-classes. The generalization association is not considered from the registry point of view as an association but it reflects the already defined inheritance between registry object types as defined in the PIM Class Mapping section.

Table 3 shows the whole mapping between the source UML class diagram model PIM and [ebRIM].

<table>
<thead>
<tr>
<th>Association Source Object Type</th>
<th>Association Target Object Type</th>
<th>[ebRIM] Association Type</th>
<th>[ebRIM] Association Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>BirthEvent</td>
<td>Birthings</td>
<td>Birthings</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td>MarriageEvent</td>
<td>Marriages</td>
<td>Marriages</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td>DeathEvent</td>
<td>Death</td>
<td>Death</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td>BirthEvent</td>
<td>Birth</td>
<td>Birth</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td>Person</td>
<td>Child</td>
<td>Child</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td>Person</td>
<td>BirthFather</td>
<td>BirthFather</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td>Person</td>
<td>BirthMother</td>
<td>BirthMother</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td>Person</td>
<td>Spouse</td>
<td>Spouse</td>
<td></td>
</tr>
<tr>
<td>MarriageEvent</td>
<td>Person</td>
<td>Husband</td>
<td>Husband</td>
<td>Sub node of Spouse</td>
</tr>
<tr>
<td>MarriageEvent</td>
<td>Person</td>
<td>Wife</td>
<td>Wife</td>
<td>Sub node of Spouse</td>
</tr>
<tr>
<td>BirthEvent</td>
<td>Person</td>
<td>Babys</td>
<td>Babys</td>
<td></td>
</tr>
<tr>
<td>BirthEvent</td>
<td>Place</td>
<td>Location</td>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>MarriageEvent</td>
<td>Place</td>
<td>Location</td>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>DeathEvent</td>
<td>Place</td>
<td>Location</td>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>BirthEvent</td>
<td>Place</td>
<td>Location</td>
<td>Location</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: PIM association mapping to [ebRIM]
### 5.2.1 PIM compositions mapping

As illustrated in section 3.5.5 composition can be mapped to [ebRIM] as slots.

But as already explained in the 5.1.2 section, implementers can consider composition as simple associations, or, for 1 to 1 compositions attributes of the contained class can be integrated directly as attributes of the container class.

In any case the table below lists the composition mapping applying the rule illustrated in this tutorial.

<table>
<thead>
<tr>
<th>Composition Source Object</th>
<th>Composition Target Object</th>
<th>ebRIM Slot Name</th>
<th>ebRIM Slot Type</th>
<th>ebRIM Slot Value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>PhysicalTraits</td>
<td>PhysicalTraits</td>
<td>urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef</td>
<td>$PhysicalTraits_ID</td>
</tr>
</tbody>
</table>

Only one value. (this slot is the same that the PhysicalTraits attribute)

Table 4: PIM composition mapping to [ebRIM]

### 5.3 PIM classifications mapping

The registry offers the possibility to define multiple classifications for registry object instances. This feature is really useful for grouping registry objects within a same taxonomy.

For that implementers must estimate from the source model all artifacts that are sensibles to classify the registry content. This task doesn't follow a unique way for obtaining all artifacts that could become classification, but it depends on the implementation features.

For example for the PIM model, Gender class and NationalId can be considered as classifications as well as Profession attribute, that in some cases, could be a possible candidate for a registry classification.

A general rule for defining a new registry classification could be to consider all artifacts that can be obtained from a common and sharable set of values amongst the object instances.

Following this reasoning not only Gender and NationalId, but also Person profession attribute and PhysicalTraits eyeColor and hairColor could be mapped to the [ebRIM] as new classifications.

Table 5 shows the new classification schemes that could be created on the registry to classify PIM Person registry objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Reference</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td></td>
<td>This Class provides a classification for Person.Gender. All instances of this classification (Male, Female,...) are sub-node elements of Gender.</td>
</tr>
<tr>
<td>NationalIdentifierScheme</td>
<td><a href="http://www.nationalidentifier.org/list.xml">http://www.nationalidentifier.org/list.xml</a></td>
<td>ClassificationScheme used by person:nationalId external identifier attribute.</td>
</tr>
</tbody>
</table>

Table 5: Classification mapping for the source model PIM concepts
Known Issues

These generic mapping patterns should be formalized via RIM artifacts and stored in the registry.

- UML cardinality needs to be expressed generically, like for Slots, Associations, …
- Expanding RIM ObjectType hierarchy beyond ExtrinsicObject subtree
- Objective criteria for when to use ObjectRefs vs. Values, like "NameAsRole" could refer to something like RoleTaxonomy instead of using value of UML role.
- Aggregation and Composition are Association in UML. There mapping to ebRIM is inconsistent.
- A rule for “Association classes” should be added
- Need to give example of mapping an Association class (e.g. MarriageEvent)
Appendix A - PIM to ebRIM: The Complete Mapping

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-regrep:xsd:query:3.0"
xmlns:query21="urn:oasis:names:tc:ebxml-regrep:xsd:query:3.0"
xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
xmlns:rim30="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
  <RegistryObjectList>
    <!-- ######################################################## -->
    <!-- ### Specifics ObjectType extensions                   -->
    <!-- ### Sub-nodes of ExtrinsicObject ClassificationScheme -->
    <!-- ######################################################## -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject" code="PIM"
    id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM">
      <!-- ObjectType for LifeEvent -->
      <ClassificationNode code="LifeEvent"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent">
        <!-- ObjectType for BirthEvent -->
        <ClassificationNode code="BirthEvent"
    id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:BirthEvent">
          <!-- ObjectType for MarriageEvent -->
          <ClassificationNode code="MarriageEvent"
    id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:MarriageEvent">
            <!-- ObjectType for BirthingEvent -->
            <ClassificationNode code="BirthingEvent"
    id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:BirthingEvent">
              <!-- ObjectType for DeathEvent -->
              <ClassificationNode code="DeathEvent"
    id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:DeathEvent">
                <!-- ObjectType for Place -->
                <ClassificationNode code="Place"
    id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:Place">
                  <!-- ObjectType for PhysicalTraits -->
                  <ClassificationNode code="PhysicalTraits"
    id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:PhysicalTraits">
    </ClassificationNode>
      </ClassificationNode>
    </ClassificationNode>
  </ClassificationNode>
</RegistryObjectList>
</SubmitObjectsRequest>
```
<ClassificationNode code="PhysicalTraits"
    lid="urn:oasis:names:tc:ebxml-
    regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:PhysicalTr
    aits" id="urn:oasis:names:tc:ebxml-
    regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:PhysicalTr
    aits"/>
</ClassificationNode>

<!-- Specifics AssociationType extensions -->
<!-- Sub-nodes of AssociationType ClassificationScheme -->

<!-- AssociationType for Birthing -->
<ClassificationNode parent="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType"
    lid="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Birthing" code="Birthing"
    id="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Birthing"/>

<!-- AssociationType for Baby -->
<ClassificationNode parent="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Birthing"
    lid="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Baby" code="Baby"
    id="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Baby"/>

<!-- AssociationType for Spouse -->
<ClassificationNode parent="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Spouse"
    lid="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Husband" code="Husband"
    id="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Husband"/>

<!-- AssociationType for Wife -->
<ClassificationNode parent="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Wife" code="Wife"
    id="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Wife"/>

<!-- AssociationType for Marriage -->
<ClassificationNode parent="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Marriage"
    lid="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Marriage" code="Marriage"
    id="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Marriage"/>

<!-- AssociationType for Death -->
<ClassificationNode parent="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Death" code="Death"
    id="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:AssociationType:Death"/>

<!-- AssociationType for Birth -->
```
  <ClassificationScheme lid="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:NationalIdentifierScheme"
    id="urn:oasis:names:tc:ebxml-
    regrep:classificationScheme:NationalIdentifierScheme" isInternal="true"
    objectType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode"
    regrep:ObjectType=RegistryObject:ClassificationScheme">
    <Name>
      <LocalizedString charset="UTF-8" xml:lang="en-US"
        value="NationalIdentifierScheme"/>
    </Name>
    <Description>
      <LocalizedString charset="UTF-8" xml:lang="en-US"
        value="Defines the NationalIdentifierScheme taxonomy."/>
    </Description>
  </ClassificationScheme>
```

Listing 21: PIM source model mapping to [ebRIM]
## Appendix F - Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>By Whom</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>September 22, 2004</td>
<td>Farrukh Najmi, Nikola Stojanovic</td>
<td>Initial version with core mapping pattern for input from CCTS mappers.</td>
</tr>
<tr>
<td>0.2</td>
<td>September 23, 2004</td>
<td>Farrukh Najmi, Nikola Stojanovic</td>
<td>Minor bug fixes.</td>
</tr>
<tr>
<td>0.3</td>
<td>September 24, 2004</td>
<td>Farrukh Najmi, Nikola Stojanovic</td>
<td>Added some content to chapters 4-8.</td>
</tr>
<tr>
<td>0.3</td>
<td>September 29, 2004</td>
<td>Farrukh Najmi, Nikola Stojanovic</td>
<td>Minor fixes based upon feedback from initial reviewers.</td>
</tr>
<tr>
<td>0.5</td>
<td>Avril 15, 2005</td>
<td>Ivan Bedini</td>
<td>Updated to version [ebRIM] v3.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Changed file format</td>
</tr>
<tr>
<td>0.6</td>
<td>January 2006</td>
<td>Ivan Bedini</td>
<td>Changed title and purpose of the document</td>
</tr>
</tbody>
</table>
Appendix G - References

Appendix H - Normative
[ebRIM] ebXML Registry Information Model version 3.0
http://www.oasis-open.org/committees/regrep/documents/3.0/specs/ebRIM.pdf

[ebRS] ebXML Registry Services Specification version 3.0
http://www.oasisopen.org/committees/regrep/documents/3.0/specs/ebRS.pdf

[UML] Unified Modeling Language version 1.5
http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf

Appendix I - Informative
[CMRR] Web Content Management Using OASIS ebXML Registry

[IMPL] ebXML Registry 3.0 Implementations
freebXML Registry: A royalty free, open source ebXML Registry Implementation
http://ebxmlrr.sourceforge.net
Need other implementations listed here??

[TUT] UML Tutorials
Borland Tutorial
http://bdbn.borland.com/article/0,1410,31863,00.html
Sparx Systems UML Tutorial