UML to XSD

Expressing object oriented concepts in the eXtensible Markup Language

Purpose

While data exchange and web service protocols such as XML and WSDL have an important role to play in enabling web services, they rely on the client and server sharing a common understanding of the underlying business domain. This common understanding is often expressed in UML. Indeed, the Unified Modeling Language is increasingly used to express business processes in an industry-neutral manner.

In a preliminary phase, a high-level analysis using use-case diagrams is made. The business analyst then drills down to object diagrams, defines the collaboration relationships between specific objects, and identifies individual classes. Producing a class diagram is the final step in this particular context, as it offers a starting point from which the data model can be translated into XML quite easily. The rules for this transformation are described in the XMI standard, a subject treated in another paper entitled 'XMI explained'.

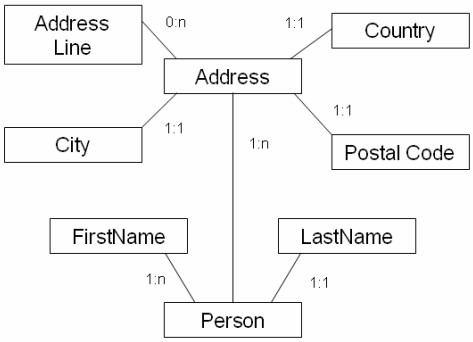
UML and schemas

In XML, data models can be expressed by using DTDs and schemas. Because of the limitations of DTDs, schema's are best suited to reflect data modeled with UML. DTDs do offer limited functionality to express associations between classes. By using ID and IDREF(S) you can create one-to-one and one-to-many relationships between elements in the XML file. However, to express the complex relationships that UML can model, more elaborated mechanisms are needed. Fortunately, many of these mechanisms are available in the schema language.

In the following paragraphs, we'll focus on schemas, and how we can use them to express relationships such as compositions and generalization that exist in a UML model. However, even though schemas do a great job in porting the aforementioned concepts to XML, some object oriented techniques such as multiple inheritance cannot be represented in the current version of schemas.

Data Models

A data model represents the relationships and multiplicity between the different classes that you may have defined for your particular business process. Classes give an abstract representation of a real-life object. A class diagram is a static diagram, and thus the information contained in the class diagram is valid in every stage of the business process's' execution. The following class diagram represents the different entities that we could define to represent a person-address relationship.



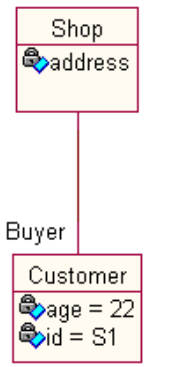
If we needed to add more information to our data model, we could choose to add this information as new classes or as attributes. For example, a country code could arguably be added as an element in its own right, or as an attribute to the country element. In this particular case we would surely opt for an attribute. Keep in mind that the data model should be as simple as feasible, keeping the number of classes to a strict minimum. This of course can prove to be a challenging task when modeling complex systems such as a business process.

Mapping a class

A class and its attributes can easily be mapped to schema elements. The class itself is mapped to a complex element by default, and the attributes can be mapped to individual attributes, or to an enumeration in the schema. Attributes in UML have no particular order, so the schema enumeration element is ideally suited to represent attributes.

Associations

Relationships and dependencies between classes are indicated by using associations, compositions and aggregations. Let's start with a simple association between classes: as shown in the following illustration, we can use an element or an attribute to express the UML association in a schema:

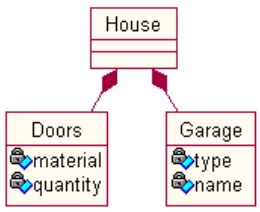



```
<!--as an element -->
<Shop>
  <Shop.Buyer>
     <Customer age="22" idref="S1"/>
  </Shop.Buyer>
</Shop>

<!--as an attribute -->
<Shop address="Bond Street 4" Buyer="S1">
</Shop>

<!-- Customer class -->
<Customer id="S1">
  <Customer.age>22</Customer.age>
</Customer>
```

As it is done in a DTD, the schema also uses ID and IDREF statements to indicate the link between the classes when expressed as elements. When an attribute is used to indicate the link, the association role name (Buyer) is used to create the link.

Compositions

Aggregations are treated just as associations; because aggregations 'own' their elements by reference, preferably ID and IDREF pairs will be used. (An example aggregation could be a car: a car is composed of an engine, wheels, etc…) But an aggregation can exist without the elements that it refers to. The same is not true for a composition! Compositions 'own' their elements by value:

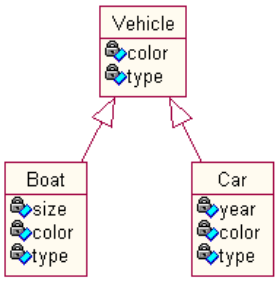



```
<House>
  <Doors quantity="6">
    <Doors.material>untreated wood</Doors.material>
  </Doors>

  <Garage type="20A">
    <Garage.name>Modular automated</Garage.name>
  </Garage>
</House>
```

As soon as the top level element of the composition is unavailable (in this example: House), all the owned elements are also unavailable. Compositions can easily be expressed in an XML file because XML elements also 'own' their child elements by value. (In this case, House is a container element for Door and Garage -House 'owns' Door and Garage.) One important thing to notice here is the way in which attributes are mapped to the schema: attributes containing string information are usually mapped to elements, because string data can become quite large rendering the XML file difficult to read.

Inheritance

Finally, there is the concept of inheritance that we would like to see expressed in a schema. Inheritance is used for example in a generalization where the sub-classes inherit features of the super-class, such as attributes and functions. When porting UML to XML, the so-called copy-down inheritance is used, meaning that all the attributes and functions of the parent class are available ('copied down') to the child class.



The important concept to grasp here is the fact that for example Boat and Car can both be used where Vehicle is allowed to be used. DTDs can use parameter entities to represent inheritance, and with schemas we take advantage of substitution groups. Multiple inheritance, where one class inherits features from more than one super class, cannot be expressed in a schema at this time.

Conclusions

Expressing UML models in XML is becoming increasingly important as companies want to communicate part of their business process through electronic messages and web services. Currently, schemas do a great job in expressing UML concepts although, as we've seen, some restrictions still exist. Rest assured though, that many shortcomings will be tackled in the upcoming version of schemas.

Genuine
Ganzendreef 12
3140 Keerbergen
Belgium

Tel. 0032 (0)2 569 79 66
Fax 0032 (0)2 569 06 36

info@genuine.be
www.genuine.be