UML and XML

G. Falquet
L. Nerima

Contents

Using UML to model documents

Representing UML models in XML (XMI)

Modeling Idea

Model the application domain with UML classes and associations

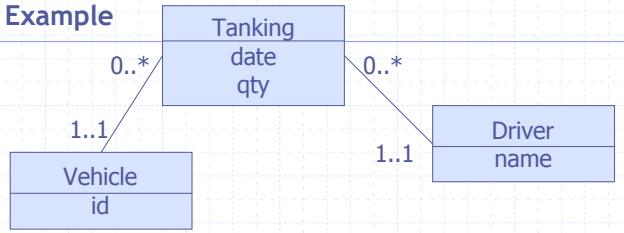
- UML has powerful modelling abstractions
- Class schemas may already exist
- etc.

Transform the UML class schema into an XML DTD

Transformation technique

1. define an element type for each class
2. define an attribute for each UML attribute
3. define the element content by
   - following the associations, starting from any class
   - respecting the multiplicities

Example



Vehicle □ Tanking □ Driver

```
<!ELEMENT vehicle (tanking*)>
<!ELEMENT tanking (driver)>
...
```


Example (cont.)

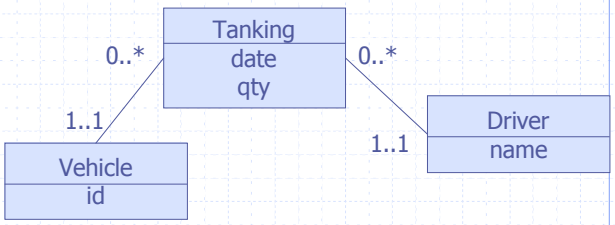
```
<!ELEMENT consumption-checking (vehicle*)>

<!ELEMENT vehicle (tanking*)>
<!ATTLIST vehicle id CDATA #REQUIRED>

<!ELEMENT tanking (driver)>
<!ATTLIST tanking date CDATA #REQUIRED>
<!ATTLIST tanking qty CDATA #REQUIRED>

<!ELEMENT driver EMPTY>
<!ATTLIST driver name CDATA #REQUIRED>
```


Example (data)

```
<vehicle id="v001">
   <tanking date="02-10-10" qty="45">
        <driver name="Bob"/>
   </tanking>
   <tanking date="02-10-10" qty="55">
        <driver name="Bob"/>
   </tanking>
   <tanking date="02-10-10" qty="65">
        <driver name="Bob"/>
   </tanking>
</vehicle>
```


Another solution

Tanking (□ Vehicle , □ Driver)




```
<!ELEMENT consumption-checking (tanking*)>
<!ELEMENT tanking (vehicle, driver)>
<!ELEMENT vehicle (EMPTY)>
<!ELEMENT driver (name, ...)>
```

Data

```
<tanking date="02-10-10" qty="45">
  <vehicle ID="v001"/>
  <driver name="Bob"/>
</tanking>

<tanking date="02-10-10" qty="55">
    <vehicle ID="v001"/>
    <driver name="Bob"/>
</tanking>

<tanking date="02-10-10" qty="65">
    <vehicle ID="v001"/>
    <driver name="Bob"/>
</tanking>
```


Redundancy in data

Suppose we have several attributes for <driver>

```
<tanking date="02-10-10" qty="45">
  <vehicle ID="v001"/>
  <driver name="Bob" famName="Stone" birthdate="1955" />
</tanking>

<tanking date="02-10-10" qty="55">
    <vehicle ID="v001"/>
    <driver name="Bob" famName="Stone" birthdate="1955" />
</tanking>
```

The same information is repeated twice (or more)

Because a *Driver* may participate in >1 *Tanking*

To avoid redundancy -- References

Attributes with type ID
- uniquely identifies an element
- all values must be distinct in a document
- only one ID attribute allowed in an element

Attributes with type IDREF
- refer to an element
- the referred element must exist in the document
- no precise type checking

Example - DTD

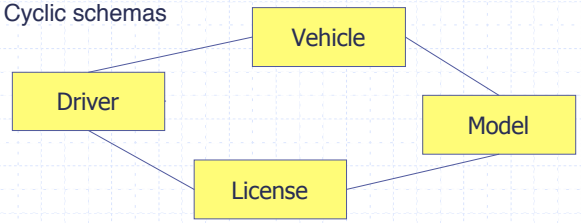
```
<!ELEMENT consumption-checking
                    (tanking*, vehicle*, driver*)>

<!ELEMENT tanking EMPTY>
        <!ATTLIST tanking veh IDREF #REQUIRED>
        <!ATTLIST tanking drvr IDREF #REQUIRED>
        <!ATTLIST tanking qty CDATA #REQUIRED>

<!ELEMENT vehicle (#PCDATA)>
        <!ATTLIST vehicle id ID #REQUIRED>

<!ELEMENT driver (#PCDATA)>
        <!ATTLIST driver id ID #REQUIRED>
```

Example - Document

```
<consumption-checking>

  <tanking veh="GE-001" drvr="bob" qty='33ltr'/>
  <tanking veh="GE-001" drvr="bob" qty='44ltr'/>
  <tanking veh="GE-001" drvr="bob" qty='55ltr'/>

  <vehicle id='GE-001'>Vehicle no. 1</vehicle>

  <driver id='bob'>Bob Schmied</driver>

</consumption-checking>
```


Another example

Cyclic schemas



To represent all the associations :

Vehicle (□ Driver □ License | □ Model □ License)

Document structure

```
<vehicle>
  <model>
      ...
      <license> L  </license>
  </model>
  <driver>
      ...
      <license>  L <!-- redundant --> </license>
      <license> ... </license>
  </driver>
</vehicle>
```


Resolving redundancy with references

```
<!ELEMENT at3 (vehicle I driver I model I license)*>

<!ELEMENT vehicle EMPTY>
        <!ATTLIST vehicle id ID #REQUIRED>
        <!ATTLIST vehicle driver IDREF #REQUIRED>
        <!ATTLIST vehicle model IDREF #REQUIRED>

<!ELEMENT driver (#PCDATA I has-license)*>
        <!ATTLIST driver id ID #REQUIRED>

<!ELEMENT has-license EMPTY>
        <!ATTLIST has-license lic IDREF #REQUIRED>
        <!ATTLIST has-license since CDATA #IMPLIED>
```

DTD (cont.)

```
<!ELEMENT model (#PCDATA)>
        <!ATTLIST model id ID #REQUIRED>
        <!ATTLIST model requires IDREF #REQUIRED>

<!ELEMENT license (#PCDATA)>
        <!ATTLIST license id ID #REQUIRED>
```


Data

```
<vehicle id="v001" driver="bob" model="m601"/>
<vehicle id="v002" driver="sam" model="m604"/>
<vehicle id="v004" driver="jack" model="m601"/>

<driver id="bob">
 This is Bob Smith (our best driver)
 <has-license lic="L1" since="1978" />
 <has-license lic="L2" since="1972" />
</driver>
<driver id="jack">
 Jack Du Pont, from London
 <has-license lic="L1" since="2001" />
</driver>
```


Data (cont.)

```
<model id="m601" requires="L2">
   Chrysler Voyager++ extra large with 200l tank
</model>
<model id="m604" requires="L1">
 Ford Fiesta
</model>

<license id="L2">
   Large cars, over 1600kg
</license>
<license id="L1">
   Small cars, under 1000kg
</license>
```


Problems with references

ID's must be assigned "by hand"

But references are not typed

- may refer to any existing element

Solutions

- XPointers : /my-site/my-doc#/../model[@type="truck"]
- XML Schemas
- ... or databases

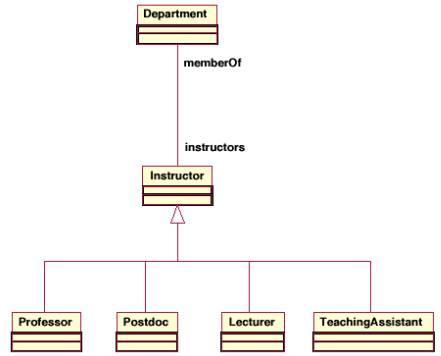
XMI

Goal: represent UML diagrams in XML

make UML specifications open and exchangeable

UML tools (diagram editors, code generators, pretty printers, …) should be able to read/write UML specs in XMI

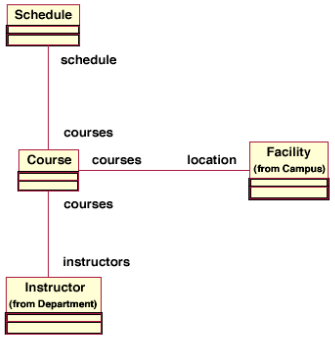
A Class Diagram



In XMI

```
<XMI.content>
<UML:Class name="Department" xmi.id="Department"/>
<UML:Class name="Instructor" xmi.id="Instructor"/>
<UML:Class name="Professor" xmi.id="Professor" generalization="Instructor"/>
<UML:Class name="Postdoc" xmi.id="Postdoc" generalization="Instructor"/>
<UML:Class name="Lecturer" xmi.id="Lecturer" generalization="Instructor"/>
<UML:Class name="TeachingAssistant" xmi.id="TeachingAssistant"
    generalization="Instructor"/>

<UML:Association>
    <UML:Association.connection>
        <UML:AssociationEnd name="instructors" type="Instructor"/>
        <UML:AssociationEnd name="memberOf" type="Department"/>
    </UML:Association.connection>
</UML:Association>
</XMI.content>
```


Another class diagram



Associations in XMI - external references

```
<XMI.header>
<XMI.model xmi.name="Schedule" href="Schedule.xml"/>
<XMI.metamodel xmi.name="UML" href="UML.xml"/>
<XMI.import name="Department" href="Department.xml"/>
<XMI.import name="Campus" href="Campus.xml"/>
</XMI.header>

<UML:Association>
    <UML:Association.connection>
            <UML:AssociationEnd name="instructors">
                    <UML:AssociationEnd.type>
                            <UML:Classifier href="Department.xml#Instructor"/>
                    </UML:AssociationEnd.type>
            </UML:AssociationEnd>
            <UML:AssociationEnd name="courses" type="Course"/>
    </UML:Association.connection>
</UML:Association>
```

I S I

 25