Identifying and Defining a Business Problem:

Assuming that your goal is to solve a real world business problem, you must already have some idea of the type of capabilities that your solution should provide. It takes very little to define a business problem once you are in dire need of a solution. For example, as businesses invest in globalizing their services to increase their revenue streams, a real world business problem is that of quickly providing companies with a global image. As general purpose automatic translation will certainly remain a dream for years to come, part of the globalization and localization efforts are conducted by human translators. Assuming you were a freelance technical translator, you would focus on providing quality translation services as part of the overall globalization process. As time is of the essence, the quality of the work you produce depends on the quality of the tools you use to help you achieve your translation work. Specifically, reliable glossary management is most definitely key to quality technical translations.

Today, translators use various ways to create glossaries mostly based on the use of document management tools such as word processors, spreadsheet editors, or other specialized glossary tools. Unfortunately, most of these tools lack interchange capabilities with other tools, and limit the horizons of glossary management capabilities to the capabilities of the tool being used to capture the glossaries. As some of the tools were not even designed with glossaries in mind in the first place, it is no surprise to discover that most translators are either too busy to care, think that good is good enough, or have simply given up dreaming about ideal solutions.

To summarize, an interesting real world business problem, relevant to translators in the globalization industry, is the creation of extensible glossary management tools. Instead of focusing on using an existing tool to create glossaries, your first concern should really be that of understanding the glossary information structure. Once you understand that structure it should be easy for you to select the best suited tool, if such tool exist, or build your own. This is really why, the XML application modeling process starts with the modeling of applications of XML required by the problem at hand.

Modeling Applications of XML:
The process of coming up with a suitable application of XML for a problem at hand includes the following steps:

- Documenting the information structure
- Representing the information structure in XML form
- Formally defining the information structure in XML form

The first step, referred to as **information modeling**, focuses on identifying the correct meaning for the information being modeled. As per the accepted definition, basic information components includes content, structure, and form. Understanding the meaning of information makes it possible to assign a name to a piece of content, and use that name to refer to it. As you come up with names for your content components you are effectively documenting your information structure. The second step in the application of XML modeling process consists of representing your information structure in XML form. This step requires a mapping of your content into one or more XML documents. XML documents are specific types of data objects which abide to XML rules of well-formedness. XML provides both physical and logical structuring capabilities. The physical structure lets you represent content in terms of storage units called entities, while the logical structure is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated by explicit markup and are ordered in a specific way to ensure well-formedness. The third step and last step in the application of XML modeling process formally specifies your information structure so that you can use an XML parser to enforce the validity of your XML documents with respect to the formal structure you have specified.

**Modeling Applications of XML v.s. Database Information Modeling:**

The process of modeling applications of XML clearly resembles that of data modeling, which is one of the activities in the overall process of database design. In fact the database design process is centered on conceptual, logical, and physical design phases. Similar to the first step of the application of XML modeling process, Conceptual Database Design is also an “information modeling” phase that focuses on identifying the correct meaning for the information being modeled. It uses a data modeling approach to assemble a conceptual schema for the information model of interest. Logical Database Design focuses on identifying a correct structure for representing the conceptual information model captured in the conceptual database design phase. For example, when a relational database is being used, Logical Database Design involves Normalization to come up with the best possible set of relational tables that could be used to capture the semantics (i.e. “the meaning”) expressed in the conceptual information model. The Logical Database Design phase produces an “external schema”, which is presented to the users and processing programs interested in using the data. The Logical Database Design phase is similar to the second and third steps in the modeling process of applications of XML. Finally, Physical Database Design focuses on improving the overall performance of data access using access strategies, pointers, and capabilities such as indexing, and partitioning. The Physical Database Design process produces the “internal schema” or the “physical
view” of the data. The internal view is not a concern for the users or the processing programs. In the three-schema architecture (i.e., “conceptual, logical, and physical schemas”) which result from the database design process, a database management system (DBMS) stands between the conceptual schema and the description of the data storage (internal schema). It is interesting to note that, at the difference of XML, database systems requires the definition of a formal structure for data (i.e., a “logical schema”), which takes away some of the flexibility provided by XML. Although XML provide physical structuring capabilities, the Physical Database Design phase does not have an equivalent in the XML world. This makes that databases still remains the most efficient way to store structured data. As a result, once more, you should view XML and database technologies as complementary. Mapping XML data to an underlying database is an important part of the modeling activity which is part of the modeling methodology for XML applications.

**XML Information Modeling « à la » UML :**

The following are UML use case and class diagrams that illustrate the use of UML to model applications of XML. Existing tools provides ways to extract a W3C XML Schema from a Rose model. You can also build a tool that uses XSLT to transform and extract an XML Schema from an XMI representation of a Rose model. The UML and related documents are illustrated below and are provided in electronic form in the “demo” area on the course Web site.

**Use Case View:**

![Use Case Diagram](image)

**Logical View:**

![Logical Diagram](image)
W3C Schema:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XML Spy v4.0 beta 1 build Jun 13 2001 (http://www.xmlspy.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="Glossaries">
    <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Glossary" type="GlossaryType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="GlossaryType">
    <xs:sequence>
      <xs:element name="topic" type="xs:string"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Entry">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="authorName" type="xs:string" minOccurs="0"/>
            <xs:element name="english" type="xs:string" minOccurs="0"/>
            <xs:element name="french" type="xs:string" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```
XML 1.0 DTD

<!ELEMENT Glossaries (Glossary)>
<!ELEMENT Glossary (topic, Entry*, description?, Glossary*)>
<!ATTLIST Glossary
  creationDate CDATA #IMPLIED>

<!ELEMENT topic (#PCDATA)>
<!ELEMENT Entry (authorName?, english?, french?, (frenchCanadian | dialectCode)*)>
<!ATTLIST Entry
  creationDate CDATA #IMPLIED>

<!ELEMENT description (#PCDATA | bold | italic | underline)>
<!ELEMENT authorName (#PCDATA)>
<!ELEMENT english (#PCDATA)>
<!ELEMENT french (#PCDATA)>
<!ELEMENT frenchCanadian (#PCDATA)>
<!ELEMENT dialectCode (#PCDATA)>
<!ELEMENT bold (#PCDATA)>
<!ELEMENT italic (#PCDATA)>
<!ELEMENT underline (#PCDATA)>

Sample XML File Using DTD:
<xml version="1.0"?>
<Glossaries>
  <Glossary creationDate="2001-06-15">
    <topic>Software Applications</topic>
    <Entry creationDate="2001-06-17">
      <english>Server</english>
      <french>Serveur</french>
    </Entry>
  </Glossary>
  <Glossary creationDate="2001-06-30">
    <topic>XML Applications</topic>
    <Entry creationDate="2001-06-30">
      <english>Language</english>
      <french>Langage</french>
    </Entry>
  </Glossary>
</Glossaries>

Sample XML File Using XML Schema:

<?xml version="1.0"?>
<Glossaries xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="C:\XMLBus\chapter3\uml\glossaries.xsd">
  <Glossary creationDate="2001-06-15">
    <topic>Software Applications</topic>
    <Entry creationDate="2001-06-17">
      <english>Server</english>
      <french>Serveur</french>
    </Entry>
  </Glossary>
  <Glossary creationDate="2001-06-30">
    <topic>XML Applications</topic>
    <Entry creationDate="2001-06-30">
      <english>Language</english>
      <french>Langage</french>
    </Entry>
  </Glossary>
</Glossaries>