Moving from Conceptual to Logical Data Models

This document outlines issues from the EuroRoadS workshop that show the difficulty of moving from a conceptual->logical model in an automated way. There are also a number of uml conceptual model suggestions that could result in a more direct transfer to an implementation model.

I started my logical model work from a piece of paper printed from the Rose model. I had a number of implementation decisions to make, and it’s usually easier to start from scratch rather than try to automate the creation of a logical data model.

Field Naming Conventions and Reserved Words

First, a very simple issue: People who use Oracle will have all class names and attribute names converted to uppercase in the database. Most designers implementing Oracle will want class and attribute names to have underscores separating the words in the names. For other database implementations people want to use mixed upper and lower case to separate the words. If the ISO uml standards use a different approach then everyone using databases will need to re-type attribute and or class names. I understand the reasoning for uml, xml, and programming purposes to distinguish between classes and attributes, but in the database world this looks peculiar. Regardless of opinions about how to solve this, the practical reality is that conceptual and logical models will look different for anyone implementing in a database environment.

Another issue for the logical model is SQL Reserved words. For instance, the EuroRoadS model has the attribute “level”, which is a reserved word. I had to change the name for storage in a relational database. We had the same issue in US content standards but I don’t think those issues were ever resolved because the technical spec writers didn’t seem to care.

Names, Attributes vs. Classes, and Uniqueness for Associations

In uml it is suggested that association names can be used more than one time, for instance "owns" or "uses" etc. For implementation, however, this gets a bit complicated because for storage in a relational database we needed to make the association names unique. Why? Because M:N associations are implemented as database tables. Also, the technique we use for modeling associations with attributes involves a separate class in the model/table in the database. So in going from the conceptual model we need to have named associations that are unique. When associations are inherited from higher-level abstract classes we either had to make up names or use an alternate approach in the logical model. What was implemented was that the logical model can only have associations between concrete classes. In the EuroRoad example that means we would draw the association between FerryAttribute and FerryLink at the leaf/concrete class level.

So for example we had:
Notice that the Associations are not named in the diagram, and that we would have multiple non-unique association names if the names were inherited down to the concrete classes.

In the diagram above, an association between ER_FerryLink and subclasses of ER_FerryLinkAttribute is implied. For implementation in a logical model we have 2 issues:

1. How to make the association in the logical diagram between concrete leaf classes.
2. How to actually store the attribute with the Ferry features. We can either make these attributes of the FerryLink class (a segmented model like we might see from TeleAtlas), or use linear referencing to locate the attributes along the FerryLink.

There is a third issue about whether a FerryLink attribute can apply to a RoadLink, but that’s more than a conceptual->logical issue so I won’t address that here. In general I will say that the precise list of attributes and content guidelines is more important than the convenient notation style shown here. While it’s ok for this workshop, more thought should be put into a clear conceptual model for EuroRoads.

In the following diagram I presumed that the attributes will become properties of the feature:
This approach works, but it does not allow for attribute identity and time periods as shown in the conceptual model. To handle this and the linear referencing approach, the logical model would like like:
Notice that several attributes are added to a new Object Class called ER_RoadnetAttributeClass that is used for inheritance into each of the Attributex classes. In addition, in the Geodatabase model Measures on the roads would be implemented as part of the geometry model through a tagged value on the FerryLink class (HasM = True). Z elevations are handled in a similar way.

**Multiple Implementation Models from a Single Conceptual Model**

If you follow both of these approaches to their logical conclusion you end up with 2 different datasets based on the same conceptual model. That is what has been developed in Canada as part of their National Road Network. I suggest the same thing will need to happen in Europe for Roads and other datasets.

www.geobase.ca

**ESRI Classes, Attributes, and Tagged Values**

You will also notice a few other differences in the logical model in the abstract classes. First, the top-level classes here are called Feature, a class from the ESRI Classes package. We use these classes to indicate classes that inherit from feature will end up as Feature Classes in the Geodatabase, and that classes that inherit from Object will end up as Object Classes. The attributes from Object and Feature mean that FerryLink will have an attribute for OBJECTID (table unique, system managed identity) and a second
attribute for Shape (the geometry of the features). Rather than have an associated GM_Curve class, each Feature will have a Shape column. In the logical model the type of geometry is entered as a tagged value. For FerryLink as an example, the tagged value looks like:

Classes vs. Attributes in the Logical Model

Next, what was a separate class for ER_Objectid needs some consideration. In the conceptual model ER_ObjectId is a 1:1 association to ER_Identifiable Object. In the logical model a decision has to be made whether this is a separate class or if those attributes are just properties of ER_RoadFeature or another class. In this case I chose to make them attributes.

Primary/Foreign Key Relationships

There is a second problem in that uml does not have a concept of primary and foreign keys. In the conceptual model this is great because we can represent many types of associations without regard for the implementation. For instance, a city can contain many neighborhoods, but we don’t need to worry about how to implement this in a conceptual model. For GIS this is really good because we can model many types of relationships and worry about how and when to enforce rules later in the design process.

For implementation we add tagged values onto associations that will be implemented using tabular relationships by indicating the appropriate pk/fk attributes. We also had to add some direction/origin tagged values but I think that was due to tool bugs rather than necessity.

Here is an example from one of the associations for FerryLink Attribute1:
OriginPrimaryKey and DestinationForeignKey are the pk/fk attributes for the Geodatabase. OriginClass indicates that FerryLink is the origin. Notification involves Geodatabase behavior that allows messages between classes to be pre-defined at a class level. In almost every case this is set to none.

**Geodatabase Model Packages**

A small note on the diagram for Geodatabase purposes. We stereotype packages for FeatureDatasets – containers for groups of feature classes with a common spatial reference. In uml logical models we need to organize the Objects into different packages than the features. It is a restriction of the Geodatabase that Objects cannot reside inside of feature datasets in the Geodatabase model. This is a practical issue for the logical model organization for packages. In the Road example I made the following packages:
**Geodatabase Data Types**

ESRI had to make a set of Classes and Data Types for implementation. For example, each of the database platforms handles text differently. We needed a datatype that supported our needs for implementation on multiple dbms platforms. Could we have used a different approach/standard? The risk is that by using another template we would have logical models that are difficult for our Technical Support group to support. In a logical model we also needed to specify field length and some other dbms properties for numeric values. An example from the EuroRoads model to set the length of a character string to 60 characters:
In this area I have no idea why ISO TC211 would see a need to standardize on data types for conceptual models, especially when “characterString” and other data types are not something we need to standardize in a Geo community. Do we even need data types for conceptual models? I try to avoid thinking about this level of detail in a conceptual model and just think about classes, names, and associations.

**Multiple Concrete Classes with Generalization Associations**

One of the presenters suggested that he could subclass RoadLink into NLRoadLink and have both as concrete classes. I think this may break some uml rules, but we certainly do not support multiple concrete classes in Geodatabase logical models connected with the generalization association. I think the resulting classes are ambiguous with respect to attribute content and relationships.

**Rules and Rule Evaluation**

Some of the rules expressed in the model are composition and multiplicity. In general, the Geodatabase approach is to allow data to be loaded into the database and have rules evaluated in the appropriate point in the users workflow. In that sense, I would not use the composition (black diamond) in an implementation model unless I understood the data management and performance characteristics. For instance, a RoadNetLink is composed of attributes, but the model also says the attributes can exist independent of the road link, and that the road link can exist independent of the attributes. Is that really composition? Anyway, for implementation I need to make choices about how and when to evaluate rules.

A specific example is the topology part of the model. It is ok to express some of these constraints for TP_Edge etc. in the conceptual model, but I would prefer to just describe the associations in conceptual terms without regard for the encoding of the data in a particular exchange format. In this case ESRI will not be using the ISO (GML) style approach in conceptual or logical models. I like the one diagram from the Visio model provided for the workshop:

![Diagram](image)

This type of approach makes sense for a conceptual model.

I am not in favor of the technical implementation required for:
ESRI can provide software interfaces that support these types of associations, but the data management and performance implications of data content for this approach will not be acceptable to our users.

Similarly, some of the following model is a conceptual data model, part of it will be implemented with software and tools in the ESRI environment rather than a complex data structure:

We suggest simple feature classes like Nodes and Links with software methods that can report the topological relationships through web services or other methods. We know this approach works, and we doubt some of the other approaches that have been implied/embedded in standards will perform well. The mixing of concepts and format (above) does not make much sense because the implementation models for different systems will encode topological rules in different ways. Also, one of the best things about
UML is the ability to represent conceptual associations without regard for implementation. I think the approach suggested in this diagram will actually hurt interoperability and the development of good conceptual schemas because it mandates a method for encoding data that will not perform or scale well. I should be free to implement the datasets however I want, and simply be able to support standard software interfaces.

**Domains**

In the Geodatabase we have Range Domains and Coded Value Domains. For implementing the EuroRoads model I had to choose a method for implementing the Codelists. I presumed that it was ok to store the strings in the rather than store integers. I am not sure what the implementation intent was from the conceptual model.

```
«CodeList»
ER_FormOfNode
+ER_Roundabout
+ER_EnclosedTrafficArea
+ER_PseudoNode
+ER_GradeSeparatedCrossing
```

What this means is that the user will see ER_Roundabout in ArcGIS applications, and that same value will be the string stored in the database. Based on other road network projects I doubt this is actually the intent.

**WFS, Transactions etc.**

While I certainly agree that transactional updates are important, I have no idea how this discussion crept into conceptual schema models. There seems to be an approach here of throw all ISO/OGC material at the problem rather than sticking to the problem at hand – a conceptual schema for roads.

I also question the significance of WFS and GML for this discussion. From the thousands of Internet mapping applications our users have developed, very few have implemented feature services because of the uncertainty of data volumes/network loading and impacts on scalability. I think this is an interesting intellectual discussion that has little to do with Conceptual Schemas.
Summary

In this document I tried to outline the key issues with moving from a conceptual to logical model in an automated way. I do think there are several discussion areas that could be further explored to understand where this discussion could lead. ESRI, as I stated in the workshop, is not in the CASE tool business but we need to understand how to support people that want to use CASE tools to build and share schemas. Below are the main discussion points described in some detail in this document.

- Documenting Spatial patterns
- Associations
  - Primary and foreign keys
  - Topological
  - Rules and constraints
- Database properties vs. object properties
  - Field lengths
  - Data types
  - Spatial patterns
  - Reserved words (tables, columns)
  - Spatial References and Grid sizes
  - Measures and Z values
- Codelists, enumerations, domains
- Behavior
- Accommodate multiple physical approaches
- Template UML versioning for tools
- Semantics checkers / validation

There would be some value in simplifying the transition from conceptual to logical schemas, but the result would impact only a few designers rather than the thousands of potential users for EuroSDI data and web services. The needs of designers should be prioritized against the return on other investments.