

Those Unfamiliar UML Concepts - An Introduction

Since UML (the Unified Modeling Language) is now the basis for further Modeling development of the SDSFIE, it might be useful to understand some of the special concepts that make it the desired modeling alternative for the standard. You may recall that IDEF1x was the modeling standard for earlier Releases of the TSSDS/SDSFIE, and that these models, when viewed, were a group of "boxes and lines". These were, in fact, a type of Entity Relationship Diagram, where the entities were the boxes and the relationships between entities were the lines. Implementing a model meant that each entity became a database table and the relationships indicated, but did not enforce, data integrity. In much the same way, UML Diagrams are a group of "boxes and lines". But "thus endeth" the similarity.

Boxes appear in UML Diagrams, but they are NOT (at least not normally) tables. And the lines, while they may represent relationships (what UML calls Associations), can show a lot more information. So it gets a little complicated. The "boxes" in a UML Diagram (at least for SDSFIE purposes) are what is called a class. Do not confuse this with Entity Class in the SDSFIE, or even Feature/Object Class in ESRI. Think of a class as nothing more than a collection of attributes (also called columns or data elements). Some folks refer to these classes as a "bag of attributes" (one or more attributes in a bag).

So why is this not a table? It has to do with the different kinds of lines, and the nature of these boxes/classes. This introduces the first significant difference between IDEF1x and UML. This is the concept of **inheritance**. The concept behind this is pretty much just what the term means. Let's say your parents had specific genes that you also have. You inherited these specific properties/characteristics from your parents. The same is true of classes in UML.

Let's say we have a class called "road_centerline". In IDEF1x, the box contained the attributes required of that table when constructed. Now let's say we take out all of the attributes that have to do with the characteristics of the surface of the road. They are removed from the "road_centerline" class, and placed in a class called "transportation_surface". This "transportation_surface" class is the parent and "road_centerline" is the child. When "road_centerline" is constructed, it will inherit these additional attributes from the parent. Will these attributes be in the "road_centerline" table when constructed? You bet.

This inheritance between the two classes is known as a **generalization**. The "road_centerline" class is known as a **leaf** class, and the "parent" is known as an **abstraction** class. And that class may have **abstraction** classes as parent of it (kind of grandparents), etc etc. So, when the final road_centerline table is constructed (sometime called instantiated), it contains all the attributes (specific genes) from parents, grandparents, etc. all that back to what is known as the **root class**. In the case of SDSFIE, the root class for all practical purposes is called the **SDSFIE_Feature** class. And the line associated with the generalization has an arrow at one end, pointing to the **abstraction** class.

In the model, this visually looks like Figure 1. SDSFIE_Feature contains feat_name and grid_value. When the future_projects_site gets constructed (instantiated), it will have these attributes in the table, even though they do not appear in the future_project_site class. In fact, if you look closely, only the top attributes in the future_project_site class (note that it is a **leaf**), are in that class. All of the others, are inherited from either parent (**abstraction** class SDSFIE_FutureProject) or grandparent (**abstraction** class SDSFIE_Feature). Now that you are confused, the next several papers will begin to actually look at how the Features in the SDSFIE will eventually be organized.

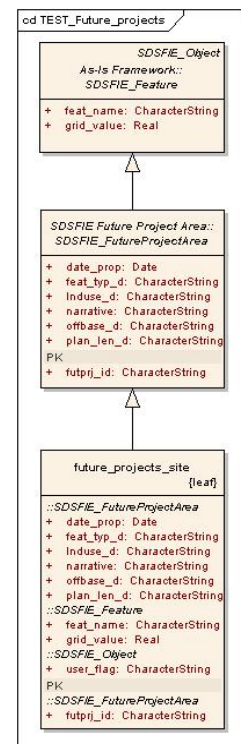


Figure 1