**OWL-RIM Improvement and Next Step**

Suggested improvements are described along classes and properties.

1. **OWL-RIM Class Restrictions**

The hierarchy (i.e., inheritance) of classes is well-defined. Classes defined using separate name spaces (e.g., dt, Hl7, rim) is a good design choice. However, the problematic part is a set of restrictions that makes subclass relationships. For example, "Class: Act SubClassOf: InfrastructureRoot " comply with the RIM semantics but restrictions like below makes the Class:Act SubclassOf *anonymous* sets of individuals that fulfil the property restrictions (e.g., Act.activityTime min 0 Thing)

|  |
| --- |
|  |
| Class: Act SubClassOf: (Act.activityTime min 0 Thing) and (Act.availabilityTime min 0 Thing) and (Act.classCode min 1 Thing) and (Act.code min 0 Thing) and ....() |
|  |

I can understand why Llyod used these restrictions for a class definition. In UML (OO), a class contains a set of attributes/properties, i.e., an attribute is always within the context of a single class. In UML or OO attributes cannot exist in isolation (without out a class definition). However, an ontology property is a first-class modelling element, means that, properties are stand-alone entities that can exist without specific classes. Therefore, their "connection" to classes are via "domain" and "range" values. To improve this, two steps are required:

* Remove all property restrictions from class definitions.
* Describe the restrictions within property definition, as below

|  |
| --- |
|  |
| ObjectProperty: Act.activityTime Domain: Act, Range: min 0 GTS |
|  |

Now the property "Act.activityTime" restriction is defined as "Range: min 0 GTS". Similarly, we have to update the other restrictions.

1. **OWL-RIM DataTypes**

In the current version, datatypes are simply expressed without any inheritance relation, i.e., all of them are subClassOf owl:thing. Attached is a sample datatype ontology showing inheritance among various types.

1. **OWL-RIM Property Reduction**

 No need to have separate properties with a same range value. For example below, "Act.id" and "Entity.id" range is SET\_II

|  |  |
| --- | --- |
| ObjectProperty: Act.id Domain: ActRange: SET\_II | ObjectProperty: Entity.id Domain: EntityRange: SET\_II |

We can combine this in one statement (as below) using multiple domain values (e.g., Domain: Act, Enity ) for the object property "id".

|  |
| --- |
|  |
| ObjectProperty: id Domain: Act, EnityRange: SET\_II |
|  |

This will reduce the number of object proprieties (a good practice) and still obeying the RIM semantics.

1. **OWL-RIM Vocabulary**

 For properties such as "Act.classCode" the range value is "CS". We need to describe the vocabulary domain for coded values (CS, CD, etc.) . Example vocabulary ontology for V3 is attached.

1. **OWL-RIM, DMIM, RMIM**

 Similar to OWL-RIM we need to define a methodology for building DMIM/RMIM equivalent ontologies. The scope of DMIM/RIMIM would be local to a domain/institution implementing them. The methodology should describe (1) how to reuse existing ones, may through some automated XSLTs rules; and (2) how to build DMIM/RMIM local ontologies from scratch.

**Next Step**

To implement all the five points discussed above, we need to build a set of lifting rules (MIF->OWL).