Annotation Properties and Individuals for driving the Application

A set of annotation properties and individuals created specifically to guide application functionality live in the application layer of the eagle-i ontologies (specifically in the eagle-i-app-def.owl file). These are used to create annotation axioms on classes and properties that also live in the eagle-i application layer. These annotation properties individuals and axioms function solely to guide application behavior.

Tables that define the eagle-i application annotation properties and individuals, respectively, and describe conventions for their use can be found here: **Table of Annotation Properties and Individuals**. The *InClassGroup* and *InPropertyGroup* annotation properties hold owl:individual IRIs as values, each of which guides a distinct application function as detailed in the table and below.

Conventions for Using Application Annotations

1) File location of axioms driving application functionality. As noted above, all axioms created using these properties and individuals should live in the appropriate application layer ontology file. Conventions for different properties are detailed below, but generally speaking, there is an application layer file paired with each domain file, and annotations on content in a given domain ontology file should live in the respective application file.

In addition to these annotation properties, logical domain and range restriction axioms are also consumed by the application logic. These axioms should live in the domain layer where the restriction is generally true and re-usable. But if the restriction is too narrowly defined for a given property domain or range in order to drive the desired application behavior, this axiom can live in the application layer. Alternatively, the *eagle-i domain constraint* or *eagle-i range constraint* annotation properties can be used to restrict domain and ranges using annotation axioms for such cases

Note: If contradicting restrictions exist in both the domain and application layers, the app layer annotations are dominant.

2) Domain and Range constraints as logical vs annotation axioms. Related to the issue above is when to define a domain or range for a property using a logical axiom or an annotation axiom. As noted above, if the property or the constraint is meant to drive application functionality, and not something deemed relevant for the domain model released to the community, it should live in the application layer. That said, there are two approaches for asserting domain and range constraints that the application will understand: (1) as a logical axiom using rdfs:domain and rdfs:range constructs, or (2) as an annotation axiom using the application-specific annotation properties *eagle-i-domain constraint* or *eagle-i range constraint*. Note here that in cases where both types of assertions are present, the annotation axiom will take precedence.

Going forward, a general rule would be that in cases where the domain and range needed to drive the desired application functionality are more narrowly scoped than the general intent of the property, use the annotation axioms to implement these constraints. This will be needed only in cases where the property is a re-used community property imported from another ontology (e.g. develops_into RO_0002202, or located_in RO_0001025), or an ERO property that is meant for community re-use as part of the released domain model -- where we want the application to read a more constrained domain or range than it would be logically correct to assert in the Domain model). But these are rare cases. In practice, most properties implemented in eagle-i are application-specific, and very tightly constrained in their domain and range to support application needs. They are too restrictive to be re-used in broader contexts by the community. Examples here include derives_from_anatomical_entity and has_part_construct_insert. For these, it is OK to define domain or range constraints as logical or annotation axioms. But it would be good to define a consistent best practice here and adhere to it going forward (and ideally apply it retrospectively in a refactoring effort). In practice, you will note that logical axioms are used in most cases -- but this is primarily because most of the ontology was developed before the annotation axiom functionality was implemented.

Finally, with the new approach of generating the eagle-i ontology directly from the VIVO-ISF, another place the annotation axiom approach has been useful is in defining domain and range constraints for eagle-i where our needs are not consistent with what we find in the VIVO-ISF. This use case may come up more often as eagle-i 'specializes' more general VIVO-ISF content for use in its application ontology.

Distinguishing Resources from Referenced Taxonomies: Any class referenced in the range of a property needs to have an *InClassGroup* annotation that defines it as a resource or referenced taxonomy root. To define the class as a resource, the value Class Instance Create is used (along with a second Class Primary Resource Type or Class Embedded Resource Type annotation as applicable). To define a class as a reference taxonomy root, the *InClassGroup* annotation on the class should have the value Class Referenced Taxonomy. Here, a second annotation using the eagle-i Reference Taxonomy IRI annotation property is also required, with the IRI of the owl file hosting the root class as the value. More detail on usage conventions here are given in the Task Workflow and Scenario section below.

3) Inheritance of Application Annotation Axioms. Generally speaking, *InPropertyGroup* annotation axioms are not inherited by descendent properties. By contrast, *InClassGroup* annotations are inherited by descendent classes, and therefore do not need to be repeated. So if a descendent of a class previously defined as a referenced taxonomy root with a *InClassGroup* annotation is set as the range of another property (and thus made the root of a sub-referenced taxonomy), this root class does not need an additional *InClassGroup* Class Referenced Taxonomy axiom. But eagle-i referenced taxonomy IRI annotations are not inherited, so this descendent taxonomy root would still need an eagle-i Reference Taxonomy IRI annotation axiom pointing to the owl file where the root class lives.

Next: Core Task Workflows and Scenarios