ERO and VIVO migration/refactoring

As described in the introduction, the main content of ISF is based on merging the ERO and VIVO ontologies. The following describes how this merge is implemented in the SVN repository.

A copy of the original ontology files of VIVO and ERO was obtained and placed in the “vivo” and “ero” modules folders respectively. ERO also had a separate set of “application” OWL files that contained additional content relevant for the use of the ERO ontology in the eagle-i application. These files were placed in the “eagle-i” directory under the “app-views” directory.

A top level OWL file was created for each of ERO and VIVO (named ero.owl and vivo.owl) in the corresponding modules. These files import the parts of ERO and VIVO ontologies that are being reused by ISF. Application specific files named “ero-app.owl” and “vivo-app.owl” were also created in the corresponding “app-views” directory and they import the corresponding ontology files “ero.owl” and “vivo.owl”. These application specific files contain mostly sets of annotations that allow correct displaying of ISF content under eagle-i and VIVO applications.

A top importer file named arg.owl (to be renamed isf.owl) in the “ontology” directory imports the above two files (not the application specific files) and the top OWL for each module. This file gives a full view over the ISF ontology content. A corresponding “arg-app.owl” is created and placed in the “eagle-i” application directory for the purpose of being able to browse the ISF ontology in the eagle-i ontology browser.

Entity and axiom migration

After the initial merge described above, the ISF content is being reviewed and refactored as needed. The result of this ongoing process is that a set of OWL entities (classes, properties, etc.) and OWL axioms (logical definitions or annotations) are being removed from ISF. This occurs when there is an overlap between ERO and VIVO, or when one or the other, or both, are migrating their logical representation to a new representation developed during the refactoring process. The new representations are meant to be more detailed and generalized so that they can be applied to a wider range of use-cases.

When an entity or axiom is removed, it might still be needed by an ERO or VIVO based application until the application is also migrated. To support this possibility, the entities and axioms are pushed to the corresponding ero-app.owl and vivo-app.owl files so that an application (currently eagle-i and VIVO applications) can still have an application view that does not appear to be changing. The process of pushing an axiom from ISF to an application specific file is called “ISF deprecation”. An “ISF deprecated” entity or axiom could still be an extension of an ISF entity, but the extension is considered too specific and, for the time being, is being pushed to an application view until wider need for the entity or axiom becomes apparent. At that point, the entity or axiom can be moved back to ISF. Other entities and axioms are truly “ISF deprecated” and will not be incorporated in ISF until the specific deprecation reason is resolved. For example, an OWL class could have been vague and used with multiple distinct meanings. Such a class needs to be re-represented with multiple classes (one of which could be the original class) with clear and distinct definitions. At that point, one or more of the re-represented classes could be incorporated into ISF.

A subset of the “ISF deprecated” entities and axioms will be fully deprecated and not used by any applications or data. At that point, they will be moved to the ero-deprecated.owl and vivo-deprecated.owl files in the same directories where the application files exist.

Ontology vs. application separation

Any application that uses the ISF ontology will likely need additional OWL content to support the specific application. This content can be extensions of ISF classes, preferred annotations, equivalence axioms, etc. The current SVN repository includes this content for the eagle-i and VIVO applications under the “app-views” directory. Other applications such as Plumage [8] would be welcome to include such application configurations in this directory.

Next: Ontology browser