ProMo Foundation Ontology Editor

Purpose:

Ontologies capture domain-specific knowledge. Thus being adaptive to specific application domains was a primary design specification. Thus, the user defines the disciplines or involved scientific domains involved in the user's application domain. The same applies to the terminology being defined.

Behaviour:

The software copies a required directories tree that holds the user data and resources when defining a **new** ontology. Alternatively, if an **existing** ontology is to be edited, the software will ask if the defined variable/expression file is to be kept or deleted. If the variable file is deleted, all elements are subject to being edited. Otherwise, the ontology can be extended, but the other elements are frozen.

Note: The generated ontology is used in all remaining ProMo programs.

Note: As part of the development of the remaining ProMo programs, some terms are in the process of being fixed.

Description:

ProMo models processes as coupled directed graphs in which abstract **tokens** live. The token concept allows ProMo to model a wide range of processes broader than physical processes but may include abstract domains like control, optimisation, life-cycle analysis, and the like. The nodes represent capacities for the tokens, while the directed arcs represent the transfer of tokens between nodes.

For **physical** systems, the tokens are the conserved quantities and the species mass.

Tree structure:

The first operation is to define a tree of the domains involved.

The application lives in a set of disciplines, scientific domains.

The tree's branches may be related in one of two ways: either **inter** or **intra** relation. Inter defines the two branches connected over **inter-**faces not transferring tokens but information, while **intra** imposes **intra-**faces transferring tokens.

For example the root has two inter-children physical and control. The physical node has three inter-children: material, reactions, macroscopic. Macroscopic then may have two intra-children. Namely fluid and solid. Fluid again has two intra-children: liquid and gas.

```
Root - || - control
. . . . . .|| - physical - || - materials
. . . . . . . . . . . . . . . . . || - reactions –
. . . . . . . . . . . . . . . . . || - macroscopic - | - solid
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | - fluid - |
- liquid
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | - gas
```

The tree also defines what kind of interactions are defined. A || indicates an inter-relation, realised in the form of interfaces, whilst a single bar | indicates a intra-relation, realised as intrafaces.
The intra-realation implies that tokens are transferred across the intraface.
The inter-relation implies that information is exchanged between the nodes connected by via an interface. A typical example is getting physical properties from a database where a node in a phase like liquid is providing the information of pressure, temperature and composition asking for getting back a partial molar enthalpy.

Each node in the domain tree has two sets of information: One associated with building the structures and one associated with defining the variable/expression multi-way bipartite graph.

Section structure defines the basic entities being later modelled by sets of equations. The sample ontology uses engineering terminology. The entities are placed in a time scale: constant, dynamic or event-dynamic. They also capture the information of being distributed: lumped, where the relevant intensive properties are not a function of the location, while for distributed systems, they are a function of the coordinates.

Typical physical entities are

- reservoirs being infinitely large and thus constant
- dynamic or event-dynamic lumped systems
- dynamic or event-dynamic 1D | 2D | 3D distributed
- lumped arc transferring energy or mass via different mechanisms

Section behaviour defines the variable classes.

– end –