Towards an ontology-based software standard for models

ICME 2016

Heinz A Preisig*, Arne Tobias Elve

* Department of Chemical Engineering
Norwegian University of Science and Technology
Trondheim, Norway
e-mail: Heinz.Preisig@chemeng.ntnu.no *https://www.ntnu.no/ansatte/heinz.a.preisig*

ABSTRACT

Motivation

We want to put models into the centre of computational engineering. As it is now, the software topology is mostly vertical with the model on the top end, whilst we want to have a more circular structure, more like a spiral with the models being in the centre.

Today, we mostly use models cast into a form that reflects the problem we want to solve, for example a simulation problem: given initial conditions, boundary conditions a fully covering set of parameterised equations and the parameters to complete the instantiation. A parameterised model provides some flexibility to utilise the model for a range of problems though for the same application. A strong limitation of this approach is that the model cannot be modified in terms of structure except that we can eliminate some terms be setting a factor in the term to zero, thus "zero" it out. Neural nets may be a illustrative example of such an approach. These model are most commonly also available in coded form and wrapped into a software structure that matches the used solution tool. For dynamic models, for example, this will be the right-hand-side of the differential equations representing the dynamic behaviour of system, is to be provided as a software module, with the time and state as inputs, whilst the derivative are the outputs. The specifics of the interface structure is dictated by the application.

Thinking pattern

Our approach is to provide a model in a generic form, as a **coded algebraic structure** that does not reflect any particular problem that one wants to solve. The model should be purely algebraic and not instantiated with any numerical information. The formal representation must be simple using a minimum number of objects. Thus a very simple language serves the purpose of representing the algebraic structure. This requires an appropriate parser, which can be combined with a template machine to generate target code and one set of templates per language.

This infrastructure is to be augmented with information that provide the additional pieces of code required to make the resulting computational modules to fit the solver structure. Again, the template machine can take care of this splicing operation.

Infrastructure must also enable mathematical manipulations to modify the form of the equations to match the numerical problem to be defined and it must also be possible to apply assumptions such as time-scale assumptions in a systematic way.

Having these construction elements in place, one has a factory for constructing generic as well as dedicated software tools using the same library of models. This not only enhances the reliability of the code, but also makes it easy to construct applications quickly and adjust their behaviour if so required.

What do we gain ?

The main advantage is that the same models are made available for different applications. Manipulations and simplifications become programmatically executed operations and the same

applies for the coding. In contrast to doing these operations manually and then code it, these procedures will be error free, thus no manipulation errors, coding and transcript errors -- and **very fas**t. Changing the model and make it available for a particular problem, as well as instantiating and solving the problem is much more efficient and systematic.

Consequences are that many more computational experiments can be performed in the same time and whilst this has the obvious advantages of being able to explore a much larger problem space, it also is more time efficient. Thus more is achieved in a shorter time with higher level of correctness.

This contribution

The presentation will update on our effort to capture multi-disciplinary models in a common framework, which we term *ontology* as it captures the behaviour of the included classes of models. The resulting ontology forms a tree with the branches inheriting the properties from below adding increasingly more details and refinement as one moves outwards to the leaves. The leave nodes represent the ontology of a specialised field, which in sequel are linked together through inheritance from both branches to realise the integration of the different disciplines.

Whilst so far we have tested the representation on controlled physical systems, we are confident that the approach also seamlessly applies to multi-scale system, as we already use the method for the representation of multi-phase systems.

References:

Preisig, Heinz A.. (2015) Constructing an ontology for physical-chemical processes. Computer-aided chemical engineering. vol. 37.

Preisig, Heinz A.. (2015) Constructing process models systematically. Chemical Engineering Transactions. vol. 43.