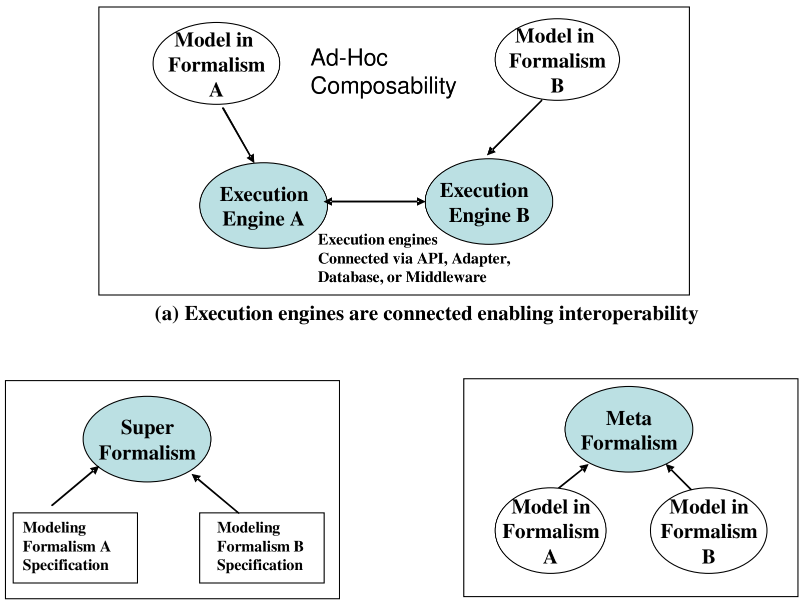
Position Statement on CAMPAM

In our work, we are primarily interested in multi-formalism models where each of the formalisms requires different types of execution algorithms. This is different than other research areas currently in found in the existing literature. The motivation of this research is to enable efficient ways to model planning and manufacturing systems seen in semiconductor supply chain networks. Both planning and manufacturing models are very complex. Optimization techniques have been shown well to work for many types of planning problems while simulation is very good for modeling manufacturing processes. Trying to combine the development and execution of these models into the same environment does not work well with today's tools. Very good tools can be found to support modeling optimizations or simulations, but we have not been able to find one that enables the composition of both types of these models in well integrated manner.

We have been working on a multi-formalism modeling approach that enable the integrated development of these two types of models and also allows the use of most efficient algorithms that supports the models.

Background

The different types of multi-formalism modeling research we encountered in current research are illustrated in Figure 1.



(a) Execution engines are connected enabling interoperability

(b) Formalism specifications mapped into new super formalism

(c) Models transformed into different formalism

Figure 1: Different approaches for multi-formalism modeling

The first approach addresses composability in an ad-hoc way (see 1a). The different execution algorithms are connected in an informal way to enable the systems to interoperate. Many commercial packages offer this type of connection via middleware, or data connection through database. These implementations work well when the relations and interface between the two models can be easily formulated in terms of inputs and outputs and have simple interoperability protocols. How to maintain proper composability of the models is not addressed and left up to the modeler to work out in whatever way is possible.

A step forward from ad-hoc toward multi-formalism is to characterize ways in which two models can exchange their data under some control regime[1]. The general characteristics of inputs and outputs supported by two modeling formalism can be used to map data from one to another. For example, outputs generated from a discrete event simulation (data delivered as messages on ports) can be mapped onto inputs (data received as messages) processed by an agent model. This type of input to output (and output to input) mappings can be formalized in a module that sits between models described in discrete event system specification and reactive action planning modeling approaches. This type of composability, however, cannot account for internal structures of discrete event simulation and agent model specifications and furthermore does not account for consistency of data mappings.

The next multi-formalism approach is to map the formalisms into a new super formalism (see Figure 1b). If a supporting algorithm is created, then only a single execution engine is needed. For example, a formalism and supporting algorithm exists for representing combined discrete event and continuous model [2]. This approach would not be practical for formalisms that have fundamentally different structure and objectives. For example, linear programming and discrete event simulation have very different purposes and their underlying algorithms are fundamentally different and non-compatible.

The last approach shown in Figure 1c transforms the formalisms into a meta-formalism that can subsume the capabilities of the sub formalisms. If the structure and behavior of the sub-formalisms can be mapped into an equivalent representation of the meta-formalism, the aggregate model can be validated against the properties of the meta-formalism. This formalism only requires the execution engine of the meta-formalism . Model transformation also enables the validation of a model against properties of the other formalisms . The meta-modeling approach also would not work well with fundamentally different formalisms. Using the example of LP and DES again, these models do not have a good correlation. DESs are hierarchical system models which use ports for coupling and have a concept of time. LP models consist of an objective function and constraints. Transforming these two very different types of models into a single representation would be very difficult, and probably not practical.

Different Approach to Multi-Formalism Modeling

In this approach, we propose that a model and a transformation algorithm be introduced between the two different formalism models (See Figure 2). We refer to this layer as a knowledge interchange broker (KIB). It must address composability mismatches between the different models. The figure identifies 4 different items.

First there is formalism composability. This must address the mismatches between the modeling structures between the two formalisms. Using the example of linear programming and discrete event simulation, the data and timing of discrete events must be mapped to the linear programming multi-dimensional vector structures used to define objective functions, data variables, data constraints and decision variables. Things such as when an event occurs, where the event came from, and what the data contents of the event from the DES must be transformed into data elements in the different LP arrays. The first two items, when an event occurs and where the event came from can be straightforward to map. However, mapping the data contents of an event can be very difficult if the allowable data structures are unbounded. For a feasible implementation, the contents must be constrained, or a general purpose programming language will be required to do the transformation modeling. We have found that constraining based on the domain you are modeling has become a very useful technique.

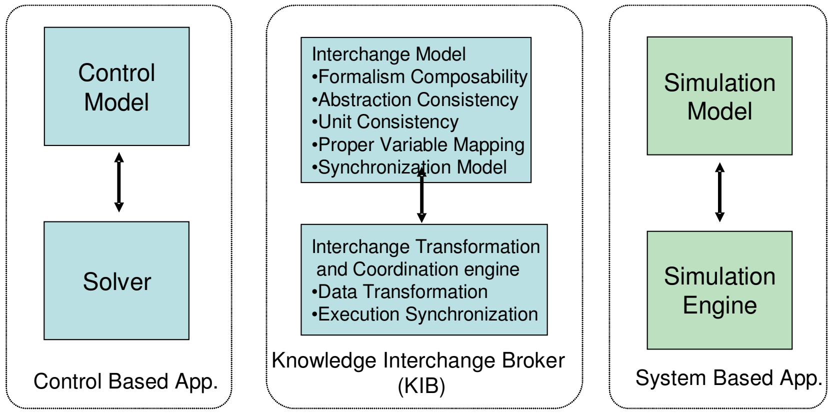


Figure 2. Functions of KIB in multi-formalism modeling

The next four items listed: *abstraction consistency, unit consistency, variable mapping, and synchronization model* must be mapped between the models when considering the domain semantics. Abstraction consistency must be considered if the same type of data is represented at different levels of abstraction in each of the models. Aggregation / disaggregation algorithms can be used to match abstractions of data. Unit consistency mapping is required if models have been written using different units of measurement. Variable mapping is specifying which output variables from one model map to the inputs of the other. The synchronization model specifies how and when the two models should pass control and data at execution time.

The KIB engine can execute the transformation models and send the data through whatever middleware has been chosen at implementation time.

References

[1] H. S. Sarjoughian and J. Plummer, "Design and Implementation of a Bridge between RAP and DEVS," Computer Science and Engineering, Arizona State University, Tempe, AZ, Internal Report, August, 2002.

[2] H. Prähofer, "Systems Theoretic Foundations for Combined Discrete Continuous System Simulation," Johannes Kepler University, Linz, Austria, Ph.D. Dissertation, 1991.

[3] J. de Lara and H. Vangheluwe, "ATOM3: A tool for multi-formalism and meta-modelling," In *Proc.* European Joint Conference on Theory and Practice of Software (ETAPS), Fundamental Approaches to Software Engineering (FASE), Grenoble, France, pp.174-188, 2002.

[4] H. Vangheluwe and J. d. Lara, "Computer Automated Multi-Paradigm Modelling for Analysis and Design of Traffic Networks," In *Proc.* Proceedings of 2004 Winter Simulation Conference, Washington, D.C., pp.249-258, 2004.