Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

MODEL COMPOSABILITY

Hessam S. Sarjoughian

Arizona Center for Integrative Modeling & Simulation
Computer Science & Engineering Department
Arizona State University
Tempe, Arizona 85281-8809, U.S.A.

ABSTRACT

Composition of models is considered essential in developing heterogeneous complex systems and in particular simulation models capable of expressing a system's structure and behavior. This paper describes model composability concepts and approaches in terms of modeling formalisms. These composability approaches along with some of the key capabilities and challenges they pose are presented in the context of semiconductor supply chain manufacturing systems.

1 INTRODUCTION

Many contemporary and future systems are integrated from a range of simple to complex sub-systems. Examples are information, manufacturing, and transportation enterprises. The purposes for these systems vary significantly as each system is to satisfy a set of users and system requirements. To understand a system's capabilities and limitations, dynamical models are developed. They help to specify structural and behavioral specifications that can be simulated and, thus, can be used to evaluate analysis, design, development, and testing decisions.

Enterprise (or distributed) systems are complex and to understand their intricate dynamics it is often beneficial or necessary to use different kinds of models to represent each sub-system. Heterogeneous model types can offer greater flexibility as opposed to homogenous model types. However, combining different model types poses a variety of challenges depending on the system being modeled (Page and Opper 1999, Davis et al. 2000, Kasputis and Ng 2000, Sarjoughian and Cellier 2001, Davis and Anderson 2004, Fujimoto et al. 2002). Views regarding the kinds of problems encountered in simulation composability and future advances and investments in the context of the Department of Defense needs are described in (Davis and Anderson 2004).

Partitioning a system into layers, each of which consists of a set of components, is a crucial step in high-level model specification. Decomposition and composition of models are challenging when models are heterogeneous in terms of their formal specifications – e.g., discrete-event and optimization models have different structural and behavioral specifications.

Given the diversity of parts and resulting complexity of the systems, criticality of operation, and enormous financial consequences, simulation is being used more and more as the primary science and technology to aid analysis, design, implementation, and testing. Indeed, simulation can be used across each phase of system development as proposed in Simulation-Based Acquisition (SBA 1998). For example, a suite of system-level simulation models may be developed to help analyze requirements and evaluate potential architectural solutions far in advance of defining detailed design specifications. Yet, another suite of models may be used to help develop detailed design specifications which can be implemented.

Systems theory, object-orientation, and logical processes worldviews are well known approaches for describing dynamic systems (Cellier 1991, Banks et al. 2004, Fishwick 1995, Fujimoto 2000, Zeigler et al. 2000). Different modeling paradigms are suitable for different kinds of needs – for some examples of modeling approaches see (Sarjoughian and Cellier 2001, Mosterman and Vangheluwe 2002, Barros and Sarjoughian 2004). A discrete event modeling approach may be used to describe manufacturing processes and their interactions as a collection of model components. Alternatively, given historical data, a continuous model can be developed to show how inventory holding varies in relation to factory throughput. Optimization models may be developed to understand logistics and financial impact of satisfying customer demand.

Manufacturing supply chains and military systems of systems are two well known network systems that are often modeled at varying levels of details and from different aspects. Depending on the system's different types of behaviors, system simulation models may be developed using a monolithic modeling paradigm in one extreme and a combination of modeling paradigms at the other extreme.

1-4244-0501-7/06/$20.00 ©2006 IEEE

For example, a manufacturing process may be modeled using a single modeling approach (e.g., agents or Petri-net). However, to effectively design, assess, and operate a manufacturing supply chain in an optimal fashion, it is useful to employ discrete-event and optimization models (Kempf 2004). Similarly, to evaluate possible system design alternatives given a variety of operational and engagement patterns among a sensor, weapons, and command/control, and communication models, discrete-event, agent, and GIS models are important to be used (Hall 2005).

2 MULTI-LAYER MODELING

Many real world network systems can be abstracted to consist of two parts – one is a *plant* and another is a *controller* (see Figure 1). At a high level of abstractions, the plant and controller models of a network system can be considered as computations which exchange data (i.e., states) and control (i.e., commands) under some well-defined constraints. The plant/controller pair can form a symbiotic relationship, where one is concerned with *operations* of a network process and the other is concerned with the *control* of the network management. The plant dynamics can be described in a variety of forms – e.g., discrete, continuous, or some combination thereof. The controller can be based on feedback control, event-based, and fuzzy control. For example, a plant can have stochastic discrete-event operations taking place at the present time and the controller can be an optimization-based feedback control. The plant's operations occur from one time instant to the next – i.e., the dynamics of the plant can be modeled as deterministic (or stochastic) continuous and discrete equations. The controller operations can be based on the current or future state of the plant as well as present and expected inputs that can affect controller's decision making.

The plant and controller is considered a two-layer system – plant and controller are at lower and higher levels of abstractions, respectively. The plant is responsible for carrying out low-level operations, some of which are requested from the control. It carries out some operations given some inputs and generates some outputs. The controller is responsible to carry out higher level operations given details provided from the plant. Its responsibility is to constrain the operations of the plant to those that are deemed desirable or acceptable for some finite time period in the future.

The controller itself can be viewed as plant and its operations sanctioned by another controller. This leads to multi-layer plant/controller specifications. This separation offers fundamental benefits in terms of developing separate and hybrid models and simulations (for example see Praehofer 1991, van Beek et al. 1997, Barros 2002). It provides conceptual separation of knowledge. Plant and controller models play distinct roles where one supplies data and the other provides control. It also enables supporting multiple levels of control for a plant. One (tactical) controller provides tactical commands to the plant and another controller provides strategic plans to the tactical controller. Another advantage is that alternative modeling choices may be used for the plant or controller. For example, the plant may be described in terms of discrete and continuous models and the controller may be described in terms of linear optimization or event-based control models. In the case of semiconductor manufacturing supply chain, the plant is the *manufacturing process* and the control is *operational/logistics* controller (see Figure 1).

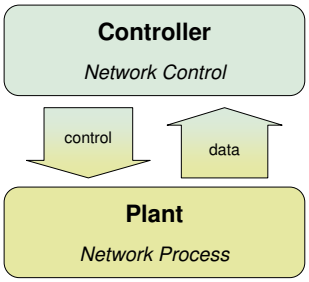

Figure 1: A Two-Level Plant and Control Model

3 PLANT AND CONTROL MODELS

In the domain of semiconductor manufacturing supply chain systems, models of discrete processes, control policies, and decision plans are important in handling stochastic dynamics of individual parts of manufacturing processes under tactical control and strategic planning, given short- and long-term supply and customer demand (Kempf 2004). These kinds of aggregated models play a central role in the understanding of not only physical operations of processes, but also how they are managed using tactical control and strategic planning. Such complex industries can gain both short-term and long-term technical competitiveness as well as financial advantages. Another crucial benefit of synthesizing different kinds of models is the ability to better handle mismatches (between competing objectives arising from operational vs. decision aspects, for example).

A canonical manufacturing process can be modeled as a collection of inventory, factory, and transportation nodes. The inventory (and transportation) nodes store (transport) products such as raw material, semi-finished goods, and finished goods. The factory node processes the products it receives from inventories and sends the processed products to inventory or transportation nodes after some time period. These nodes together characterize the state of the supply chain process at any one time instant. Each factory, inventory, and transportation node can have stochastic yield and

duration which in turn results in a chain of nodes that can exhibit complex dynamics.

A well known approach for controlling a supply chain is to use linear optimization (LP) (Rardin 1998). An optimization model describes a set of formulas and relationships (constraints c_1, c_2, etc.). A solver can be used to optimize the optimization model's variables of interests (e.g., inventory holdings) given some objective functions. Some examples of decisions can be how much should be held in the Finished Goods inventory, how many of which products should a factory produce, and what transportation mode to use for shipping products from an inventory to customer (see Figure 2). The responsibility of the controller is to optimize expected vs. actual demand and supply over some future time horizon given key data about manufacturing nodes (e.g., work-in-progress in a Finish node) (Godding et al. 2004). Alternatively, it may be useful to use a combination of optimization constraints and control-theoretic feedback/feed-forward filters which is known as Model Predictive Control (MPC) (Qin and Badgwell 2003, Wang et al. 2004, Sarjoughian et al. 2005, Huang et al. 2006).

To integrate these models requires accounting for differences between data types that are used in, for example, DEVS and LP models. The structure of input and output data for DEVS models is complex as compared with the data for LP models. Data for DEVS models are specified in terms of messages each defined in terms of *port* and *value* pairs – the port is a string and the data can be an arbitrary expression (e.g., a queue of key and value pairs). The structure of input and output data for LP models are simple data types which may be formed into vectors (e.g., array of reals). Therefore, for manufacturing process and optimization models to interact, their data types needs to be syntactically mapped from one to the other (Godding et al. 2004, Huang et al. 2006). For example, the Finish process produces a set of finished products Lots at output port *Data-Out*. The number of tested products is provided asinput to the LP model.

Aside from input and output exchanges and mappings, it is also necessary for the behaviors of the manufacturing process and optimization models to be well-defined – data exchanges need to occur at appropriate times and frequency. For example, the manufacturing process model advances from day to day whereas the optimization model is solved at the start of each day. In this scenario, a command determined by the optimization model requests n products from the Semi-Finished Goods to be released into Finish - i.e., the Finish receives a release command on port *Control-In* with value n. Details of the DEVS, LP, and MPC models partially shown in Figure 2 can be found in Singh et al. (2004), Godding et al. (2004), Sarjoughian et al. (2005), Wang et al. (2005), Huang et al. (2006).

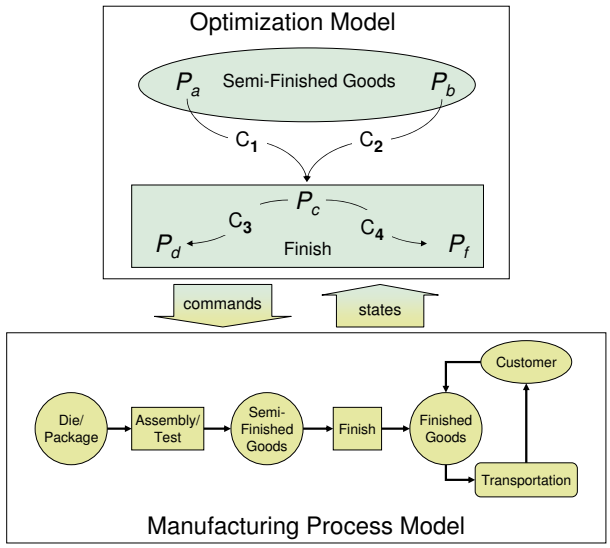

Figure 2: A Snippet of a Semiconductor Manufacturing Process and Optimization Model

4 MODEL COMPOSABILITY CONCEPTS AND APPROACHES

Considering the multi-layer modeling concept, it is possible to use a single modeling formalism to describe a plant and its controller. However, if a system under consideration has parts with dynamics that are intrinsically different, it is crucial to use multiple modeling formalisms. The above semiconductor manufacturing supply chain system is an example where the plant (manufacturing process) is suitable to be modeled as discrete-event model and the controller (logistic control) as an optimization model. A key difference between these models is that the discrete-event model describes how nodes of a manufacturing process network affect one another and the optimization model searches for optimal operation of the overall manufacturing process.

4.1 Modeling Formalisms

A modeling formalism can be defined to consist of two parts: *model specification* and *execution algorithm* (Sarjoughian and Zeigler 2000). The former is a mathematical theory describing the kinds of structure and behavior that can be described with it. The latter specifies an algorithm that can correctly execute any model that is described in accordance with the model specification.

A model A that can be specified in a modeling formalism Ψ is denoted as $M_{A,\{\Psi\}}$ – i.e., model A adheres to the model specification Ψ and can be executed using its execution algorithm. A model composed of a finite number of disparate models (e.g., $A, B, \ldots, K$) specified in a finite number of

distinct modeling formalisms (e.g., $\Psi, \Phi, \ldots, \Omega$) is denoted as $M_{\cup A,B,\ldots,K,\{\Psi,\Phi,\ldots,\Omega\}}$.

Approaches to model composability are classified into *mono*, *super*, *meta*, and *poly* modeling formalisms. These approaches provide different kinds of capabilities toward model composition. The first two are grounded in the concept that in some cases principally a single formalism is well suited for modeling different parts of a system. In contrast, the latter two are aimed at some other cases where disparate modeling formalisms are crucial to describe the parts of a complex system. Despite the differences among model composability approaches, every approach must ensure interactions among composed model components are structurally and behaviorally well-defined.

Exchange of data and control, causality of input and output interactions, and synchronization of models' executions with respect to time must be well-defined. This requires that not only individual model specifications are executed correctly, but also their compositions (i.e., the execution of multiple execution algorithms are well-defined with respect to the composition of their model specifications).

To help describe the above model composability approaches, model specifications are shown as rectangle, rounded rectangle, oval, and hexagon icons (see Figures 3 and 4). Process and control models are shown as rounded rectangles and ovals, respectively. A composite model that consists of two or more models (whether specified in one or multiple modeling formalisms) is shown as a rectangle. For example, Figure 3 shows that the manufacturing and control models as well as their composition are described in Ψ. A Knowledge Interchange Broker model (Sarjoughian and Plummer 2002) that composes multiple models specified in multiple modeling formalisms is shown as a hexagon (see Figure 4).

A modeling formalism can have several implementations (i.e., software realizations) based on the choice of programming languages – e.g., a mathematical model can be designed and implemented using object-oriented modeling concepts and a programming language. Software realizations of a modeling formalism also can be affected given the intended or expected modeling and simulation applications (e.g., parallel/distributed simulation for large-scale application domains). In addition, entity relations (ER models) and the XML family of models are also referred to as models. Here these are not considered since (i) such models are primarily aimed at describing structure of data instead of a system's structure and the behavior and (ii) they can be subsumed in component-based and object-oriented modeling approaches.

Before proceeding further, it is useful to note that model specifications are also described and implemented using (high-level) programming languages. Therefore, a model specification can be referred to as a mathematical model specification or a software model specification. Here mathematical specifications are considered (e.g., $\langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{conf}, \lambda, ta \rangle$ (Chow 1996) is the mathematical specification for Parallel DEVS atomic model). It is also helpful to note that different terms are used for execution algorithms. An algorithm which is void of software design choices and implementations is sometime referred to as an abstract simulator or solver.

4.2 Mono and Super Model Composability

It is common to use a single modeling formalism to specify one aspect of a system. Using a mono modeling approach provides key advantages - decomposition (or hierarchical composition) of a model into (from) parts can be carried out systematically (e.g., Wymore 1993). Modeling formalisms help modelers specify dynamical systems given well-defined syntax and semantics. The syntax specifies the allowed structures for inputs, outputs, states, and functions. This is the model specification. The semantics specify the behavior of the structural elements. This is the execution algorithm.

For example, discrete-event modeling is often used to specify discrete processes. A mono modeling formalism can also be used to specify different aspects of a system - detailed process flow dynamics with simplified event-based control. Using this modeling approach, models of different parts of a system adhere to a single structure and behavior specification. That is, $M_{A\cup B,\{\Psi\}}$ may be used to model discrete-event process model A, event-based control model B, and their interactions using modeling formalism Ψ. For the semiconductor manufacturing supply chain systems, the manufacturing process and control models are identified as A and B (see Figure 3). For example, these models can be specified in the DEVS formalism (Zeigler, Praehofer, and Kim 2000) (shown as in Figure 3(a)). A fundamental benefit of using a mono modeling formalism is that manufacturing and control models as well as their interactions are described in Ψ. This approach significantly simplifies integration of models and their executions.

Use of a mono modeling formalism, however, may not be suitable if the models that are to be composed are intrinsically different — the models are best described in different modeling paradigms — and are not fully supported. This is because a formalism is suitable to model only some parts of a complex, heterogeneous system while all the remaining parts must either be abstracted away or formulated within other modeling paradigms. To overcome this difficulty, a super modeling formalism may be used as shown in Figure 3(b).

In the super modeling formalism approach, a modeling formalism ($M_{\tilde{B},\{\Phi\}}$) is encapsulated within the super-formalism $M_{A\cup B,\{\Psi\}}$. Models $\tilde{B}$ and B are different and the former must be encapsulated inside the latter. A general approach is to use multiple model specification abstractions and hide the details of an encapsulated lower-level model

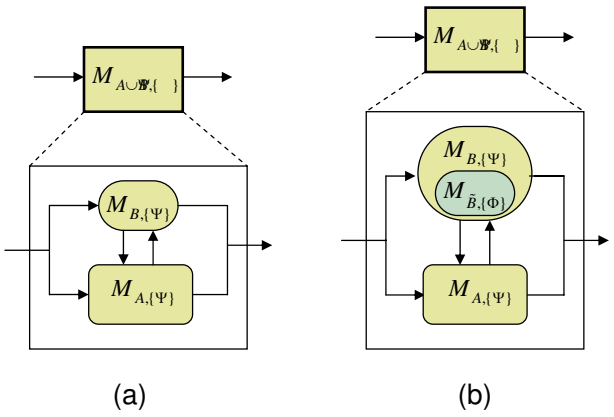

Figure 3: (a) Mono and (b) Super Modeling Formalisms

specification inside an enclosing higher-level model specification - e.g., a simple optimization model described in LP can be encapsulated as an I/O System (i.e., atomic) DEVS component. This allows the specification of the interactions between models A and B to be described within Ψ. This approach, however, has to ensure the modeling formalisms that are encapsulated within it have a well-defined relationship with respect to one another. That is, the super modeling formalism must assert the kinds of disparate dynamics that can be specified within it. Model encapsulation (e.g., enclosing $\tilde{B}$ inside B) requires the encapsulated model to be well-defined - the input/output structure and behavior of the encapsulation of the optimization model is guaranteed by the enclosing model specification. In the semiconductor manufacturing supply chain example, a logistic controller can be a linear optimization model $\tilde{B}$ in formalism Φ and B can be a discrete-event model in formalism Ψ. This enables composition of the plant and the controller models to be specified with the DEVS formalism, for example.

Unlike the above scenario, it is possible for a super modeling formalism to support model specifications that are at the same level of abstractions. Super formalism can support composition of state-based models rather than composing a state-based model and an input/output model (Fishwick 1992, Zeigler et al. 2000). This is a *strong* form of super-formalism since it supports composition of different kinds of model specifications and their composition at the level of state-based specification rather than input/output. For example, the DEVS formalism can support combined DEV&DESS modeling (Zeigler 2006) – i.e., model transformation (and thus user-defined model encapsulation) is unnecessary. Another approach is Ptolemy which supports modeling of mixed signal systems (Eker et al. 2003). It formalizes input/output interactions among actors (model components) under the control of directors.

5 META AND POLY MODEL COMPOSABILITY

A model can also be composed from models that are described according to two or more modeling formalisms. In the meta modeling formalism approach, different types of model specifications are transformed or abstracted to another modeling formalism. The meta modeling formalism must be able to account for the differences between disparate model specifications that can be composed.

As depicted in Figure 4(a), $M_{\hat{A}\cup\hat{B},\{\Theta\}}$ is a composition of models with their interactions specified in formalisms Θ . Here models A and B specified in Ψ and Φ are mapped to $\hat{A}$ and $\hat{B}$. This requires $\Psi \rightarrow \Theta$ and $\Phi \rightarrow \Theta$. The transformation $\rightarrow$ defines structure and behavior of formalisms Ψ and Φ to that of the formalism Θ – i.e., the interactions between models A and B are specified in terms of $\hat{A}$ and $\hat{B}$.

A standardized modeling approach as shown in Figure 4(a) is the High Level Architecture (HLA) (Dahmann et al. 1999, HLA 2000a, HLA 2000b). It provides a suite of generalized services where different simulation models are mapped to. This approach is strongly aimed at handling interoperability needs with some limited capability for model composability as supported by the Object Model Template (HLA 2000c). The interaction of the different types of model specifications is supported with the publish/subscribe technique and data management service while handling of timed interactive behavior of execution algorithms is the responsibility of the time management service (Allen 1997, Fujimoto 1998).

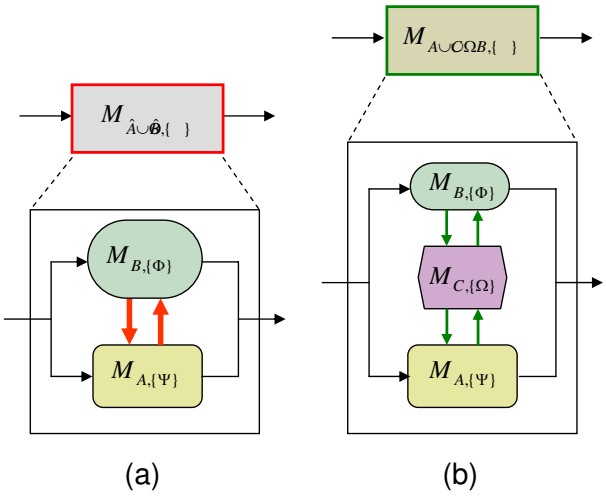

Figure 4: (a) Meta and (b) Poly Modeling Formalisms

Another approach is to use meta-modeling and model transformation (Jaramillo, Vangheluwe, and Alfonseca 2002). In this approach, meta-modeling allows determining whether or not two models described in different modeling formalisms can be transformed completely or partially de-

scribed within a meta-modeling formalism. In some cases a model described in Ψ can be transformed completely into another model described in Φ (without any loss in model dynamics) whereas in other cases some constraints must be defined and placed on models described in Ψ to be represented in Φ. For example, a discrete-time model specified in Discrete Time System Specification and a discrete-event model specified in Petri-nets can be completely transformed into models described in Discrete Event System Specification (Vangheluwe 2000).

Given the semiconductor manufacturing supply chain system, its manufacturing process and logistic control can be specified in the DEVS and LP modeling formalisms (denoted as Ψ and Φ in Figure 4(a)). The composition of the meta models of A and B is denoted as $M_{\hat{A}\cup\hat{B},\{\Theta\}}$. The composite (or federated) DEVS and LP models may be specified in HLA (denoted as Θ in Figure 4(a)).

The remaining approach, called poly modeling formalism (or multi-formalism modeling), composes disparate models using a model (referred to as Knowledge Interchange Broker) which handles the differences between disparate *modeling formalisms* (Sarjoughian and Plummer 2002). The KIB formalism enables composing model specifications and execution algorithms of disparate modeling formalisms. The concept and formulation of KIB was developed in the context of a simple intelligent transportation system (Sarjoughian and Plummer 2002, Sarjoughian and Huang 2005). As shown in Figure 4(b), $M_{A\cup C\cup B,\{\Omega\}}$ is the composition of $M_{A,\{\Psi\}}$ and $M_{B,\{\Phi\}}$, using $M_{C,\{\Omega\}}$. The interactions between models A and B are specified in Ω. The KIB approach in realistic semiconductor manufacturing supply chain systems has been realized – i.e., detailed models described in the DEVS, LP, and MPC modeling formalisms are developed and composed with $KIB_{DEVS/LP}$ and $KIB_{DEVS/MPC}$ (Godding et al. 2004, Sarjoughian et al. 2005, Huang et al. 2006). These introduce modeling capabilities to the KIB that can support discrete-event and optimization model interactions that are essential in modeling of discrete part manufacturing supply chain systems.

Unlike the meta modeling formalism approach, in poly model composability approach models are not transformed to a set of models all of which are described in accordance to a single modeling formalism. Furthermore, poly modeling formalism is distinct from the strong form of super formalism as KIB can both support the interactions (data and control exchanges) among different model types (e.g., discrete-event and linear optimization) and also support different kinds of data transformation and control schemes that are described external to the models that are composed. Using the poly model composability approach, common forms of data transformation (i.e., aggregation and disaggregation) is inherently supported. Furthermore, it supports alternative forms of common control schemes (i.e., sequential, synchronous, and asynchronous). These general data transformation and control scheme concepts need to be extended based on individual the application domains that are may be modeled.

6 MODEL SPECIFICATION AND EXECUTION ALGORITHM

In the previous section, the composability modeling approaches were described in terms of model specifications. However, as noted earlier, a formalism is defined as a pair. The separation of model specification and execution algorithm enables model specification composability and execution algorithm interoperability differently depending on the composability approach (Sarjoughian and Plummer 2002; Godding, Sarjoughian, and Kempf 2004; Sarjoughian and Huang 2005; Huang et al. 2006). As shown in Figure 5, the former is concerned with composition of syntax and semantics of different formalisms, whereas the latter is concerned with execution protocols and their interoperation.

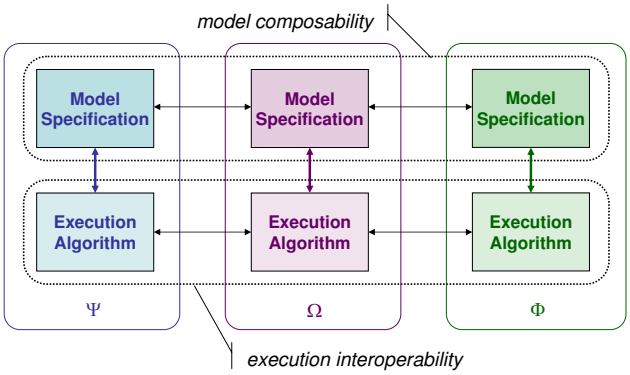

Figure 5: Separation between Model Specification and Execution Algorithm

In the mono composability approach, one execution protocol is used for a set of models that are all defined using a single model specification approach. In the super composability, execution algorithms of the enclosing models and those that are encapsulated are interoperated under the super formalism's execution algorithm. That is, the execution algorithm of the model that is encapsulated in super formalism is viewed as an atomic operation within the super formalism execution algorithm.

The meta composability approach, in contrast to super composability, is primarily concerned with interoperability. The execution algorithms of the disparate model specifications are cast into model constructs and operations (i.e., a set of services as in HLA) that are intended for interoperating different execution algorithms. In this approach, model composability is not strongly supported and the model specification and execution algorithm are weakly separated from one another.

The poly composability approach, unlike the other approaches, is concerned with both model composability and simulation interoperability (Sarjoughian and Plummer 2002). As shown in Figure 5, not only are modeling specifications of Ψ and Φ composed using the model specification of Ω, but the execution algorithm is responsible for interoperability between the execution algorithms of Ψ and Φ.

In the domain of semiconductor manufacturing supply chain systems, Ψ can be DEVS, Φ can be LP or MPC, and Ω can be $KIB_{DEVS/LP}$ or $KIB_{DEVS/MPC}$ (Godding et al. 2004, Sarjoughian et al. 2005). In the domain of intelligent transportation systems, Ψ can be DEVS, Φ can be RAP (Firby and Fitzgerald 1999), and Ω can be $KIB_{DEVS/RAP}$ (Sarjoughian and Plummer 2002, Sarjoughian and Huang 2005).

This separation between *model composability* from *simulation interoperability* is key given different application domains (Sarjoughian and Plummer 2002). For example, the $DEVS/LP$ or $DEVS/MPC$ model composability approach (Godding et al. 2004, Sarjoughian et al. 2005, Huang et al. 2006) is based on the manufacturing process (plant) and decision (control) models to interact with one another based on a synchronous control scheme consistent with actual operation of a semiconductor manufacturing supply chain system. In contrast, the $DEVS/RAP$ model composability approach (Sarjoughian and Huang 2005) is based on asynchronous interaction between plant and control models. The differences between control schemes defined for interactions in the KIBs play a key role in poly model composability approaches given not only different classes of modeling formalisms, but also different application domains.

With poly model composability shown in Figure 5 where only two modeling formalisms are composed, another type of model (e.g., RAP) that is not well suited to be specified in Ψ (e.g., DEVS) or Φ (e.g., LP) needs to be cast into either Ψ or Φ. This suggests, when more than two modeling formalisms need to be composed, multiple or hierarchical KIBs are needed or alternatively a combination of super, meta, and poly formalism may be used instead.

A key practical distinction between the meta and the poly composability approaches is that the latter can systematically support composition of syntax and semantics of known disparate modeling formalisms – it provides mappings and rich data transformations. For example, the differences between DEVS and LP/MPC syntax and semantics is supported with a suite of general mappings and domain-specific data transformations. This is in contrast to the meta composability approach where differences between different types of models can be partially accounted for in the meta modeling formalism, thus requiring all remaining differences to be handled on a case-by-case basis. This is because models described in different modeling formalisms need to be augmented to handle differences that cannot be handled via meta modeling and interoperability regimes.

A common approach to overcoming differences between different types of models is illustrated in Figure 6. Adapters can be used to facilitate model interactions in the absence of super, meta, and poly formalisms. Here an *adapter* is needed to transform the data that is specified in Φ but also needs to be used with models specified in Ψ. For each modeling effort, this approach relies on uni- and bi-directional data and control in terms of software design techniques and their implementations developed. This approach requires defining data mappings and transformations individually for every model that is described either using Φ or Ψ. With this approach, individual adapters need to be defined separately. The consequence is that consistency among the definitions of the adapters must be maintained for all models manually – i.e., data mappings and transformations must be handled for every set of models that are defined in disparate modeling formalisms on a case-by-case. Furthermore, the control of execution between execution algorithms must be handled as part of the modeling effort instead of making use of well-known sequential, synchronous, or asynchronous execution schemes.

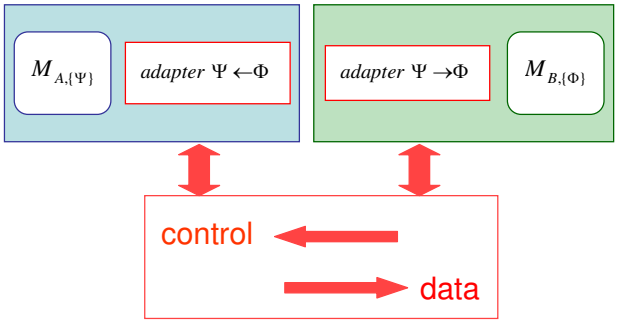

Figure 6: Model Integration with Adapters

For example, given the DEVS, RAP, LP, and MPC formalisms, the models described in them can be augmented with adapters written to handle the disparities among them. For example, while inputs and outputs of DEVS model specification are messages, the inputs and outputs of MPC model specifications are primitive data types (e.g., integer and string arrays). This requires customizing DEVS and MPC models to handle data received for every model. This is in contrast to a modeling formalism that supports all models that can be described in the DEVS and MPC modeling formalisms. The aggregation and disaggregation of data and alternative control schemes can be supported at the level of model specifications instead of using adapters or interoperability services. For example, data and control interactions between the manufacturing process (plant) and the optimization control (controller) models are handled

using modeling constructs that are given in the KIB. With the poly model composability formalism approach, reuse is supported at the level of modeling formalisms.

6.1 Role of Domain Knowledge

A general purpose modeling formalism supports describing and executing a model; it is not intended to account for domain-specific modeling. Therefore, until now, model composability has been considered independent of the application domains to which it may be applied. However, given the central role domain knowledge plays in developing composable models, it is important to consider it for model composability. Domain-neutral modeling formalism is key for modeling specific systems (application domains). This is because the complexity of composite models is in part due to the application domain that is being modeled. For example, discrete-event modeling can be used in the domains of manufacturing, information systems, and event-based control. The interactions to be modeled among disparate models need not only to capture data transformations and executions between composed models, but also be appropriate to the system being modeled since general purpose modeling formalisms are void of domain knowledge. For example, data transformation between a discrete-event model and an LP optimization model of a manufacturing process has to handle a list of products that are specialized to hold Finished Goods having some defect distribution (Godding et al. 2004, Huang et al. 2006). The frequency between the process and controller models can also depend on the domain – e.g., interactions can be sequential or synchronous.

Domain specific modeling is not only central to the mono and the super model composability approaches, but also the meta and the poly model composability approaches. Separating domain-neutral and domain-specific model interactions is also key for handing general purpose modeling concepts while extending them with specific needs of a domain. Given the separation between model specification and the execution algorithm of a modeling formalism, the model specification can be extended to support domain knowledge. In the case of the mono and the super formalism model composability which are based on the concept of components, general purpose model components can be specialized for a given domain. For example, DEVS model components are specialized to the entities of a manufacturing process network. In the case of the meta model composability and HLA, in particular, domain specific federates are modeled using specialization supported by object-oriented concepts and methods.

In addition, data engineering (XML family of data representations and ontologies) may also be used for handling static (i.e., non-behavioral) domain-specific knowledge (Tolk and Diallo 2005). For poly model composability, due to strong separation between modeling formalisms that are composed, existing approaches to domain-specific modeling may be used. The existence of the KIB offers a basis to explicitly account for interaction of domain knowledge.

Given the composability approaches, each modeling formalism provides a different degree of support for describing domain knowledge. The degree of support for domain-specific modeling is grounded in the kind of model composability that is enabled in each approach.

6.2 Distributed Execution of Composable Models

Another consideration for model composability is distributed execution of execution algorithms and their interoperation. All of the composability approaches lend themselves to distributed execution. For example, HLA supports distributed execution of federates and federations (Fujimoto 1998). These allow distributed execution of models synthesized using any of the mono, super, meta, and poly model composability approaches.

To support distributed execution of models, it is important to develop software design and implementation that account for efficient data exchanges and overall execution speed. Given disparate model specification and execution environments (e.g., DEVSJAVA (ACIMS 2001), Opl-Studio (ILOG 2005), and Matlab/SIMULINK (Mathworks 2005)), other considerations (e.g., scale of individual models and data exchanges, frequency of model interactions, and length of experiments) that can affect execution of combined models must be accounted for. This is because the separation of model specification and execution algorithm can adversely affect execution of the composed models. Nonetheless, given sound software designs and implementations, models that are specified with an eye on simple model designs and appropriate use of programming constructs can be executed efficiently.

7 CONCLUSIONS

A classification of model composability approaches is presented. Formulation for each of the model composability approaches is described in terms of modeling formalisms. These approaches are described from a multi-layer modeling vantage point to highlight the importance of using different modeling formalisms. How disparate modeling formalisms affect model composability is described using a simplified example from the domain of semiconductor supply chain systems. The discussion on separating model specifications and execution algorithms reveals the impact of model composability choices and in particular support for varying degrees of model compositions and consequently limitations and complexity of specifying disparate composite models, level of support for domain-specific modeling, and degree of support for distributed execution.

ACKNOWLEDGMENTS

The author acknowledges support of this work by Intel, Lockheed Martin, and the National Science Foundation. Thanks are due to Dongping Huang, Gary Mayer, Hans Mittelmann, Daniel Rivera, and Wenlin Wang of Arizona State University, Gary Godding, Karl Kempf, and Kirk Smith of the Intel Corporation, and Steven Hall of Lockheed Martin Space Systems for stimulating discussions on integrating agent, discrete-event, linear optimization, discrete-time, and cellular automata models in the domains of semiconductor manufacturing supply chain systems, system of systems, and human landuse studies. Acknowledgement is due to Paul Davis of the Rand Corporation for fruitful discussions on simulation composability and a review of an earlier version of this paper. Special thanks to David Goldsman of Georgia Tech for his encouragement and support of this paper.

REFERENCES

ACIMS. 2001. *DEVSAJVA, Arizona Center for Integrative Modeling and Simulation*. Available from <http://www.acims.arizona.edu/> [cited 2006].

Allen, R. 1997. A formal approach to software architecture. Ph. D. thesis, Computer Science Department, Carnegie Mellon University, CMU-CS-97-144.

Banks, J., J. Carson, B. Nelson, and D. Nicol. 2004. *Discrete-Event System Simulation*. 4th ed. Prentice Hall.

Barros, F. 2002. Modeling and simulation of dynamic structure heterogeneous flow systems. *Simulation: Transactions of the Society for Modeling and Simulation International* 78 (1): 18–27.

Barros, F., and H. Sarjoughian. 2004. Guest editorial: Component-based modeling and simulation. *Transactions of The Society for Modeling and Simulation International* 80:319–320.

Cellier, F. 1991. *Continuous system modeling*. Springer Verlag.

Chow, A. 1996. Parallel DEVS: A parallel, hierarchical, modular modeling formalism and its distributed simulator. *Simulation: Transactions of the Society for Modeling and Simulation International* 13:55–67.

Dahmann, J., C. T. M. Salisbury, P. Barry, and P. Blemberg. 1999. HLA and beyond: Interoperability challenges. In *Simulation Interoperability Workshop*. Orlando, FL, USA.

Davis, P., and R. Anderson. 2004. Improving the composability of DoD models and simulations. *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 1 (1): 5–17.

Davis, P., C. Overstreet, P. Fishwick, and C. Pegden. 2000. Model composability as a research investment: Responses to the featured paper. In *Proceedings of Winter Simulation Conference*, 1585–1591. Orlando, FL.

Eker, J., J. Janneck, E. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. 2003. Taming heterogeneity — the Ptolemy approach. *Proceedings of the IEEE* 91 (2): 127–144.

Firby, R., and W. Fitzgerald. 1999. *The* RAP *system language manual, version 2.0*. Evanston, IL: Neodesic Corporation.

Fishwick, P. 1992. An integrated approach to system modelling using a synthesis of artificial intelligence, software engineering and simulation methodologies. *ACM Transactions on Modeling & Computer Simulation* 4 (2): 307–330.

Fishwick, P. 1995. *Simulation model design and execution: building digital worlds*. Prentice Hall.

Fujimoto, R. 1998. Time management in the High-Level Architecture. *Simulation* 71 (6): 388–400.

Fujimoto, R. 2000. *Parallel and distributed simulation systems*. John Wiley and Sons, Inc.

Fujimoto, R., D. Lunceford, E. Page, and A. Uhrmacher. 2002. Grand challenges for modeling and simulation. Schloss Dagstuhl.

Godding, G., H. Sarjoughian, and K. Kempf. 2004. Multi-formalism modeling approach for semiconductor supply/demand networks. In *Proceedings of Winter Simulation Conference*, 232–239. Washington DC, USA.

Hall, S. 2005. Learning in a complex adaptive system for ISR resource management. In *Proceedings of Spring Simulation Conference*, 5–12. San Diego, CA.

HLA. 2000a. *IEEE high level architecture framework and rules — IEEE 1516-2000*. IEEE.

HLA. 2000b. *IEEE high level architecture framework and rules — IEEE 1516.1-2000*. IEEE.

HLA. 2000c. *IEEE high level architecture object and model template — IEEE 1516.2-2000*. IEEE.

Huang, D., H. Sarjoughian, D. Rivera, G. Godding, and K. Kempf. 2006. Flexible experimentation and analysis for hybrid DEVS and MPC models. In *Proceedings of Winter Simulation Conference*. Monterey, CA, USA.

ILOG. 2005. *OPL Studio*. Available from <http://www.ilog.com/products/oplstudio/> [cited 2005].

Jaramillo, J., H. Vangheluwe, and M. Alfonseca. 2002. Using meta-modelling and graph grammars to create modelling environments. *Electronic Notes in Theoretical Computer Science* 72 (3).

Kasputis, S., and H. Ng. 2000. Composable simulations. In *Proceedings of Winter Simulation Conference*, 1577–1584. Orlando, FL, USA.

Kempf, K. 2004. Control-oriented approaches to supply chain management in semiconductor manufacturing. In *Proceedings of IEEE American Control Conference*, 4563–4576. Boston, MA, USA.

Mathworks. 2005. *MATLAB/Simulink*. Available from <http://www.mathworks.com> [cited 2005].

Mosterman, P., and H. Vangheluwe. 2002. Guest editorial. *ACM Transactions on Modeling and Computer Simulation* 12 (4): 249–255.

Page, E., and J. Opper. 1999. Observations on the complexity of composable simulation. In *Proceedings of Winter Simulation Conference*, 553–560. Orlando, FL, USA.

Praehofer, J. 1991. System theoretic formalisms for combined discrete-continuous system simulation. *International Journal General Systems* 19 (3): 219–240.

Qin, S., and T. Badgwell. 2003. A survey of industrial model predictive control technology. *Control Engineering Practice* 11 (7): 733–764.

Rardin, R. 1998. *Optimization in Operations Research*. Prentice Hall.

Sarjoughian, H., and F. Cellier. (Eds.) 2001. *Discrete event modeling and simulation technologies: a tapestry of systems and AI-based theories and methodologies*. Springer Verlag.

Sarjoughian, H., and D. Huang. 2005. A multi-formalism modeling composition framework: Agent and discrete-event models. In *Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real Time Applications*, 249–256. Montreal, Canada.

Sarjoughian, H., D. Huang, W. Wang, D. Rivera, K. Kempf, G. Godding, and H. Mittelmann. 2005. Hybrid discrete event simulation with model predictive control for semiconductor supply-chain manufacturing. In *Proceedings of the Winter Simulation Conference*, 256–266. Orlando, FL, USA.

Sarjoughian, H., and J. Plummer. 2002. Design and implementation of a bridge between RAP and DEVS. Computer Science and Engineering, Arizona State University, Tempe, AZ. Internal Report.

Sarjoughian, H., and B. Zeigler. 2000. DEVS and HLA: Complementary paradigms for modeling and simulation? *Transactions of the Society for Modeling and Simulation International* 17 (4): 187–197.

SBA. 1998. Simulation based acquisition: a new approach. Defense Systems Management College. Report of the Military Research Fellows DSMC.

Singh, R., H. Sarjoughian, and G. Godding. 2004. Design of scalable simulation models for semiconductor manufacturing processes. In *Proceedings of the Summer Computer Simulation Conference*, 235–240. San Jose, CA, USA.

Tolk, A., and S. Diallo. 2005. Model-based data engineering for web services. *IEEE Internet Computing* July/August:54–59.

van Beek, D., S. Gordijn, and J. Rooda. 1997. Integrating continuous-time and discrete-event concepts in modelling and simulation of manufacturing machines. *Simulation Practice and Theory* 5:653–669.

Vangheluwe, H. 2000. DEVS as a common denominator for multi-formalism hybrid systems modelling. In *IEEE International Symposium Symposium on Computer-Aided Control System Design*. Anchorage, Alaska: IEEE Computer Society Press.

Wang, W., D. Rivera, and K. Kempf. 2005. A novel model predictive control algorithm for supply chain management in semiconductor manufacturing. In *American Control Conference*, 208–213. Portland, OR, USA.

Wang, W., D. Rivera, K. Kempf, and K. Smith. 2004. A model predictive control strategy for supply chain management in semiconductor manufacturing under uncertainty. In *American Control Conference*. Boston, MA, USA.

Wymore, A. 1993. *Model-based systems engineering: an introduction to the mathematical theory of discrete systems and to the tricotyledon theory of system design*. Boca Raton:CRC.

Zeigler, B. 2006. Embedding DEVS&DESS in DEVS. In *DEVS Integrative Modeling & Simulation Symposium*, 125–132. Huntsville, AL, USA.

Zeigler, B., H. Praehofer, and T. Kim. 2000. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. 2nd ed. Academic Press.

AUTHOR BIOGRAPHY

HESSAM S. SARJOUGHIAN is Assistant Professor of Computer Science and Engineering at Arizona State University, Tempe and Co-Director of the Arizona Center for Integrative Modeling and Simulation. His research includes simulation modeling theories and methodologies with emphasis on multi-formalism model composability, visual component-based system modeling, collaborative modeling, co-design modeling, agent-based modeling, software architecture, and distributed simulation. His educational activities have led to the establishment of an Online Masters of Engineering in Modeling & Simulation in the Fulton School of Engineering at ASU. For more information contact the author at <hss@asu.edu> or visit <http://www.eas.asu.edu/~hsarjou/index.htm>.