Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

HETEROGENEOUS MODELS IN A MULTI-MODEL SYSTEM

Charles Turnitsa

Information and Communications Laboratory
Georgia Tech Research Institute
305 Fifth Ave.
Quantico, VA 22134, USA

ABSTRACT

Multi-model systems may arise in a number of configurations. Other configurations are also possible. In these cases, however, a persistent problem that exists is one of heterogeneous models (and simulators based on them) having to exchange information with each other, but that differences exist in how the individual models represent information about the systems they are describing. A chief distinguishing feature is simply the difference between the representation of time and process flow – the difference between discrete time models and continuous time models. Setting aside that difference, there are other dimensions of heterogeneity that may be enumerated. These are presented, along with indicators of some current examples/approaches to mitigate the differences.

1 INTRODUCTION

In the literature dealing with military modeling and simulation, as well as in the literature dealing with model interoperability, it has long been remarked on the situation that many different model based systems exist that have representation of similar referents to other model based systems. Beyond the domain of military modeling and simulation, this is a research area that has great impact on multi-model solutions. It is often the case that such systems may be expected to share information with each other, or interact in some other way as part of a multi-model solution. The challenge has always been that models, as they are first conceived of by the modeler, and then as they are used as the basis for implementation, have differences in their internal representation from each other. This exists with regards to the objects that are defined in such models, as well as the processes and algorithms that dynamical models describe. But also, when such models serve as the basis for simulators or other model based systems that time based execution, there is a particular possibility for differences in how time is handled, and even if time is thought of as a discrete structure, or a continuous structure. This paper represents an effort to describe certain enumerated differences, in a distinct theoretical sense, and then see how that applies to several recent examples.

The author contributed to an earlier paper, where such differences were mentioned as the dimensions of difference between models, however while the dimensions themselves were enumerated, the specific description of how such differences affect multi-model solutions was not detailed (Turnitsa and Tolk 2008). And as a multi-model solution may incorporate multiple models within different simulators, or within the same simulator, the interpretation of the differences in both cases are also described here. The purpose of this enumeration is one of survey, to incorporate other references and efforts that also deal with these differences, but also as one that lays out the theoretical differences, as a basis for future work on general case solutions (if such are possible).

2 MODEL COMPOSITION

In order to discuss heterogeneous models, comparison of the models and their composition may be required, in order to describe specific differences. One rational method for describing models is to capture an enumeration of the set of all components that make up a model – treating a model as a set of constituent components. In such a way, specific claims can be made when making the comparison of two models.

2.1 Model Components

The suggestion that a model can be described by enumerating its constituent components is based on the proposal that such components can be known about a model, and that a model's identity is capably described by such an enumeration, that it can be compared to other models.

The first proposal – that the components of a model can be known – is somewhat inherent in the practice of modeling. A language to describe such components can be thought of as a meta model, in the language of Henderson-Sellers (2012), which offers the definition that a meta model is itself an abstraction, used to describe models, with the specificity of a defined language. It should be relied on a language that offers up a definition of types, rather than more general instances, and it should provide enough definition that any models described, can be fully defined. An example given by Seidewitz (2003) is that the UML specification, for instance, is a meta model that describes, by defining types, all of the constituent components that make up any specific UML model.

The second proposal – that a model's identity is capably described by such an enumeration, is suggested by and accepted within the work on model composability and efforts to use a variety of different types of models to compare and describe interacting models. In the area of composability, the various strata found within the levels of conceptual interoperability model (LCIM) are predicated on, and have been shown to be useful, in modeling models well enough such that analytical (descriptive) and synthetic (predictive) findings about models, and their interoperability, can be made. The LCIM was first introduced by Tolk and Muguira (2003), but was subsequently expanded by Turnitsa (2005). Implications of how the LCIM can describe other models, and its strength in doing so can be found in Tolk, Turnitsa and Diallo (2008). Modeling the comparative meaning of models, however, is also an undertaking that is accepted within the literature. This can be seen in Yilmaz and Paspuleti (2005) and also in the literature on using ontologies to describe and define models, such as the work by Guizzardi and Wagner (2010) to describe discrete event simulation languages, with an ontological model.

2.1.1 OPR

The method used here, to describe the components of models for comparison purposes, will be the object, process, relation (OPR) method. Other methods would be possible, but OPR is used here because of the familiarity of the author, who first derived it for Turnitsa (2012). It has since been described as a method to describe conceptual models (Tolk and Turnitsa 2012) and also explored as the means for representing characteristics of process within models (Turnitsa and Tolk 2013). Alternate formal methods that could also be used, with equal representative strength include the set of theoretical statements describing how set theory could be applied to the modeling of constituent objects within a model (equations 2.1 through 2.16), and the definition of processes or functions (equations 2.17 through 2.22), and finally the definition of relations or relationships (equations 2.23 through 2.52) in Henderson-Sellers (2012). Equally, situation theory could be relied on (Devlin 2006), with its defined methods of describing a situation that might appear in a discrete sequence, or a continuous sequence, through the use of formally defined components (temporal reference, spatial reference, situation, infon, parameter). However, here OPR will be relied on, again, because of familiarity.

OPR is a method for describing the constituent components of a model by treating, as a set, the enumerated objects and process that make up that model, and also determining their ordering by describing the relations between those other components. It uses types in order to capture the characteristics of instances within the models, but does not attempt to describe the comparative relationship between such defined instances between two models. As such, an OPR description of one model would be necessary to compare it to an OPR description of another model. Then the differences between the two models would be possible. A further limitation of the OPR method is that it does not evaluate or describe the particular presentation of a model. It is instead focused on description of the knowledge that the model is trying to convey through the presentation, not the presentation itself.

As mentioned, OPR has been defined and described to prior meetings of the Winter Simulation Conference, but a brief review is presented here for the convenience of the reader. The three types of components describe below all have further definition by being associated with defining qualities. Thus, within a particular model there may be objects (one of the component types), and each individual *object* will be further defined within that model by *attributes*. Likewise with processes, each instance of a *process* would be defined by a set of *characteristics*. Finally, each model is likely to have one or more defined *relations*, and each would be defined by a set of *rules*.

- **Objects** are any components (declared or implied) that have a stable, continued existence – until operated on by a process. They are the things that can be observed independently within the Model, with identity separate from all other components. Giving them definition, however, are their attributes. These give the objects their qualitative and quantitative distinction from other objects.
- **Processes** are the components of a model that describe any change that the model might imply. What is being changed is not limited (it is typically an object, but could equally be another process or a relation). The defining qualities of processes, as they are definable by the OPR method, are referred to as characteristics. These define the process by illustrating what is being changed, and the definition of when and how that change takes place (for instance, is an entire object changed, or just one or more of its attributes). A treatment of how processes are defined, along with their characteristics, has been presented to the Winter Simulation Conference community previously (Turnitsa and Tolk 2013).
- **Relations** are the means for associating together the constituent components of a model. This is to allow objects to be associated with each other, or with processes, or process with each other. Relations, of course, can also describe associations of other relations. The defining qualities of each relation are known as the rules of that relation, and give it definition (such as when the relationship actual rather than potential, and so forth).

It has been shown that as a metamodeling technique, OPR is capable of exhibiting information about a wide variety of different modeling languages, and the models that they describe.

2.1.2 Topology

The interaction of the components of a model, when described using OPR, describe any categorical or semantic groupings within a model. While such a situation might be clear from the graphical presentation of a particular model, the use of OPR with the specific representation of associations through defining the relation component makes it possible to compare such groupings between models of different types. There are several reasons why knowing about such groupings may be valuable to understanding what a model is representing. Some of these reasons include (1) A process (which might be an algorithm, activity, function, or other method describing how change occurs within the model) will have as it's modeled effect not only one attribute of one object, but potentially many attributes in a categorical grouping.. Similarly, many attributes of many objects (or whole objects) may be affected with a single modeled operation of the process; and (2) associative relations can be used to group together several sequential processes, that may be useful to model separate, but which often function together as a group.

This associative grouping, visually apparent in a graphical model, is described in OPR by the technique of explicitly defining the relation component. Within a graphical model, the differences between different models that have different topologies (different associative structure of otherwise similar objects and processes), is again apparent visually, but when using a formal model description, such as OPR (or other formal techniques), those associations, and the resulting topology, become components of potential difference, as much as differences in the structure of objects and/or processes is a part of the measurable difference between models. This is described here, because it is a basis for the dimensions of difference described below.

2.1.3 Notable Characteristic - Time

As mentioned in the introduction, one of the first types of differences remarked on in different models has to do with differences in how time is handled, especially for the modeling of processes. This could include using discrete structures in a time model, where processes are represented by time discretized type presentation, of approximations of a processed change at discrete time intervals (some referents readily are very appropriate for this type of modeling, as discrete event situations, where identifiable states are very distinct and discrete from each other). Other modeling techniques take an approach that is closer to performing a differential equation type solution, where the changes resulting from modeled processes are presented as a formulaic change, that continuously takes place over time. Automata are configured to apply a number of different techniques to make this continuous type modeling as smooth as possible, using approximations such as a Runge-Kutta method, where values that result at some point along the continuous timeline are calculated using approximation and interpolation resulting from differential representation.

In a hybrid model that may include both of these situations, it is the case that two different models exist the discrete model, describing the portion of the simulation that are described by discrete time structures, and the continuous model, describing the portion of the simulation that are described by continuous time structures. In this case, although the differences between the two modeling approaches are solved within the hybrid model, when this type of situation is presented in a multi-model situation, then deriving a solution of how interaction with other models takes place, becomes an issues of being aware of not only the differences within the hybrid simulation, but also the implied differences between shared information with other models.

Time representation, however, is still a problem to be dealt with, in situations where the solution includes models of a similar type. When dealing with models that both have discrete time representations, then the issues that arise have to do with representation of the discretized time model. There are problems when there is a heterogeneous mixture of discrete time models with discrete event models. But there are also potential problems even when both models are either discrete time or discrete event. Some of the possible questions that arise, that must be dealt with when dealing with merged time models, are described here. Do both models subscribe to a similar event list in showing the change in discrete times? If not, how is the difference handled where processes that might, at an event-point in time, result in a discrete state, are compared at different times in between their states? Is the last state the value used? Is the next state the value used? In the case where the prior state is used to determine values, this might be a pre-fix solution, where all data remains the same until after the event transpires. If there is a need to look ahead, with pre-simulation, and use the values that will result once the next state is arrived at, this might be a post-fix type situation. Finally, if there is an attempt, as with continuous time models, to interpolate the values between states, then this might be referred to as an in-fix solution to value determination. Knowing how individual models handle these time differences, even when both models are discrete (whether discrete time, discrete event, or a mix), is something that can be helped by having a formal representation of the models in question.

3 MODEL DIFFERENCES

Beginning with the consideration of a model as a collection of different components, it follows that if such models can be described, that a definition of differences between models can be derived and described. Many of these differences have become apparent in exercises to apply multi-model systems in the past, and indeed some of them have a broad body of research that has been developed, in terms of providing general case solutions. The presentation here, is to give definition to the dimensions of difference mentioned in earlier publications (Turnitsa and Tolk 2008) and also to provide a more formal description of those differences, given a formal method such as OPR to describe them.

Previous efforts have been introduced to describe the differences between models, including a taxonomy of such model combinations presented in Lynch and Diallo (2015). From various domains, there have been efforts to discuss multi-model efforts in manufacturing from Rabelo et al. (2003), logistics from Lee et al. (2002), and healthcare from Chahal and Eldabi (2008). A representation of the impact on such multi method modeling approaches is discussed in Balaban and Hester (2013). Building on those efforts, the presentation of the types of differences here is done with respect to how the modeling approaches, and the resulting models, can be understood from a systemic perspective by applying a common component framework to describing the models as well as the differences. This is the reason for introducing OPR into this conversation, and it is applied as the system framework to enable the identification of differences.

Before proceeding with the differences, it is useful to keep in mind that we are talking here with models that have some overlapping requirement to describe some, or all, of the same referent. Models are expressions of description, and what they are describing is a referent. When models have to exchange information, because they are composed with each other in a multi-model solution, then those models necessarily have overlapping representation of at least some part of the same referent. These dimensions of difference are attempts to understand, and describe, how different models describe the same referent, in different ways. These dimensions of difference are typically not applicable to models that have no overlap in the referent systems that they are describe, nor is it to be expected that any such models would ever interact in a multi-model solution.

3.1 Static Differences

This first group of differences described are referred to as static differences. The basis for this name is because these differences have to do with a static assessment of the models in question, not because it is only applicable to static models. The differences here are based on specifically how models internally have differences in how they represent the referent system they are describing. These differences can typically apply to any models that might have to exchange information, whether or not they will exist within a simulator or some other executable system. For that reason, these differences are applicable to things such as data models or descriptive models, as well as conceptual models and simulation-models that might serve as the basis for simulator software.

3.1.1 Multi-Resolution

The first of the static differences between models is the multi-resolution model difference. This refers to differences in how models, which are each typically describing the same referent, are doing so at different resolutions of detail. An early, and still influential work, on the topic is (Davis 2000).

A very common example of this is in military modeling, where two different combat simulators may be based on different resolution models. In such an example case, simulator ***L*** might have internal model representation of a battlespace at a battalion level. It represents, and internally has objects and processes, that model combat from the perspective of treating battalions as the maneuver elements, and having processes that affect battalions. The characteristics and behaviors of the elements that make up the battalion, are abstracted into the overall modeled characteristics and behaviors of the battalion itself. If such a simulator has to interact with simulator ***H***, and that second simulator is derived from a model that represents the battlespace at an understanding of individual entities, such as individual soldiers and vehicles, rather than formations such as platoons and companies, then the resolution of the objects and processes will be different from those in simulator ***L***. In this case, the model in simulator ***L*** is considered to be of a lower resolution (i.e. – the internal details of the referent system are at a higher level of aggregation), than the model in simulator ***H***.

Internally, each of the models in a multi-resolution model system have distinctive components, meaning that the objects, processes and relations of the model in simulator ***L*** are distinctive (and different, in nature) from the components that make up the model in simulator ***H***. The only overlap, is that they are based on different descriptions of the same overall referent. Resulting problems might be that the lower resolution system (the model in ***L***) is representing information that might be referred to as emergent, through its abstraction and combination of the overall effects of the behaviors of the lesser aggregated represented components (objects, processes and relations) within the model of simulator ***H***. Problems that arise come from the understanding of, and assumptions about, what behaviors emerge at the aggregated representation of entities making up the referent. If two such models have to interact, then translation between the representation of components and defining qualities between the two different "resolutions" must be accommodated in the interaction. For instance, the objects and processes of the higher resolution system, ***H***, might mostly fall within the assumptions made within the lower resolution ***L*** about how those objects and processes will behave (assuming that the assumptions made about the referent system are correct), but what is ***L*** to do when it must deal with outlying cases, where some number of specific high resolution components fall outside those assumptions?

In this case as described, one of the details that must be dealt with is how to handle statistical outliers. The data is still valid, but it lies outside the assumptions made. In the language of the example given earlier, of the combat models at different resolution, this would be a case of where the lower resolution system represents the basic location, and are of control, of an aggregated combat unit (such as a battalion). But what happens when it has to deal with representation of data of individual soldiers and/or vehicles that belong to that battalion, but are outside the modeled area of occupation of that battalion. A battalion might encompass 1000 soldiers, and the assumption that the main combat strength is concentrated in certain area is a correct assumption, and the low resolution system handles that correctly. But some number of those 1000 soldiers will be outside that area at any one time. If the two models have to exchange information that involves the data describing those objects and processes, then the interaction technique will have to handle those outliers. As mentioned, there is a broad field of literature already addressing this topic, and much of it delves into the relationship between resolution, fidelity, aggregation and abstraction (Page 2000) (Yilmaz et al. 2007). A specific series of experiments was performed on combat models of differing resolution, and results highlighted the requirement for ameliorating structure and processes to translate not only object representation, but also process execution in Bowers and Prochnow (2003). By defining the different resolution following an OPR understanding of a model's components, the relationship of resolution to fidelity becomes defined as the amount of assumption that must be included into the modeled components. If a model is lower resolution than another model, yet they represent the same referent, then the lower resolution system must necessarily include more assumed representing in its components, as it is not modeling the interaction of a finer level of detail (i.e. – higher resolution representation) of the components making up the same referent.

3.1.2 Multi-Scope

Multi-scope, the second difference has to do with the particular representation of a referent within two different models. This problem arises from the fact that all models are abstractions. They neither cannot, nor should they try, to represent everything that is knowable about a referent. Because of this, it is quite possible that although two different models are representing the same referent system, internally the modelers that developed the representative models, chose different elements to abstract away, in their understanding of, and representation of the referent. Overlap of the models could be expected, because they are representing some overlapping part of the same referent, but the internal composition of each model differs from each other, as represented by the different components and their defining qualities.

The concept of scope is related to, very strongly, the theoretical basis for defining abstraction. Which elements of the referent are abstracted, and in what way are they abstracted. There are, according to the work on the subject by Moon and Hong (2013), three potential relationships between the abstraction of two different models dealing with the same referent. The first is that each model could have the same abstraction, wherein they include the same components representing the referent. The second is that they have an overlapping abstraction, such that the first model includes a number of components that represent the referent, and the second model has a separate set of components that represent the referent, but there are some components in common between the first and second model. The third possibility is that the abstraction in the first model only leaves components left within the model that do not exist at all within the second model. In the term introduced here, in the second case and the third case from (Moon and Hong 2013), there is a difference in scope between the models.

An example of this could be a situation where two different traffic simulators have to be combined into a multi model solution in order to provide information for urban planning. Both simulators represent the traffic patterns in the same urban grid map. One simulator, based on model ***A***, represents the number of vehicles that each part of that grid can handle, on average. The other simulator, based on model ***B***, represents the amount of degradation that the road gets, based on a number of factors. Model ***A*** receives information from model ***B*** on the state of the roads. Model ***B*** receives information from model ***A*** on the amount of traffic each road carries. Each uses the information received from the other in its own calculations, but each also introduces other components into their internal representation that is not part of the exchanged information. So that model ***B*** might include factors in road degradation that include aging materials and infrastructure, weather impacts, financial concerns, etc. And model ***A*** might, in representing average traffic loads, ignore peak traffic times, as they only representing less than 5% of the total time in a week. Assume that the road degradation factors limit the amount of traffic that a road can handle, but model ***A*** is not aware of that, and only thinks that the amount of traffic is based on load (and calculates urban choke points based on that). Also, assume that Model ***B*** understands that the worst degradation occur during those 5% peak times that model ***A*** has factored out of its averages. In both cases, the scope of what was modeled internally was different, even though it would be reasonable to think that they could work together.

In each case, if the final results of the multi-model simulation system have reported outcome, it will be incomplete, and inaccurate. The solution is that when a model is selected to be part of a multi-model solution, the understanding of the model, and how it works internally, is crucial to understanding what must be done to accommodate it in a multi-model architecture. The use of a model as a black box component of an architecture is quite potentially problematic, as we can see from the example. In terms of OPR, although the modeled components that serve as outputs from one model, and inputs from another model, might be based on a very similar component (for instance, the output of an object representing total number of cars, per hour), how that component is used internally in the model (which other objects and processes it interacts with, and which relations associate it together with those other components) is crucial to understanding what will happen to that component once it is received by the other model in the architecture.

3.1.3 Multiple-Structure

Multiple-structure, the third of the static dimensions of difference between models has to do with the topology issue described above, but also in how the internal defining qualities of the components of a model are associated with each other. OPR describes a situation where the objects of a model are given the defining qualities (called attributes) that provide definition for those objects (identity, qualitative and quantitative definition, parametric values, etc). Similarly, processes have the defining qualities (called characteristics) that provide definition of how each process behaves (timing, effects, halting, and so forth). The relations have defining qualities that provide definition as to how each associates together other components (the rules of when the association is in effect, which components are associated with each other, the identity of the association, etc). Frequently, the defining qualities of processes and relations are based on the attributes of objects. Assume, for instance, that all of the same attributes within one model's objects are present in another model's objects, but they are assembled in a different way. This is the essence of the multiple structure difference. The defining qualities of the components is what is important about them, from a data and definition perspective, but what if they are assembled in different ways within different models that have to interact with each other?

Consider a situation where a process in one model, ***N***, must receive information from another model, ***C***, but the exchanged information is just an attribute of a single object. In model ***N***, all of the attributes are combined into objects labeled X_O, Y_O, and Z_O. But in model ***C*** they are found within objects P_O, Q_O, and R_O. The total list of attributes is the same, with the exception of the object identity attributes (the objects' names, if you will), so leaving out those attributes, we have the following list of objects, within each model (***N*** and ***C***), along with their attributes.

Table 1: Example of similar attributes, ordered in differing structured models.

Model ***C***		Model ***N***	
Objects	Attributes	Objects	Attributes
P_O	PX_A, PY_A, PZ_A	X_O	PX_A, QX_A, RX_A
Q_O	QX_A, QY_A, QZ_A	Y_O	PY_A, QY_A, RY_A
R_O	RX_A, RY_A, RZ_A	Z_O	PZ_A, QZ_A, RZ_A

In either case, if an attribute needs to be referenced, or altered, by a process – it is available. However, if there result some changes to entire objects, then the alteration of different sets of attributes results, from the fact that the two models, ***N*** and ***C***, have different internal structuring of the components and their defining qualities.

One method for dealing with this dimension of difference between models, is to ensure that all interactions between the models, and all actions within the models (at the process level) only deal with defining qualities (such as attributes of objects), and not with entire components (such as entire objects). As described earlier, the reason for topological structure in a model, is to ensure that the understood semantic value of the model components are representative of how the describe referent behaves, or is structured. If a referent system, for instance, has constituent entities (for instance, manufactured parts), which are described in a model as objects, and there are processes that either create or destroy those entities as whole items then it makes sense to have modeled processes that affect those objects as contained items (meaning, all of the attributes of an object are affected by a single process), and not dealing with the object at an attribute level.

The technique of dealing with components at the defining quality level, however, may be useful for model interaction. Regardless of the internal structure of either model that may interact with each other (in our example, the two models ***C*** and ***N***), ensure that specific defining qualities are what populates any interaction, even if it means dealing with exchanged component describing data at the defining quality level. In that way, at least for the purposes of the interaction, the assumption can be exposed as much as possible,.

3.2 Dynamic Differences

The second group of differences described here are the dynamic differences. Again, as with static differences, this title does not mean that they are differences that apply to dynamic models, vs. static models, but the name comes from the fact that these differences arise out of how the models are assembled together into a multi-model simulator or simulation event.

3.2.1 Multi-Phase

This dimension is based on the idea that as the processes within a model, as defined by OPR, can have as the effect of their change any other component of the model (including objects, relations, or other processes), that it is possible to have a model that changes its own definition over execution of the model within a simulator. The defined behavior of the model, if thought of as a function, has its definition change over the period of execution, just as a derivative function changes with respect to time. In the case where two or more instances of models might even begin as homogenous models, if they begin with different initial conditions, or different initial input values, then the amount that they will change over time will vary from each other. It is possible that the model, as defined prior to the execution of a simulator based on the model, might be conducive to sharing information with another model, but as the software based on the model is executed, it becomes changed enough so that it is no longer conducive to interoperability.

3.2.2 Multi-Stage

This dimension is again reflective of differences that can arise when the same model is run multiple times, but with a range of values relied on to give definition to the defining qualities of the components making up the model. Consider that a valid representation of a referent could be managed by a model, with one or more of the defining qualities of components in the model being equally valid across a range, rather than as a set point. If a multi-simulation solution is sought, where a variety of different simulation events are run, using the same simulator, but with an introduction of differing values into the defining qualities of the components defining the model the simulator is based on (for instance, the values themselves could be derived by a Monte Carlo technique, or they could be methodically chosen across a range, with a preset interval period between different candidates), then it is possible that the different values chosen for defining qualities could have non-foreseen impact on the outcome of the multi-simulation. An example would be where introducing values in a range that cause the executing simulator to enter a state where its model is no longer a valid representation of the referent. Work defining the concept of a multi-stage, multi-simulation solution is well defined in (Yilmaz et al. 2007).

4 OPR FOR MULTI-MODEL HYBRID SIMULATIONS

If the various differences between models can be identified and described with a method such as OPR, post hoc, then having a systemic framework to identify the components of a model during model development, or during the merging of models into a hybrid solution, may also be possible. Specific methods are not presented here, however two broad types of approaches are described below, that could support efforts in a two different scenarios. The first is a bottom-up approach, where model construction for the purpose of being included in a hybrid solution, is being accomplished. The second is a top-down approach, where the selection and orchestration of merging existing models into a hybrid solution is the effort.

4.1 OPR in Support of Conceptual Modeling

Applying the OPR method of classifying the components of a model can be followed with rigorous attention to specificity and decomposition of all the resultant components, the time model, the semantic representation of objects, the algorithmic details of processes and so on. However, it may also prove useful when thinking of requirements or design of a model, to consider the high level components of OPR as the method for constructing a conceptual model. In this way, understanding how the processes of a model must be supported by the objects, and how these are to be related together, gives some structure and systemic design to the conceptual model.

As highlighted in Turnitsa (2012), and the similar approach in Henderson-Sellers (2012), the understanding of a model as being composed of objects, processes and relations can be exhibited in a number of different conceptual modeling approaches, including graphical approaches. An example of this being applied, although not with the formal nod to OPR or another formalism, was performed as part of the conceptual design of a multi-model hybrid simulation effort reported on in Zabek (1999). In Zabek, the approach to the conceptual requirements for the hybrid solution were conceptualized by defining the elements of the simulation that would be required, and then associating each with processes and objects that would be needed in order to implement such a simulation. These were then gathered, and assigned to appropriate models, in order to build the solution.

4.2 OPR Supporting DEVS Described Hybrid Solutions

One of the applications of the DEVS formalism, is in giving structure and formality to a defined simulation solution, so that understanding exists in when and where discrete changes to objects take place, the internal and external timing models involved and so forth. It has been shown by Ziegler (2006) how this can support a wide variety of different hybrid solutions, composed of models differing even in their basic timing assumptions, such as discrete time, event time, and continuous time models. In all cases, as seen from the perspective of the OPR method of describing models, there are objects that have to be changed, there is a state of the model that results from the individual states of the associated objects, and the changes are discrete processes that result in or more changed objects, or object defining parameters.

These model components (objects, processes, relations) are related to each other through the definition and structure provided by the DEVS formalism. OPR can help, as it clearly identifies the difference between objects that are subject to discrete changes, processes the implement those changes, or relations that define the association of objects and processes – and this can be applied to models that have not be defined by the DEVS formalism. Within such a multi-model hybrid solution, OPR can be relied on to help bring understanding, in a model selection and orchestration effort, to knowing the predicted and defined effects of having the models interoperate. This is seen in both Vangheluwe (2000), and more recently in Camus, et al. (2015).

5 CONCLUDING REMARKS

Relying on a formal method to describe the components of a model (Turnitsa 2012) (Henderson-Sellers 2012) has benefits of expressing one or more models in a technique so that they can be compared to each other. As mentioned in the introductory section, this is extremely important in situations where model interoperability is desired between models. Understanding what are described in this paper as the dimensions of difference between models is a method, along with understanding how the levels of conceptual interoperability model can be used in a predictive case, in order to be able to synthetically predict where model interaction is likely to have problems.

In the case where a single model might be part of a multi-simulation study, insofar that the same model might be executed but with changes to the input values and initial conditions of the model, or specifically to the defining qualities of the components making up the model, then having an understanding of the model ahead of time can also help to synthetically predict where there might be changes to the model's definition, through its own execution as part of a simulator, where the model itself becomes unreliable, invalid, or not recognizable.

Current research and applications of modeling formalisms and understanding the differences between the role of different model components for different user purposes, can also benefit. If a model exists, for instance, of a referent in a particular paradigm but it is to be cast into, or used as the basis for construction of a similar referent based model in a different paradigm (such as an engineering model, being used as the basis for a simulator model, so that simulation execution of the model is possible), then understanding what the components of the original model is composed of, becomes extremely useful in defining the components, and their topology in the new model. For these reasons, it has resulted that taking the defined differences between models that exist within literature, and which have been called out in research and publications previously, and defining those differences using a formal method, helps to understand more specifically what is being described within models.

REFERENCES

Bowers, A., and D. Prochnow. 2003. "Multi-Resolution Modeling in the JTLS-JCATS Federation." In *Proceedings of the 2003 IEEE Fall Simulation Interoperability Workshop*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Camus, B., C. Bourjot, and V. Chevrier. 2015. "Combining DEVS with Multi-Agent Concepts to Design and Simulate Multi-Models of Complex Systems." In *Theory of Modeling and Simulation/DEVS 2015*, International Society for Computer Simulation.

Chahal, K., and T. Eldabi. 2008. "Applicability of Hybrid Simulation to Different Modes of Governance in UK Healthcare." In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, 1469-1477. Piscataway, New Jersey: IEEE.

Davis, P. K. 2000. “Exploratory Analysis Enabled By Multiresolution, Multiperspective Modeling.” In *Proceedings of the 2000 Winter Simulation Conference*, edited by J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 293–302. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Devlin, K. 2006. “Situation Theory and Situation Semantics,” *Logic and the Modalities in the Twentieth Century, Elsevier Handbook of the History of Logic, Vol. 7*, edited by D. Gabbay and J. Woods, Elsevier.

Guizzardi, G., and G. Wagner. 2010. “Towards an Ontological Foundation of Discrete Event Simulation.” In *Proceedings of the 2010 Winter Simulation Conference,* edited by P. Fishwick, A. Coffey, R. Kamhawi, and J. Henderson, 652-664. Piscataway, New Jersey: IEEE.

Henderson-Sellers, B. 2012. *On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages*. New York: Springer.

Lee, Y. H., M. K. Cho, S. J. Kim, and Y. B. Kim. 2002. “Supply Chain Simulation with Discrete–Continuous Combined Modeling.” *Computers & Industrial Engineering* 43 (1–2):375-392.

Lynch, C., and S. Diallo. 2015. “A Taxonomy for Classifying Terminologies that Describe Simulations with Multiple Models.” In *Proceedings of the 2015 Winter Simulation Conference,* edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rosetti, 1621-1632. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Moon, I. C., and J. H. Hong. 2013. “Theoretic Interplay between Abstraction, Resolution and Fidelity in Model Information.” In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S. H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, 1283-1291. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rabelo, L., M. Helal, A. Jones, J. Min, Y. J. Son, and A. Deshmukh. 2003. “A Hybrid Approach to Manufacturing Enterprise Simulation.” In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice, 1125-1133. Piscataway, New Jersey: IEEE.

Seidewitz, E. 2003. “What Models Mean.” *IEEE Software* 20: 26-31.

Tolk, A., and J. A. Muguira. 2003. “The Levels of Conceptual Interoperability Model (LCIM).” In *Proceedings of the Simulation Interoperability Workshop*, 14-19.

Tolk, A., C. D. Turnitsa, and S. Y. Diallo. 2008. “Implied Ontological Representation within the Levels of Conceptual Interoperability Model.” *International Journal of Intelligent Decision Technologies* 2 (1): 3-19.

Tolk, A., and C. D. Turnitsa. 2012. “Conceptual Modeling with Processes,” In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 2641-2653. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Turnitsa, C. D. 2005. “Extending the Levels of Conceptual Interoperability Model.” In *Proceedings of the Summer Computer Simulation Conference*. Philadelphia: Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Turnitsa, C. D., and A. Tolk. 2008. “Knowledge Representation and the Dimensions of a Multi-Model Relationship.” In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. Hill, L. Moench, O. Rose, T. Jefferson and J. W. Fowler, 1148-1156. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Turnitsa, C. D. 2012. *Exploring the Components of Dynamic Modeling Techniques.* Ph.D. Dissertation, Batten College of Engineering, Old Dominion University, Norfolk, VA.

Turnitsa, C. D. 2013. “Representing the Characteristics of Modeled Process,” In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S. H. Kim, A. Tolk, R. Hill, and M. E. Kuhl: 1304-1315. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Yilmaz, L., and S. Paspuleti. 2005. “Toward a Meta-level Framework for Agent-supported Interoperation of Defense Simulations.” *Journal of Defense Modeling and Simulation* 2 (3): 161-175.

Yilmaz, L., A. Lim, S. Bowen, and T. Oren. 2007. “Requirements and Design Principles for Multisimulation with Multiresolution, Multistage Multimodels.” In *Proceedings of the 2007 Winter Simulation Conference*, ed. S. G. Henderson, B. Biller, M.-H. Hsieh, H. Shortle, J. D. Tew, and R. R. Barton, 823–832. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Vangheluwe, H. 2000. “DEVS as a common denominator for multi-formalism hybrid systems modelling. In *IEEE International Symposium on Compuater-Aided Control System Design, 2000 (CACSD’00*), 129-134. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zabek, A. 1999. “Lessons Learned from the Design and Execution of a Federation for Joint Experimentation,” In *Proceedings of the 1999 Winter Simulation Conference*, edited by P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, 1032-1038. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zeigler, B. P. 2006. “Embedding DEV & DESS in DEVS.” In *Proceedings of DEVS Integrative M&S Symposium*: 125-132, Huntsville, AL, USA.

AUTHOR BIOGRAPHY

CHARLES D. TURNITSA is a Senior Research Scientist within the Information and Communications Laboratory of the Georgia Tech Research Institute. He currently serves as the Modeling and Simulation Research Scientist at the GTRI Quantico Modeling and Simulation Facility, and also as a faculty member in the GTRI Applied Systems Engineering program. His research interests are in areas of model semantics, and the use of models to represent knowledge. His email address is cturnitsa@gmail.com.