An Analysis of the Cost of Validating Semantic Composability*

Claudia Szabo and Yong Meng Teo
Department of Computer Science
National University of Singapore
Computing 1, 13 Computing Drive
Singapore 117417
claudias@comp.nus.edu.sg; teoym@comp.nus.edu.sg

Abstract

Validation of semantic composability is a non-trivial problem and a key step in component-based modeling and simulation. Recent work in semantic composability validation promise to reduce verification, validation, and accreditation efforts. However, the underlying cost of current validation approaches can undermine the promised benefits, and the trade-off between validation accuracy and validation cost is not well understood. In this paper we present, to the best of our knowledge, the first quantitative study on the cost of validating semantic composability. Firstly, validation approaches are categorized into techniques that validate general model properties, and that compare the composed model execution with a reference model. Secondly, we focus on two key factors that influence validation cost: characteristics of the simulation problem and the validation approach adopted. Our study covers four representative validation approaches, two DEVS-based approaches, the Petty and Weisel formal validation, and deny-validity, and for simplicity, we use computation time as a measure of validation cost. Based on a queueing model with 1,000 components, the cost ratio between validating general model properties and model execution is 55:45. A 10% increase in the complexity of the composition structure, such as fork and join component interconnections, increases validation cost by more than half. In model execution validation, there is a trade-off between validation accuracy and validation cost, with the time-based deny-validity approach costing seven times that of timeless Petty and Weisel formalism.

1. Introduction

In component-based simulation model development, developed and validated simulation components are combined and re-combined to suit specific user requirements [10, 17]. In the development process, the user ideally specifies his requirements to an integrated component-based system

*A version of this paper is published in the Proceedings of the 25th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation, IEEE Computer Society Press, Nice, France, pp. 1-8, June 2011.

that builds the simulation in real time from a library of simulation models that can be easily combined to produce desired functionality [17]. *Syntactic verification* and *semantic validation* are two key steps in component-based simulation model development [25]. In syntactic composability, components have to be properly connected and must interoperate, which assumes common communication protocols, data formats, as well as a common understanding of the time management mechanisms employed. In semantic composability, the composition must be meaningful for all components involved, in terms of data exchange and context. Furthermore, the composition must be valid [25]. Model Verification, Validation and Accreditation (VV&A) is one of the most important steps in the simulation life-cycle. Classical, i.e. non component-based, validation methods include various stages and processes, and often result in a costly, time-consuming process [1, 12]. The validation cost is further aggravated in component-based modeling and simulation where the complexity of the models increases rapidly with size, and semantic composability validation remains a non-trivial problem [4, 10, 25, 32].

Despite advances in simulation validation, the validation of semantic composability remains a non-trivial problem because of several issues. Firstly, composition is not a closed operation with respect to validation since valid components do not necessary form valid compositions [1]. Secondly, reused components are developed for different purposes and when composed may result in emergent properties [15]. This implies that the overall behavior of the composed model cannot be obtained as a union of the individual behaviors of its constituents. Similarly, the context in which a reused component was developed and validated might differ from the new context of the composed model [5, 32], thus influencing the component interaction in unspecified ways. Thirdly, there exist various validation perspectives on the component interactions over time, e.g. *model behavior* aspects such as deadlock, safety, and liveness, *temporal* aspects, and *formal* measures of the validity of compositions, also called “figures of merit” [17]. Lastly, the composed model must be similar or close to the real system it abstracts [1]. Very often, the implementation of an automated process to evaluate this similarity is difficult, if not impossible [1, 10, 32].

In recent years, several automated approaches have been proposed for the verification and validation of composed simulation models, most of which target the validation of semantic composability [25, 34, 35, 36, 16]. In this paper, we have selected for evaluation a Z-based specification for validating Discrete Event System Specification (DEVS) models [34], an automated input/output transformation validation for CD++ and DEVS models [36], a formal theory of composability [25], and our proposed deny-validity approach [37, 38]. These approaches focus on various aspects of validity, with several underlying assumptions and limitations that influence their outcome and applicability. An important issue remains their underlying computational cost, especially since in the context of large-scale composed simulation models this cost increases exponentially [18, 37]. This is because simulation components are complex entities with a variety of attributes and complex behavior, and their composition often leads to state space explosion. In particular, the validation of the entire model space as proposed by model checking or theorem proving approaches

[34] becomes unfeasible once a certain number of components, connections, and/or interactions is achieved. Similarly, the cost of comparing the composed model with a reference model grows exponentially with the number of components and their attributes and states. Besides problem size, other factors can influence the computational cost of semantic composability. These factors can be classified in two main categories: (i) simulation problem characteristics (i.e. number of components, the composition structure, etc.), and (ii) validation approach characteristics (abstractions, validation steps etc.). We discuss these factors in detail in the following sections.

To distinguish between existing validation approaches that can be employed in the simulation life-cycle, a common metric to evaluate the cost of validation is needed [22]. In this paper, we propose to quantify the *computational cost* of semantic composability validation. We first analyze the validation cost of existing semantic validation approaches in terms of simulation problem characteristics such as the number of components and composition structure. We next analyze the scalability of existing approaches using composed models with a large number of components. Next, we evaluate and quantify the trade-off between validation accuracy and computational cost. Our main contribution is a comprehensive, quantitative analysis on the computational cost of semantic validation in terms of problem characteristics and validation accuracy.

This paper is organized as follows. Section 2 presents an analysis of the computational cost of semantic composability validation. Section 3 presents an evaluation of current validation approaches in terms of computational cost, number of components, and validation accuracy. We compare and contrast our work with existing evaluation studies in Section 4. Section 5 concludes this paper and discusses future work.

2. Cost of Semantic Composability Validation

The factors that influence the computational cost of semantic composability validation can be classified in two main categories: (i) simulation problem characteristics, and (ii) validation approach. From a component-based perspective, measurable *simulation problem characteristics* include the number of components, the description of the components, the composition structure, and the degree of interaction between components. *Validation approach characteristics* include the validation techniques and abstractions employed. The *number of components* in the composed model describes the size of a component-based simulation model. More fine-grained measures of the composition size are determined by the way in which components are described, in terms of the *number of attributes*, and the *number of states* per component. A particular importance, depending on the validation approach, has the number of time delay attributes. The *composition structure* also influences the computational cost of validation. If a component-port paradigm is adopted, such as in DEVS [40], CoDES [39], or OSA [9], the composition structure is best described using the *number of connectors*. A large number of connectors, in particular those with complex semantics such as fork/join, can lead to state-space explosion even for composed models with a small number of components, when model checking or theorem proving validation approaches are employed

[37]. This is also the case when *component interaction*, expressed in terms of the number of events per unit time, is frequent. Validation approach characteristics that influence computational cost include validation techniques and validation abstraction. The *validation techniques* employed can have a major influence on the computational cost of validation. Classic examples are model checking techniques, which are not feasible when large models, in terms of size and complexity, are validated [18]. The type of *abstraction* employed by the validation approach has the potential to reduce the computational cost. For example, if time is not considered in a *timeless* validation abstraction, the computational cost can be reduced, as we will show in Section 3 for the Petty and Weisel approach.

The validation of semantic composability includes a comparison between the composed model and the reference model, and an evaluation of the closeness between them. The outcome of this evaluation depends on the formalism used to abstract the two models to facilitate reasoning. In this paper we analyze the influence of a time-based and a timeless formalism on computational cost. Another factor that influences the comparison process is the validation window size, defined as the interval during which the comparison between the composed model and the reference model is performed. Intuitively, given a single validation window, a larger window size ensures that deviant behavior of the composed model from the reference model can be identified. However, we have found that an increase in size of the validation window beyond a certain knee value comes at very high computational cost. This is similar with the relationship between accuracy and cost suggested in [27], in which after a certain knee value, increases in accuracy come at disproportionately larger increases in cost.

3. Cost Analysis

In this section we propose to evaluate the computational cost of four validation approaches, namely, the CD++ DEVS-based validation approach [36], the DEVS-based Z specification validation approach [34], the Petty & Weisel formal validation approach [25], and our deny-validity approach [37]. Our selection was based strictly on the availability of the implemented code, or in the case whereby the code was not available, on the completeness of the descriptions in the published papers to facilitate faithful implementation. The Z specification language has been proposed to formally validate models represented in the DEVS formalism [34, 35, 40]. The atomic DEVS model is represented in Z in a time-less manner, and a theorem proving tool based on Z such as Z/EVES [26] is used to verify the model and prove certain properties. However, the Z specification language limits the applicability of this approach to coupled DEVS models. Wainer et al. proposed an input/output transformation validation tool that inputs certain data in the DEVS coupled model and expects specific data through an output point [36]. This lightweight approach treats a DEVS coupled model as a blackbox and is easy to use since it employs two well-known DEVS/CD++ constructs, i.e. *Generator* to generate input, and *Acceptor* to accept input. However, only a single output point can be tested and only primitive data types such as real and integer are considered. In the formal theory of semantic composability validation proposed by Petty and Weisel a composed simulation model is modeled as the composition of mathematical functions that represent components over one-dimensional integer domains [25]. The simulation of the composed model is represented as a Labelled Transition System (LTS), where nodes are model states, edges are function executions, and labels are model inputs. Using several metrics, the distance between simulations of the composed model and a perfect model is calculated. However, time is not modeled and the approach is not feasible for compositions with feedback loops and fork and join connectors.

In our previous work, we proposed a deny-validity approach in which the composed model was subjected to a battery of tests aimed at discarding it as semantically invalid[1] [37, 38]. Informally, we attempt first to eliminate models in which components cannot communicate and coordinate meaningfully. While properties such as communication and coordination fall into the general definition of simulation verification [1], they are included in the definition of semantic composability [32, 25] and as such they are validated in this layer. Next, models with invalid semantic composability are also those that have valid model properties, but whose execution is not close to that of the real system the composed model abstracts [1]. To eliminate models that are not similar to the real system being abstracted, we compare between the composed model and a reference model. In contrast to current static perfect model validation [25], our proposed *time-based formalism* represents dynamic component behavior, can represent fork and join connectors, and is applicable to a composed models with wide variety of topologies and interactions. Furthermore, we are able to quantify the similarity between the composed model execution and the reference model execution using our defined semantic metric relation.

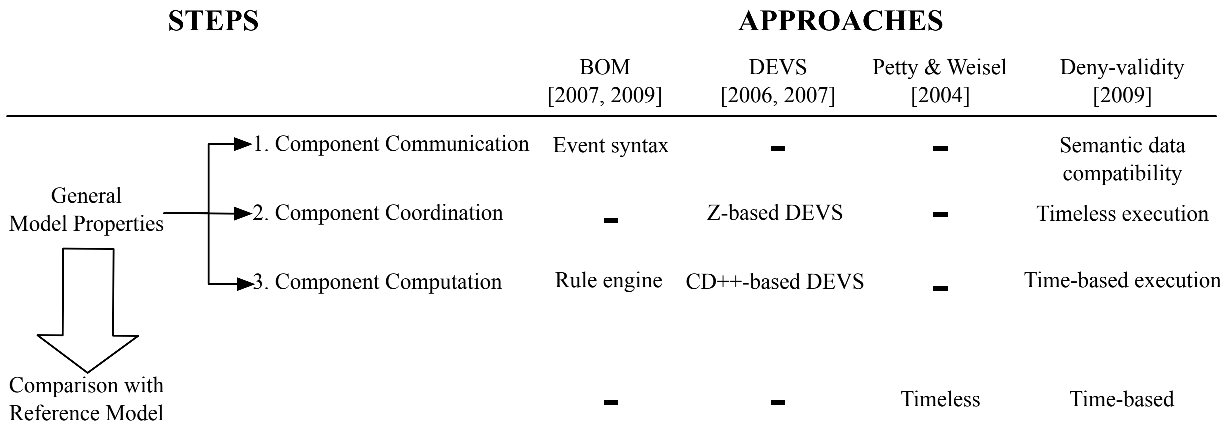
We divide our evaluation process in two stages. Firstly, we evaluate the cost of validating general model properties. Secondly, we evaluate the cost of comparing between the composed model and a reference model. The factors that influence the computational cost of validation form a complex parameter space in which the influence of each individual factor is difficult to identify. Thus for simplicity, we employ a single-server queue model with a varying number of components. A more complex example is analyzed in Section 3.4.

3.1. Evaluation Methodology

To better understand the landscape of semantic composability validation, we summarize existing validation approaches in Figure 1. An important observation is that most component-based verification and validation approaches focus on two main aspects, namely, on the validation of general model properties, and on the comparison of the composed and the reference models. General model properties are validated by the CD++ DEVS-based approach (input/output transformations), the DEVS-based Z specification approach (component coordination), and BOM-based approaches

[1]We ensured that models with invalid syntax were eliminated using an approach based on compositional grammars expressed in EBNF [39].

(component communication and input/output transformations). Comparison between the composed model and a reference model is performed by the Petty & Weisel approach. Lastly, our deny-validity approach targets both validation stages.



STEPS		APPROACHES			
		BOM [2007, 2009]	DEVS [2006, 2007]	Petty & Weisel [2004]	Deny-validity [2009]
General Model Properties	1. Component Communication	Event syntax	-	-	Semantic data compatibility
	2. Component Coordination	-	Z-based DEVS	-	Timeless execution
	3. Component Computation	Rule engine	CD++-based DEVS	-	Time-based execution
Comparison with Reference Model		-	-	Timeless	Time-based

Figure 1. Landscape of Recent Validation Approaches

For our study, we were able to obtain the code for the CD++ DEVS-based approach. We implemented the second DEVS-based approach following the complete descriptions given in [34]. However, the Z specification does not permit the representation of coupled or connected objects, and as such it cannot be used to specify component-based models. Instead, we employed the Object-Z formalism [13], which can be easily adapted in a similar manner as that suggested in [34], to cater for DEVS coupled models. However, model checking based on Object-Z is still in its early stages. Nonetheless, we were able to reach the outcome described in [34], i.e. syntax, type, and inconsistency checking by employing the Wizard checker for type and syntax checking on the composed model [29]. Ongoing work exists to include the Object-Z specification into theorem proving tools such as Isabelle/HOL [30], but these were beyond the scope of our study. The advantage and major improvement of our implementation is that it caters for component-based models. We implemented the Petty & Weisel approach following details from the published papers [23, 24, 25]. A fundamental assumption in the Petty & Weisel approach is that a simulation component can be transformed into a mathematical function over integer domains. Nonetheless, no details about how this transformation should be performed are provided. This transformation is crucial when computing the mathematical composability of the functions that represent components, which by definition translates to verifying integer domain inclusion. In our implementation, we employed a brute-force transformation by mapping every component output into an integer number (iteratively for each output) and assuming that the functions are mathematically composable. This is not a limitation in our study of computational cost because checking for mathematical composability is not a major component of the validation cost in the Petty & Weisel approach[2].

[2]In particular, provided that the transformation from the meta-model to integers is sound, the algorithm to determine mathematical composability could employ a Radix sort algorithm [19] to first sort the integer values, followed by an inclusion check, resulting in $O(kn)$ steps, where n is the number of elements in the interval, and k is the average element length.

Our study follows the two main validation steps described in Figure 1. Firstly, we evaluate the computational cost of approaches that aim to validate general model properties, namely, the CD++ DEVS-based validation approach [36], the DEVS-based Z specification validation approach [34], and our deny-validity approach [37, 38]. We employ a simple queue model with varying number of service unit components as shown in Figure 2. We have chosen this queue model because of

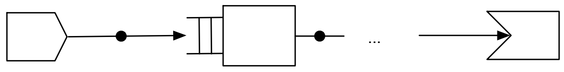


Figure 2. Simple Queueing Model

its relative simplicity with respect to factors such as the number of attributes per component and composition structure. Besides the need for simplicity, we have also excluded these factors because they are not considered by most validation approaches. Secondly, we study two approaches that compare the composed model with a reference model, namely, Petty & Weisel's validation approach [25] and our deny-validity approach. We evaluate the variation of cost with model size and analyze these approaches in terms of accuracy and cost. We first show the impact of timeless abstractions, such as that employed by Petty & Weisel, on the validation process. Next, we evaluate the variation of cost with the validation accuracy, expressed using the size of the validation window during which the composed model is compared with the reference model. We present an average of ten execution runs and execute all experiments on an Dell PowerEdge SC1430 Dual Quad Core Server, with Intel Xeon, 1.83 GHz, and 4GB RAM.

3.2. General Model Properties

The validation of general model properties focuses on three main properties, namely, component communication (P1), component coordination (P2), and component computation (P3), in the composed model. The validation of component communication aims to check data compatibility between connected components in the composition. Current approaches ensure that components follow a common reference model [33], verify syntax in terms of event parameters and data types [21], or validate semantic compatibility using a component-based ontology [39]. The validation of component coordination aims to check that interleaved executions of components in the model are correct [11]. This is usually done using model checkers, and in general by abstracting time in instantaneous transitions [18, 37]. Lastly, the validation of component computation aims to check that the components can execute during the composition run. For simplicity, the Z-specification DEVS-based and the CD++ approach will be hereafter referred to as DEVS1 and DEVS2 respectively.

Table 1 presents the variation of computational cost when varying the number of components from 10 to 1,000. While the computational cost increases proportionally with the composed model size, the increase is insignificant in the DEVS1 and DEVS2 approaches, in which the computational cost approaches one second and 17 seconds for 1,000 components respectively. In contrast, the deny-validity approach takes close to 7 minutes for the same model. The reason for this discrepancy becomes evident when we look at the cost components of the deny-validity approach,

# Components	Runtime (s)					
	DEVS1	DEVS2	Deny-validity			
	P2	P3	P1	P2	P3	Total
10	< 0.1	0.2	< 0.1	5.3	51.1	56.5
100	< 0.1	0.5	0.2	146.6	43.5	190.3
500	0.2	4.5	0.3	193.6	67.0	260.9
1,000	0.7	16.7	0.5	330.4	130.9	461.7

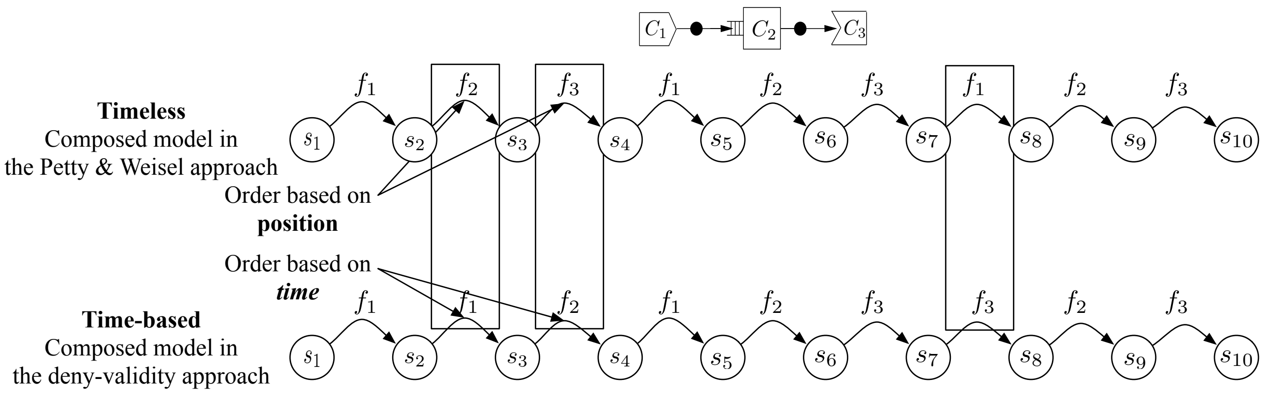
Table 1. Computational Cost of Validating General Model Properties

which is the only approach that validates all general model properties. The cost of validating property P1 is insignificant for this model, but increases with the number of data types that a component outputs. This is because P1 is validated by establishing data compatibility between components using our proposed ontology, which is queried for every pair of connected components [39]. Space constraints prevent us from showing these results here. The validation of property P2 incurs the largest cost of the three properties. The validation is done using the SPIN [6] model checker to validate a specification of the composed model, and becomes highly unfeasible when the composed model size increases beyond 250 components. As such, flags that limit the search space are employed for validation of models with 500 and 1,000 components. In particular, we use the `` `-w' `` flag to reduce the depth of the search tree, and `` `DMEMLIM' `` to cap the amount of memory used. Property P2, component coordination, is also validated by the DEVS1 approach. However, this approach focuses only on type and syntax checking, and does not look at all possible interleaved execution states, like the deny-validity approach.

The validation of property P3 is performed in a similar manner for DEVS2 and the deny-validity approach, by checking several types of data at a connection point in the composition. However, the DEVS2 approach can only process a limited number of events and requires the user to input the exact moment in time when the output is expected. Moreover, the DEVS2 approach validates input at the last connection point in the composition. Lastly, only primitive data types such as real or integer can be validated. In contrast, in the deny-validity approach the user can specify desired data of any type, including domain specific, at any connection point in the composition. Additional liveness properties are also validated [37].

3.3. Comparison with Reference Model

Comparing the simulation model with a real or referent system is a traditional validation approach [1]. This comparison is done in component-based simulation by the Petty & Weisel [24] and the deny-validity approaches [37]. These two approaches follow a similar validation sequence, which can be separated into three major steps: (i) transformation of components into formalism; (ii) transformation of composition formalism into a Labeled Transition System (LTS) [31]; and (iii) comparison with reference model. Both approaches rely on a formal comparison between the simulation of the composed model and that of a reference model. The major difference between the two approaches lies in that Petty & Weisel employ a static formalism, in which a component is represented as a function over integer domains, whereas the deny-validity approach proposes a time-based formalism in which a component is represented as a function over a three-coordinate domain containing time, state, and input/output. The Petty & Weisel approach offers a high level of abstraction, which permits reasoning about closeness under composition and has reduced computational cost as we will show below. However, it is difficult to transform component representations into integer values automatically, and the approach assumes that model properties, such as syntactic composability, safety, liveness, are validated beforehand. Moreover, the comparison process orders the component functions based on the location (left to right) of the components in the composition, which does not permit the validation of compositions with fork and join connectors and feedback loops. This is because the functions representing components on the fork/join branches need to be ordered during execution, which is not possible using only integer value domains. The same applies to feedback loops, where a mathematical composability ordering based on the position of the components cannot be deduced. Figure 3 highlights the difference between timeless and time-based ordering. As it can be seen, the LTS that represents the composed model has an



C_1
C_2
C_3
Timeless
Composed model in the Petty & Weisel approach
Order based on **position**
Order based on ***time***
Time-based
Composed model in the deny-validity approach
f_1 f_2 f_3 f_1 f_2 f_3 f_1 f_2 f_3
f_1 f_1 f_2 f_1 f_2 f_3 f_3 f_2 f_3
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10}

Figure 3. Execution Order: Timeless vs Time-based Validation

equal number of states in both approaches, but a different sequence of function execution. The deny-validity approach orders the functions based on the time moment when the component interacts with its neighbors, resulting in the sequence $f_1, f_1, f_2, \ldots$. In contrast, the Petty & Weisel approach orders the functions based on the position of the components in the composition, resulting in the sequence $f_1, f_2, f_3 \ldots$. Although there are instances where a position ordering is useful, e.g. when reasoning about validation closeness under composition, the latter time-based ordering paints a more realistic picture of the execution of the composed model. However, the time-based formalism employed in our deny-validity approach incurs an additional validation overhead as we discuss below.

The computational cost of validation can also be divided into three components, namely, (i) f, the *formalism cost*, as the cost of transforming from the component representation to the chosen formalism; (ii) p, the *process cost*, as the cost of transforming the composed model into a LTS using the chosen formalism; and (iii) c, the *comparison cost*, as the cost of comparing between the two LTS, representing the composed model and the reference model respectively. These cost components can be further refined as functions of various component and composition characteristics, as shown in Table 2. An important point to highlight is that the cost of obtaining or constructing the reference model is not included. This is because the reference model is assumed to exist a-priori in the Petty & Weisel approach, whereas in our approach the reference model is automatically derived based on generic descriptions of reference components [37, 38].

Cost Component	**Petty & Weisel**	**Deny-validity**
Formalism - f	at least $f(n,\tau)$	$f(n,s,t,\tau)$
Process - p	$p(n,\tau)$	$p(n,s,t,\tau)$
Comparison - c	$c(n,\tau)$	$c(n,\tau)+c_{eps}(n,a,\tau)$

Table 2. Cost Components

Our results presented below show that the number of components (n) drastically influences the computational cost. This is because the fundamental unit of each validation approach is the mathematical function, which represents a component. Other parameters, such as the average number of states per component (s) and the average number of attributes per component (a), influence the cost of the deny-validity approach but not the cost of the Petty & Weisel approach. This is because the Petty & Weisel approach deals only with integer representations. Nevertheless, the influence of the number of states and attributes on the computational cost in the Petty & Weisel approach should be more evident in the formalism cost, f, because the transformation from the component representation to unique and meaningful integer values should consider states and attributes as well. However, Petty & Weisel do not discuss details about how this transformation is performed. Another parameter that influences the computational cost is τ, the size of the validation window during which the composed model is compared to the reference model. This translates into the number of simulation steps in the Petty & Weisel approach, and into the unfolding degree in our deny-validity approach. For example, for the simple model in Figure 3, $\tau = 3$, resulting in three simulation steps and ten states for the LTS representing the composed model. The parameter τ can be seen as a measure of the accuracy of the validation process if we agree that the longer the interval under which the composition is observed as compared with the reference model, the more accurate is the validation result. We next evaluate the computational cost of semantic validation with the composition size in terms of the number of components, n and analyze the trade-off between the computational cost and the validation window size τ.

3.3.1. Results and Analysis

We implemented the Petty & Weisel approach based on its description [23, 24, 25]. Since the details of mapping each component into a function over integer domains are not documented in Petty & Weisel approach, we used a simple and fast heuristic as discussed in Section 3. Next, the composed model LTS was created as shown in Figure 3. Lastly, we implemented the comparison between the composed model LTS and a reference model LTS using the BISIMULATOR tool

[14], which is the same one we employ in comparing between the LTS of the composed model and reference model in the deny-validity approach.

Table 3 presents the variation of computational cost when varying the number of components from 10 to 1,000, with a validation window of size $\tau = 3$.

# Components	Runtime (s)	
	Petty & Weisel	Deny-validity
10	2.15	5.46
100	3.68	14.27
500	9.36	35.87
1,000	14.63	76.10

Table 3. Comparison with Reference Model

As it can be seen in Figure 4, the computational cost of the Petty & Weisel approach is reduced, e.g. to less than 15 seconds for a simple queueing model with 1,000 components. In contrast, our deny-validity approach has a runtime of around one minute for the same model. This is because in our approach, a larger number of components implies that a larger number of component executions have to be ordered towards achieving the time-based ordering presented in Figure 3.

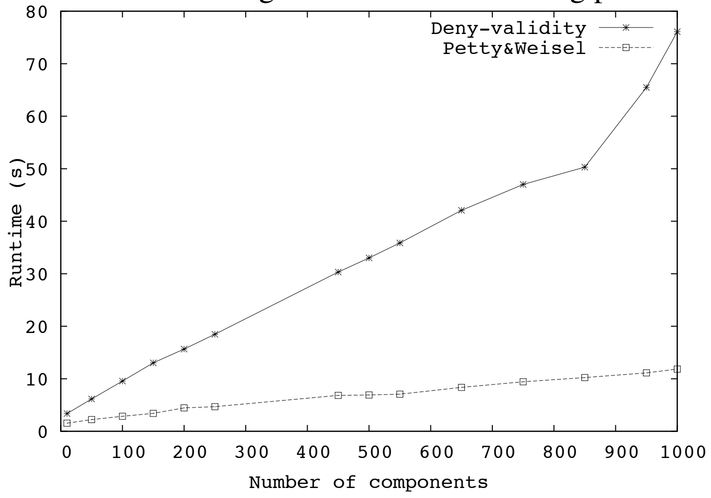


Figure 4. Varying the Number of Components

To evaluate the trade-offs between computational cost and accuracy of the validation process, we evaluate the runtime cost of validating a queueing model with 1,000 components while varying the values of the validation window size τ to 3, 10, 15, and 20. Our results are presented in Figure 5.

The increase of the computational cost with the validation window size τ is insignificant for the Petty & Weisel approach. As it can be seen in Figure 5, for $\tau = 25$, the Petty & Weisel approach takes on average 40 seconds for a M/M/1 model with 1,000 components. In contrast, there is an evident trade-off in the deny-validity approach. Specifically, the validation runtime increases from around 3 minutes for $\tau = 20$, to around 20 minutes for $\tau = 25$. The explanation for this decrease in performance is the following. Our validation process includes a time-ordering module which

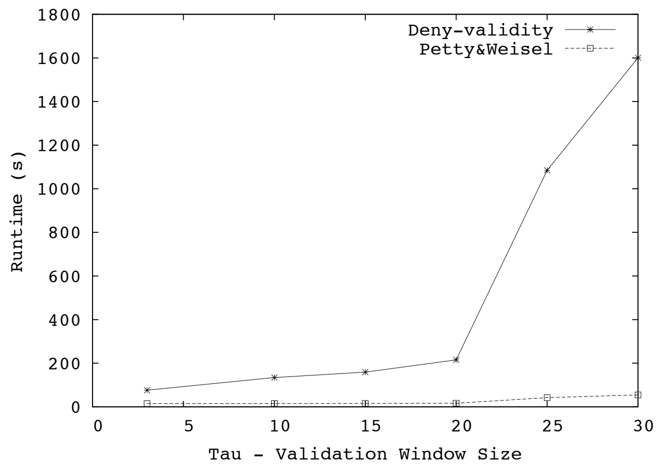


Figure 5. Cost vs Accuracy

orders all functions based on the time moment in which components interact with neighbors. This implies that a correct time ordering of components is necessary. The computation of this ordering is a constraint satisfaction problem, requiring a constraint solver to solve several equations. For $\tau = 25$, the LTS of the composed model has around 25,000 states. This translates into 25,000 constraints over the entire positive integer domain (since time values are integers) that have to be solved in order to determine the time ordering of the function executions, as required by our validation process. This operation has to be performed twice, for the composed and the reference model respectively. In solving these constraints, we employ the Choco constraint solver [7], which for our constraint types has a theoretical complexity of $O(c^3)$, where c is the number of constraints. This problem does not appear in the Petty & Weisel approach, because time is not modeled and as such a time-based ordering is not necessary. For the curve in Figure 5, we have calculated the knee values beyond which increases in validation window size come at disproportionate increases in validation cost. For the deny-validity approach, for models with 500 components, we have calculated the knee value as $\tau_{knee} = 15.22 \approx 15$. For models with 1,000 components, $\tau_{knee} = 18.99 \approx 19$.

3.4. Further Insight

In the above, we limited our evaluation study to only two factors, namely, the number of components n and the validation window size τ. Another important factor that influences the cost of validation is the composition structure, described by the number of one-to-one, fork, and join connectors. Existing validation approaches cannot validate models with fork and join connectors. However, this is possible in our deny-validity approach. We analyze the cost of validating composed models that contain fork and join connectors as shown in Figure 6.

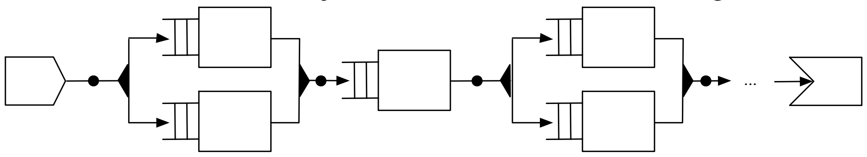
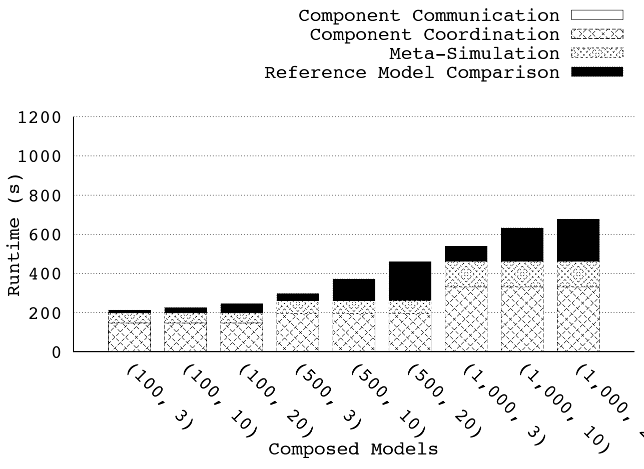
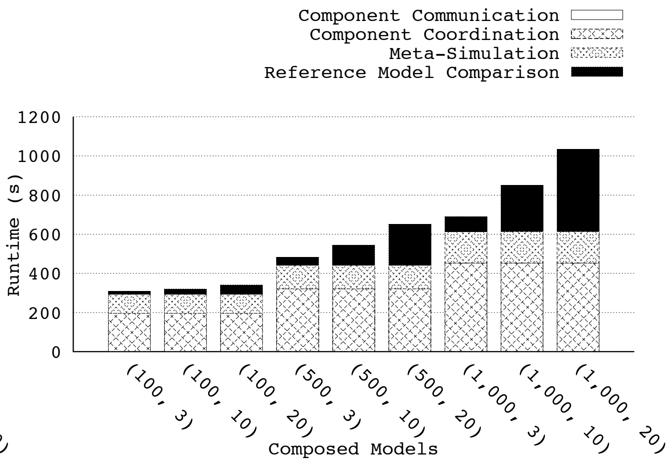


Figure 6. Queueing Model with Fork/Join Connectors

Figure 7(a) presents a breakdown of the validation cost for composed models that contain only one-to-one connectors. Figure 7(b) presents a breakdown of the validation cost for composed models that contain a ratio $r = 10\%$ of fork and join connectors, e.g. for 1,000 components there are 10 fork and 10 join connectors.



(a) Simple Composition Structure (n, τ)



(b) Complex Composition Structure (n, τ)

Figure 7. Total Validation Cost

As expected, for a composed model with $n = 1,000$ and $\tau = 20$, the total validation time increases from 10 minutes for the simple composition in Figure 7(a) to around 17 minutes for the composed model with a more complex structure in Figure 7(b). Moreover, the percentage of validation cost also changes. In the composed model without fork and join connectors, the percentage of validation cost was distributed as $50 : 19 : 31\%$ for component coordination, component computation, and validation by comparison with reference model respectively. In contrast, the presence of fork and join connectors leads to a $40 : 15 : 45\%$ cost distribution, suggesting that as the number and complexity of connectors increase, the penalty incurred by the comparison with a reference model also increases. However, the cost of verifying component coordination becomes unfeasible as suggested before [18] and is capped as discussed in Section 3.2. An important point to highlight is that the total execution cost for a SSF implementation [8] of the simple composed model with $n = 1,000$ is on average 49.57 seconds, and the cost of validating it is on average 11.28 minutes. However, because of the layered nature of the validation approach, the cost incrementally increases and the user can stop at any validation layer as desired. The validation of component communication and coordination incurs an initial overhead of 5.5 minutes. The meta-simulation layer incurs and additional 2.14 minutes overhead. Finally, the comparison with a reference model incurs a final cost of 3.64 minutes. However, space constraints prevent us from showing the analysis here.

4. Related Work

Verification, Validation, and Accreditation (VV&A) has been a principal focus of research in the simulation community since the late 70s [28], with work such as that by Balci [1, 2, 3] and Sargent [2, 3, 27, 28] among others, providing detailed guidelines and process organization. However, very few works quantify the cost of validation, more so in the new area of component-based modeling and simulation. This is mainly because validation cost is highly dependant on many factors, such as characteristics of the simulation problem, model and application domain among others, and thus is difficult to estimate. Balci and Sargent propose a methodology for cost-risk analysis in the statistical validation of simulation models, by looking at the relationship between data collection cost, acceptable validity range, and model builder and model user risks, when performing statistical hypothesis testing [2]. A validity measure is proposed and the trade-offs between data collection budget and model user's risk are analyzed. However, there is no estimate of the actual cost of validation. Historical data from the US Defense Modeling and Simulation Office suggests that VV&A activities account for 5 to 17.5 % of the total modeling and simulation budget [11]. A clear estimate of the validation cost and the key factors that influence it is also missing in the simulation industry. Reports from industry suggest that validation cost is between 5 and 19 % of the total project cost [20] but no cost model is provided. The same applies to the validation of component-based simulations. While there are many works that focus on verifying and validating general model properties [21, 34, 35, 36] or on comparing the composed model with a reference model [25, 37], existing work lacks an evaluation of the applicability and cost of proposed approaches. In contrast, in this paper we evaluate four existing approaches. We present a classification of the factors that influence computational cost and analyze the cost of existing validation approaches in terms of the number of components and composition structure.

5. Conclusion

Industry practice suggests that the cost of traditional simulation validation ranges from 5 - 19% of the entire project cost [20]. In component-based modeling and simulation, semantic validation cost can be significantly higher. To the best of our knowledge, we present the first quantitative study of the cost of validating semantic composability and its trade-offs with validation accuracy. This study compares the cost of four main validation approaches, namely CD++ DEVS [36], the Z specification based DEVS [34], the Petty & Weisel formal theory [25], and deny-validity [37, 38]. A simple queueing model with one thousand components is used and validation measurement is performed using a Dell PowerEdge SC1430 compute server. The key factors that influence the computational cost are *simulation problem characteristics*, including the composition size and the degrees of interaction, and *validation approach characteristics*, including the techniques used and the levels of abstraction. Current validation approaches focus on validating general model properties, and on comparing between the composed model and a reference model. In general, the cost of validating model properties grows, as expected, with the number of components. However, the actual cost is vastly different among approaches. The computational cost ranges from less than one second in CD++ DEVS to more than 7 *minutes* in deny-validity. The cost depends mainly on the underlying abstractions and assumptions employed.

The cost of comparison of the composed model with a reference model in the deny-validity approach is 7 times that of the Petty & Weisel approach. This reflects the time-based formalism and ordering in deny-validity, which provides increased accuracy at the expense of runtime. Moreover, our study showed that the time-based ordering in the deny-validity approach causes the cost to explode when the size of the validation window increases. In the Petty & Weisel approach, the increase in validation window size results in an insignificant increase in cost. In contrast, a time-based ordering, which facilitates the validation of composed models with complex structures such as feedback loops and fork and join connectors, causes a significant increase in the computational cost of validation. Specifically, a 25% increase in validation window size results in a five-fold increase in validation cost.

As the deny-validity approach is one of the most complete process for semantic validation, allowing different degrees of validation accuracy and cost, we analyze it further. We first analyzed the cost of the different validation layers in the deny-validity approach using a simple queueing model. Next, we analyzed the influence of the composition structure, in terms of the number of fork and join connectors, on the validation cost. Our study shows that while it takes on average 49 seconds to execute the SSF implementation of a composed simulation model with 1,000 components, it takes on average 17 minutes to validate it. The cost of validating general model properties represents 55% and the cost of validating model execution accounts for the remaining 45%. A 10% increase in the number of fork and join connectors in the composition comes with a 50% increase in the validation cost.

References

[1] O. Balci. Verification, Validation and Accreditation of Simulation Models. In *Proceedings of the Winter Simulation Conference*, pages 135–141, Atlanta, USA, 1997.

[2] O. Balci and R. G. Sargent. A Methodology for Cost-Risk Analysis in the Statistical Validation of Simulation Models. *Communications of the ACM*, 24:190–197, 1981.

[3] O. Balci and R. G. Sargent. Some Examples of Simulation Model Validation Using Hypothesis Testing. In *Proceedings of the Winter Simulation Conference*, pages 621–629, San Diego, USA, 1982.

[4] J. Banks, J. Carson, B. Nelson, and D. Nicol. *Discrete-Event System Simulation*. Prentice Hall, USA, 2005.

[5] R. Bartholet, D. Brogan, P. Reynolds, and J. Carnahan. In Search of the Philosopher's Stone: Simulation Composability Versus Component-Based Software Design. In *Proceedings of the Fall Simulation Interoperability Workshop*, Orlando, USA, 2004.

[6] M. Ben-Ari. *Principles of the Spin Model Checker*. Springer Verlag, 2008.

[7] Choco Constraint Programming System. http://sourceforge.net/projects/choco/, (retrieved Jan. 2010).

[8] J. Cowie. Towards Realistic Million-Node Internet Simulations. In *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 2129–2135, Las Vegas,USA, 1999.

[9] O. Dalle. OSA: An Open Component-based Architecture for Discrete-event Simulation. In *Proceedings of the* 20^{th} *European Conference on Modeling and Simulation*, Prague, Czech Republic, 2006.

[10] P. Davis and R. Anderson. Improving the Composability of Department of Defense Models and Simulations, 2003.

[11] Defense Modeling and Simulation Office (DMSO). *Verification, Validation, and Accreditation Recommended Practices Guide*. U.S. Department of Defense. Office of the Director of Defense Research and Engineering, 1996.

[12] Department of the Navy. *Modeling and Simulation Verification, Validation, and Accreditation Implementation Handbook*. 2004.

[13] R. Duke, P. King, and G. Rose. The Object-Z Specification Language: Version 1, 1991.

[14] H. Garavel, F. Lang, R. Mateescu, and W. Serwe. CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes. In *Proceedings of the* 19^{th} *International Conference on Computer Aided Verification*, pages 158–163, Berlin, Germany, 2007.

[15] R. Gore and P. Reynolds. Applying Causal Inference to Understand Emergent Behavior. In *Proceedings of the Winter Simulation Conference*, pages 712–721, Miami, USA, 2008.

[16] P. Gustavson and L. Root. Object Model Use Cases: A Mechanism for Capturing Requirements and Supporting BOM Reuse. In *Spring Simulation Interoperability Workshop*, Orlando, USA, 1999.

[17] S. Kasputis and H. Ng. Composable Simulations. In *Proceedings of the Winter Simulation Conference*, pages 1577–1584, Orlando, USA, 2000.

[18] K. E. Kennedy. Formal Methods in the Verification and Validation of Simulation Models. In *Proceedings of the Spring Simulation Interoperability Workshop*, 2003.

[19] D. E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 1997.

[20] G. Love and G. Back. Model Verification and Validation for Rapidly Developed Simulation Models: Balancing Cost and Theory. In *Proceedings of the* 18th *International Conference of the System Dynamics Society*, Belgium, 2000.

[21] F. Moradi, R. Ayani, S. Mokarizadeh, G. H. A. Shahmirzadi, and G. Tan. A Rule-based Approach to Syntactic and Semantic Composition of BOMs. In *Proceedings of the* 11th *IEEE Symposium on Distributed Simulation and Real-Time Applications*, pages 145–155, Crete, Greece, 2007.

[22] D. K. Pace. Modeling and Simulation Verification and Validation Challenges. *Johns Hopkins APL Technical Digest*, 25:163–172, 2004.

[23] M. Petty and E. Weisel. Basis for a Theory of Semantic Composability. In *Proceedings of the Spring Simulation Interoperability Workshop*, Orlando, USA, 2003.

[24] M. Petty, E. Weisel, and R. Mielke. Composability Theory Overview and Update. In *Proceedings of the Spring Simulation Interoperability Workshop*, San Diego, USA, 2005.

[25] M. Petty and E. W. Weisel. A Composability Lexicon. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 181–187, Orlando, USA, 2003.

[26] M. Saaltink. The Z/EVES System. *Lecture Notes in Computer Science*, 1212:72–85, 1997.

[27] R. Sargent. Verification, Validation, and Accreditation of Simulation Models. In *Proceedings of the Winter Simulation Conference*, pages 50–59, Orlando, USA, 2000.

[28] R. G. Sargent. Validation of Simulation Models. In *Proceedings of the Winter Simulation Conference*, pages 497–503, Orlando, USA, 1979.

[29] G. Smith. *The Object-Z Specification Language*. Kluwer Academic Publishers, 2000.

[30] G. Smith, F. Kammller, and T. Santen. Encoding Object-Z in Isabelle/HOL. In *Proceedings of the International Conference of B and Z Users*, pages 82–89, Grenoble, France, 2002.

[31] J. Srba. On the Power of Labels in Transition Systems. In *Proceedings of the* 12th *International Conference on Concurrency Theory*, pages 277–291, Aalborg, Denmark, 2001.

[32] A. Tolk. What Comes After the Semantic Web - PADS Implications for the Dynamic Web. In *Proceedings of the* 20th *Workshop on Principles of Advanced and Distributed Simulation*, pages 55–62, Singapore, 2006.

[33] A. Tolk, S. Y. Diallo, and C. D. Turnitsa. Mathematical Models Towards Self-organizing Formal Federation Languages Based on Conceptual Models of Information Exchange Capabilities. In *Winter Simulation Conference*, pages 966–974, Miami, USA, 2008.

[34] M. K. Traore. Analyzing Static and Temporal Properties of Simulation Models. In *Proceedings of the Winter Simulation Conference*, pages 897–904, Monterey, USA, 2006.

[35] M. W. Trojet, C. Frydman, and M. E.-A. Hamri. Practical Application of Lightweight Z in DEVS Framework. In *Proceedings of the Spring Simulation Multiconference*, San Diego, USA, 2009.

[36] G. Wainer, L. Morihama, and V. Passuello. Automatic Verification of DEVS Models. In *Proceedings of the SISO Spring Interoperability Workshop*, Orlando, USA, 2002.

[37] xxx omitted for double blind review.

[38] xxx omitted for double blind review.

[39] xxx omitted for double blind review.

[40] B. Zeigler, H. Praehofer, and T. Kim. *Theory of Modeling and Simulation*. Academic Press, 2000.