Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

QUANTIFYING VALIDATION OF DISCRETE EVENT SIMULATION MODELS

Mohammad Raunak
Megan Olsen

Department of Computer Science
Loyola University Maryland
Baltimore, MD 21210, USA

ABSTRACT

Simulation model validation and its rigorous assessment is known to be a difficult task. Quantification of validation is necessary to answer the question "how much validation is adequate?" One can answer this question by developing adequacy criteria to measure the validation performed on a simulation model. Developing test adequacy criteria for verifying computer programs has been useful for improving the quality and increasing confidence of regular software systems. We argue that from the validation and verification (V&V) perspective, simulation models are no different than software that are generally termed as "non-testable" due to the absence of a test oracle. There has been little research to develop V&V related adequacy criteria for this type of software. We present a validation coverage criterion, show in detail how it applies to discrete-event simulation models (DES), and discuss how it can be extended to develop V&V coverage criteria for other "non-testable" software.

1 Introduction

Testing and test adequacy criteria cover an important and large part of software verification and validation research. Test adequacy investigates approaches for determining how much testing is adequate for a software system. Over the last few decades, researchers have developed many different adequacy criteria to be used as metrics (often known as coverage criteria) to measure the level of verification performed on software (Zhu et al. 2009). Such criteria are used as guidance for selecting test cases, as well as a stopping rule for verification. The selection of test data based on a criterion routinely assumes the presence of a test oracle (Weyuker 1982), which is the mechanism through which one can definitively conclude whether a specific execution of software has met its intended and specified behavior.

A well known challenge is that not all software systems have test oracles (Bertolino 2007). Weyuker (1982) termed these programs as "non-testable". A large group of such software is written to determine an unknown answer. Machine learning algorithms, algorithms with non-deterministic outcomes, and simulations are example categories of software for which the absence of a test-oracle is a well established problem. We have yet to see research toward developing adequacy criterion for such software systems where an oracle is absent or impractical. Manual reviews (Zhu et al. 2009), n-version programming (Chen and Avizienis 1978), the use of pseudo-oracles in the form of metamorphic testing (Chen et al. 2003, Raunak et al. 2011), using runtime assertion checks (Rosenblum 1995), and development and use of incomplete oracles (Kanewala and Bieman 2013) are examples of some of the approaches used to test programs without oracles. However, we are unaware of attempts to identify methods to establish or communicate the level of verification or validation performed for "non-testable" suites of software.

A large suite of software systems where there is a clear absence of oracles is the set of simulation models that are executed for analyzing and predicting complex and dynamic systems. With the proliferation of complex algorithmic applications as well as the pervasive use of simulation in recent years, the importance

978-1-4799-7486-3/14/$31.00 ©2014 IEEE

of V&V related adequacy criterion for such “non-testable” software has increased. To discuss this issue, a clearer specification of the terms *verification* and *validation* and their relationship to testing is required. Software Engineering literature and textbooks have not been completely consistent in clearly separating these two terms. Ghezzi, Jazayeri, and Mandrioli (2002) explicitly decided to use only the term *verification* while discussing software testing and included testing, mathematical proofs, and information reasoning under this activity. Pfleeger (2009) defined *verification* to be the set of software engineering activities aimed at ensuring that different software artifacts conform to each other (e.g., the code conforms to the design, which conforms to the requirements specification), while *validation* refers to the set of activities aimed toward ensuring that the right system has been developed to solve the problem. If the software represents or automates a real world scenario, validation ensures that it does so accurately. With this definition, validation has some of the same challenges as testing a program without an oracle.

The definitions of verification and validation also vary in the field of Modeling and Simulation (M&S). In their classic simulation text, Law and Kelton (1991) state “*verification* is determining that a simulation computer program performs as intended, i.e., debugging the computer program ... *validation* is concerned with determining whether the conceptual simulation model (as opposed to the computer program) is an accurate representation of the system under study.” Balci defines verification as activities related to transformational accuracy of the simulation model with respect to the executable simulation program, while validation is the representational or behavioral accuracy of the simulation model with respect to the real world (Balci 2004). Sargent defines V&V in terms of three separate sets of activities performed in sequence: conceptual model validation, computerized model verification, and operational validation of the executable simulation model. The first step, conceptual model validation, refers to the determination that the theories and assumptions underlying the conceptual model of the real world system are consistent and reasonably match with the theories and assumptions of the real world system. The second step, computerized model verification, primarily involves the utilization of sound software engineering practices with a focus on proper ‘development’ and ‘testing’ so that the developed computerized simulation model accurately represents the conceptual model. This is identical to what Balci refers to as transformational accuracy. The third step of Sargent’s model is operational validation, which assesses how accurately the executable simulation model represents the real world and whether that is sufficient for the purpose of using the simulation. In this paper, we focus on this operational validity when discussing simulation model validation, as it is the primary validation task in M&S and is the closest to the meaning used in the larger software engineering community (Pfleeger 2009).

A primary reason we lack adequacy criteria for software without oracles is the inherent complexity associated with the problem. Balci states that we develop confidence about the accuracy and usefulness of the simulation model through continuous application of V&V throughout the simulation development life-cycle. However, Balci concludes that it is very difficult to numerically define the confidence level and therefore such nominal terms as “highly,” “exceptionally,” and “notably” should be used instead of a numeric value such as 95% in describing the amount of confidence one can gain from the application of V&V on a particular simulation model (Balci 2004).

There have been a few attempts on assessing and quantifying the V&V effort applied on simulation models. Wang proposed a method of assessing the V&V activities based on detection, classification and documentation of model defects during the simulation model construction period (Wang 2011). The focus of Wang’s work is properly classifying the defects, akin to IBM’s orthogonal defect classification (ODC) (Chillarege et al. 1993). Later Wang proposed a framework for developing a catalog of V&V techniques and selecting specific V&V technique for an M&S project based on their applicability (Wang 2013).

This paper challenges the status quo by presenting an approach for developing a numerically quantifiable adequacy criterion for validating simulation models, building on the authors’ prior work (Olsen and Raunak 2013). We improve on our prior validation coverage criterion by updating the adequacy criterion calculation to account for weaknesses of the prior work, developing a set of aspects for discrete-event simulations, and proposing importance weightings for validation techniques. We apply this new criterion to a health care discrete-event simulation case study. We argue that the community should focus on developing other adequacy criteria for V&V of simulation models and other "non-testable" programs, and discuss specific research questions that should be investigated to fully develop these criteria. We posit that the establishment and use of such adequacy criteria for reporting V&V activities will bring significant changes to the practices in the community and improve the credibility and trustworthiness of simulation models.

The remaining sections of the paper are organized as follows: Section 2 describes the proposed procedure for computing validation coverage of discrete-event simulation models; Section 3 shows the step-by-step application of our framework on a DES model of a hospital emergency department; and Section 4 concludes with a summary and discussion of the work.

2 Methodology

Our validation coverage criterion has been developed to be applicable over a broad range of simulation modeling approaches. Our previous work showed the applicability of a simpler coverage criterion on agent-based models (ABMs), which have become more popular in the past decade. ABMs provide an object-oriented view of simulation that is natural for describing many systems. However, discrete-event simulation (DES) is still better known and more popular than ABM for evaluation of systems that can be described in terms of events occurring at discrete points in time. In both cases validation must be performed to answer the simulation specific question: "do my model and simulation correctly represent the system I am studying?"

The process for computing validation coverage is akin to evaluating more general non-deterministic software against its intended behavior. In testing a non-simulation software system, where the source code is available for examination, often the adequacy of test case execution is measured by how they exercise different units of code such as a line or branch of the program. Test cases are selected based on achieving a target adequacy of covering such program units. In our proposed validation coverage scenario, the unit to cover is defined to be a simulation model element. Our approach provides a framework for identifying such simulation elements. The parallel of test cases in the test adequacy criterion in our proposed framework is the choice and application of validation techniques. We do not, however, treat all elements and all validation techniques as being of equal importance.

Law (2009) articulates a seven step process for developing valid and credible simulation models. The fifth step asks if the programmed simulation model is valid. Sargent (2010) has termed this step of ensuring that the computerized simulation model behaves as expected as "operational validity." In Figure 1 we propose a process for performing operational validation for step five of Law's process. We provide a step-by-step approach that involves determining the conceptual elements of the model that must be validated, classifying them, validating them, and then determining how well the validation 'covers' the elements of the simulation model. To determine coverage we also need weights representing the relative importance of elements to be validated, as well as the relative usefulness of techniques used for validation. To be able to fully follow the process, we must answer five research questions (RQ):

- RQ1: How do we ensure that all elements to be validated have been determined?
- RQ2: How do we determine the techniques for validation for each element?
- RQ3: Can we ensure objectivity in user tracking of validation technique application and its success?
- RQ4: How do we determine the relative importance weighting of model elements?
- RQ5: How do we compute the coverage?

In this section we will discuss each of these research questions in order. We provide complete details for research questions RQ1 and RQ5, and discuss initial research approaches for RQ2, RQ3 and RQ4.

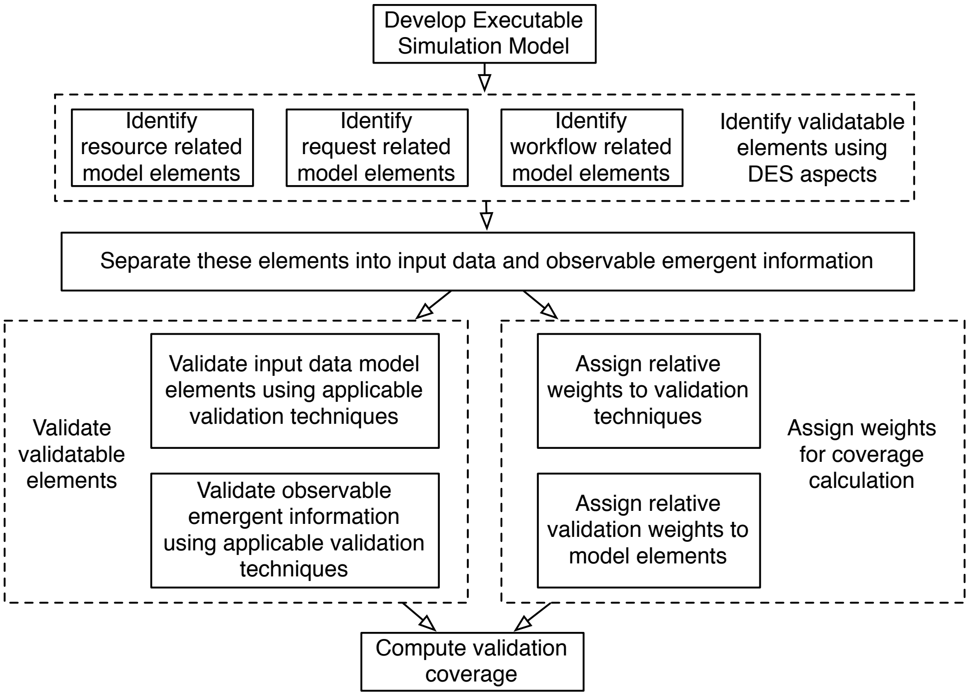
Develop Executable Simulation Model
Identify resource related model elements
Identify request related model elements
Identify workflow related model elements
Identify validatable elements using DES aspects
Separate these elements into input data and observable emergent information
Validate validatable elements
Validate input data model elements using applicable validation techniques
Validate observable emergent information using applicable validation techniques
Assign relative weights to validation techniques
Assign relative validation weights to model elements
Assign weights for coverage calculation
Compute validation coverage

Figure 1: The process of computing DES validation coverage, to define step 5 of Law's 7-step process. The details of each step will be described in sections 2.1 - 2.5.

2.1 Elements for Validation (RQ1)

We propose identifying all elements that need validation in a simulation model. To facilitate the identification of these validatable model elements, we organize them through high level groups called *aspects*. We provide a set of aspects for a particular simulation approach, with each aspect corresponding to a concept that is incorporated into at least some simulation models of that type. Each aspect is further defined by a set of *element types*. For a particular simulation, the elements to be validated can be determined by defining the particular elements in the simulation that correspond to each element type. These elements fall under two categories, input data (*id*) and observable emerging information (*oei*). The latter is defined as any observable and/or measurable behavior or data that a simulation model execution produces.

In our validation quantification framework, each element type may correspond to either emergent behavior in the system or input data. There can be multiple validatable elements for a particular element type. Identifying these aspects and element types is a crucial step of our proposed process. Given these guidelines, it should be relatively easy to determine all validatable elements of a simulation. We provide an example in section 3.

For an agent-based simulation the primary types of elements that required validation were related to agents, their environment, and their interaction with other agents or the environment. For example, “Rate of state change” is a common type of element in agent-based models that requires validation, and we proposed this element to be part of an aspect name “agent state” in an agent-based simulation (Olsen and Raunak 2013).

We present new aspects and element types for discrete-event simulations, which focus on events and their processing through three major aspects (Figure 2). These aspects are designed after the three primary groups of elements that define a discrete-event simulation model: resources, workflow, and requests for services:

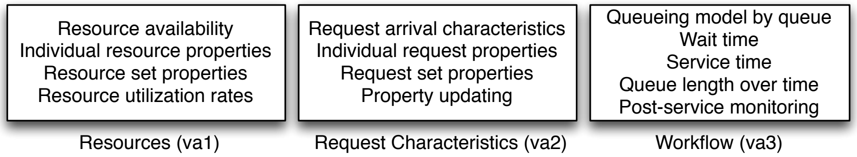

Figure 2: Proposed aspects and element types for discrete-event simulations.

1. *Resources*: Any DES simulation model needs to represent and use a number of resources. These resources may be modeled as service providers, in which case they can also be thought of as agents. Usually there are other resources these service providers (or servers) need and utilize to carry out their services. Examples of resources in a hospital emergency department simulation are nurses, doctors, registration clerks, or x-ray machines. Any validation related to the characteristics of these resources or set of resources, their availability, and their utilization will fall under this aspect.
2. *Workflow*: Most DES models capture some sort of workflow, be it a manufacturing process on a factory floor or a process of providing care in a hospital. This aspect accommodates the validation of the elements that characterize this workflow. It includes element types such as the different service times, wait times, and queuing characteristics associated with the nodes of the workflow.
3. *Request Characteristics*: There are usually entities that pass through the modeled workflow and utilize resources. For example, consider the items being manufactured or assembled in a factory floor or the role of patients in the process of getting care in a hospital. While going though the processes, these entities make requests for both agent and non-agent resources. These entities can be thought of as abstract representations of requests generated for services from different providers or nodes in a simulation model. The request characteristics aspect also captures the arrival of these entities at the beginning of the process or in any intermediate step of the process. This aspect represents validation related to the properties of these requests or set of requests.

2.2 Validation Technique Applicability (RQ2)

The equivalent of defining appropriate test cases is determining which validation techniques are applicable for validating a particular validation element. Each element of the simulation can be a candidate for application of some set of validation techniques. The existing literature enumerates the most commonly used validation techniques for simulation models, including face validation using subject matter experts, input data validation through goodness-of-fit tests, sensitivity analysis, extreme condition tests, graphical animation, and comparison with other models (Sargent 2010, Law 2009, Balci 2004).

To calculate the validation coverage criterion by our proposed approach, one needs to determine which of the available validation techniques are applicable. Ideally there would be a matching between validation element types and the most likely validation techniques. Developing this mapping requires the incorporation of additional information regarding the modeling approach and details of the particular model. In the next section we show one concrete example of a mapping between classifications of model elements (*id* and *oei*) and applicable validation techniques for discrete-event simulation models. Further research into developing frameworks for mapping applicable validation techniques by element type or aspect is necessary.

2.3 Maximizing Objectivity in Validation Coverage (RQ3)

The process for determining validation coverage involves identifying all model elements that require validation, deciding on the applicability of different techniques in validating those elements, performing validation, and then computing a validation coverage score. However, to ensure objectivity in the application of this approach we need concrete guidelines about which validation techniques to use, and how to determine their relative importance.

Table 1: Validation Techniques with Relative Weights.

	Validation Technique	Abbreviation	Relative Weight
Input Data	Face validation through subject matter expert review	FV-id	2
	Goodness-of-fit tests	GoF	3
OEI	Animation	AN	1
	Sensitivity Analysis	SA	2
	Extreme Condition test	ET	2
	Face validation through subject matter expert review	FV-oei	2
	Comparison with other model output	CM	3
	Result validation	RV	4

We propose a mapping between categories of model elements and potential validation techniques, as well as relative weights to be used in computing the validation coverage (Table 1). These validation techniques are chosen from Law and Sargent's work (Sargent 2010, Law 2009), removing duplicates and techniques that do not appear to be generally in use. The relative weights are based on the level of confidence acquired from successfully performing the technique on a simulation model. To derive the proposed weights (Table 1) we draw from our experience with the relative usefulness of the validation techniques in increasing model confidence. These decisions are influenced by a survey on the use of validation techniques in Winter Simulation Conference papers (Raunak and Olsen 2014). The techniques researchers seem to use most frequently have been assigned higher relative weights than the more infrequent ones. Using these sets of weights for computing validation coverage can provide continuity among all simulation models using the validation coverage calculation, which will in turn decrease the subjectivity of the coverage calculation. These proposed weights are used in the case study in section 3.

2.4 Determining the Relative Weights (RQ4)

Our prior calculation only assigned weights to aspects, which are high-level descriptions of the simulation. This weight could bias against elements that are in a highly filled aspect, versus elements that are the only element within the aspect. The elements are, after all, the true delineation of what needs to be validated in the simulation. Therefore, once the elements for a particular model are determined, a weight must be assigned to each element for validation coverage calculation. Some elements will be more crucial for validation than others, and thus their validation should increase our confidence in the model by a higher degree than the validation of other less important elements. However, the determination of these weights is nontrivial in most cases.

The general steps are to first rank elements in order of validation importance, with some elements ranked identically; i.e., which elements increase our confidence in the simulation the most by being validated, and which increase our confidence equally? The next step is to determine by what extent higher ranked elements outpace lower ranked elements; is element (a) twice as important as element (b), three times as important, or only barely more important (in which case they should be of the same rank)? At this point numeric values can be assigned to each element. As the weighting is only relative within a particular simulation model, the modeler may choose the scale that best suits them for making this delineation. An example of this process can be seen in section 3. Note that this step leaves some room for the simulation modeler/validator to subjectively determine what elements are relatively more important and by how much. We believe the simulation modeler is in the best position to make these decisions. In every model there is usually some elements that are more crucial than others to ensure fidelity with the SUS and those elements should receive higher weights. For example, in the case of a hospital ED simulation, the average length of stay (LOS) of patients is an important overall emerging statistic of the ED simulation experiments. Consequently, validation of LOS is likely to have a higher relative weight.

2.5 Validation Coverage Computation (RQ5)

Once a computerized DES model has been developed, we propose that a simulation modeler should first identify all model elements that require validation. Each such DES model element will be an instantiation of a validatable element type as shown in Figure 2. When the modeler is satisfied that all such model elements for validation are identified, the process asks the simulation modeler to classify these validation elements into input data (*id*) and observable emergent information (*oei*s). Depending on which category a simulation model element falls into, the process provides suggestions for validation techniques that should be applied. Based on the results of applying the validation on the model elements, the final step of this process guides the simulation modeler to compute the coverage. To calculate validation coverage, we therefore utilize the following three sets of information:

- Elements that should be validated,
- Possible validation techniques applicable for validating each element, and
- Validation techniques that succeeded when utilized.

We proposed a validation coverage calculation for agent-based simulation in our earlier work (Olsen and Raunak 2013), but it has two limitations that we address in this paper. One limitation is that it does not address how applying a validation technique to validate multiple model elements should be computed. We argued that not all validatable model elements are of equal importance and thus should have different weights, but our initial calculation only allowed weighting by aspect instead of by element. The second issue is that if one applied multiple validation techniques to validate a model element, application of each technique was treated as being equal to applying any other technique. In other words, all validation techniques were given equal weight when applied in the default calculation. In this paper, we address both issues. Using our new coverage computation, not only does each validatable model element have its own weight, but performing validation using a particular technique also has differentiated weight. This improves the flexibility of the coverage computation.

Many different simulation modeling approaches are utilized in the M&S community. Suppose there is a set of validation aspects VA that are relevant for a particular simulation modeling approach such as DES. Each aspect is defined as a set of validatable element types ($va_k \in VA$, $va_k = \{et_1^k, et_2^k, ..., et_i^k\}$ where et_i^k is a validatable element type in the k^{th} aspect va_k, and $k \in \{1,2,3\}$ for our DES framework). For a particular simulation model, we define elements corresponding to each element type within aspect va_k such that et_i^k is a set of elements requiring validation, i.e., $et_i^k = \{e_{i,1}^k, e_{i,2}^k, ..., e_{i,j}^k\}$, with $i \in \{1,2,\ldots,I\}$ where I is the total number of element types for aspect k. Note that et_i^k can be an empty set. For each $e_{i,j}^k$ (j^{th} validatable element of element type i of aspect k) we define the possible validation techniques as the set $vt_{i,j}^k$, and define the set of validation techniques for an element type as vt_i^k. In most cases, we expect to find that $\exists a,b,m,n$ s.t. $vt_a^m \cap vt_b^n| \neq 0$.

We recognize that some validation techniques are likely to be more comprehensive than others. Thus while computing validation coverage, we consider each validation technique $v_{ijl}^k \in vt_{i,j}^k$, where $l \in \{1,2,\ldots,L\}$ and L is the number of applicable validation techniques for element j of element type i, to have its own weight w_l. Validatable elements may also be considered to vary in importance for validation, with weights $r_j = \{r_{j,1}^k, r_{j,2}^k, ..., r_{j,i}^k\}$, corresponding to each element $e_{i,j}^k$ that must be validated. We do not assume that all weights for either applicable validation techniques or validatable elements must sum to one, and take that into account in the validation coverage calculation below.

Our validation coverage (vc) computation is based on whether a specific validation technique $v_{ijl}^k \in vt_{i,j}^k$ has been successfully applied or not. We define the applicability function a and specify that $a(v_{ijl}^k) = 1$ if v_{ijl}^k has been applied successfully on the simulation, and 0 if it was either not applied or was not successful. To calculate the validation coverage we compare how well each element is covered based on how many of the relevant (weighted) validation techniques have been utilized for each (weighted) element. The final validation coverage vc is therefore

$$vc = \frac{\sum_{ijk} r_{ij}^{k} \frac{1}{\sum_l w_l} \sum_l w_l a(v_{ijl}^{k})}{\sum_{ijk} r_{ij}^{k}} \tag{1}$$

where i, j refers to the j^{th} element within the i^{th} element type, and k refers to the k^{th} aspect. The validation coverage is therefore a percentage of weighted applicable validation techniques successfully applied to validate the simulation, and $0 \leq vc \leq 1$. If vc meets the predefined threshold, then enough validation has been performed on the simulation model. This process can enable us to define numeric values to Balci's nominal terms "highly," "exceptionally," and "notably" for describing the level of validation achieved on a simulation model (Balci 2004).

3 Case Study: Hospital ED Simulation

3.1 Applying DES Validation Coverage

To illustrate the use of our coverage computation to quantify validation, we show its application on a DES of patient flow in the Emergency Department (ED) of a hospital (Raunak et al. 2009, Raunak and Osterweil 2013). This simulation model was developed to model the Emergency Department of BayState Medical Center in West Springfield, MA. The model focused on the patient care path of walk-in patients in an ED. The director of ED was interested in investigating the impact of process changes in ED patient care, such as immediate placement of an incoming patient inside the ED when a bed is available followed by bedside registration in parallel with treatment to the patient. The research findings obtained through the experimentation of the model could only be trusted if the model was properly validated. To understand how to compute validation coverage on that simulation model, we need to first be acquainted with the patient care process in an emergency department.

Patient care at a typical ED encompasses a series of sequential activities starting with arrival of the new patient. A patient may arrive as a walk-in patient or she may arrive through an ambulance. The walk-in patients are first seen by a triage nurse for determination of a triage acuity level. The patient is then sent to the registration clerk. The registration clerk enters the patient's insurance and other information into the system and generates an id band. The patient then waits in the waiting room if a bed is not available inside the main treatment area of the ED. Once a bed becomes available, the patient is placed in the bed and is assessed by a nurse and then by the attending doctor. The doctor assessment may result in additional tests or procedures. Afterwards the doctor makes a final assessment and decides whether to admit or discharge the patient. The simulation cycle ends with the patient either discharged or admitted into the inpatient unit of the hospital. The simulation model defines makeup and availability of a number of resources such as doctors, nurses, clerks, beds, X-ray machines, etc. Patient arrival rates and different service time distributions are also simulation parameters.

There are two types of resources (va_1) in an ED: doctors, nurses, and clerks provide services; beds and medical equipment are resources used by the service providers. An arrival of a patient and her treatment generates the requests for services in the simulation. Thus, request characteristics (va_2) include patient arrival rate and the makeup of different patient types. The workflow (va_3) is defined by the service time distribution of the providers and the average length of stay (LOS) experienced by the patients.

3.2 Coverage Computation

Following the process described in Figure 1 with the help of the model element identification provided in Figure 3 and using the proposed weights from section 2.3, we enumerate the entire process of computing the validation coverage for our hospital ED case study. The first step is to identify all validation model elements by using the DES aspect guidelines (Figure 2). Figure 3 shows the relevant aspects and model validation elements for this hospital ED model:

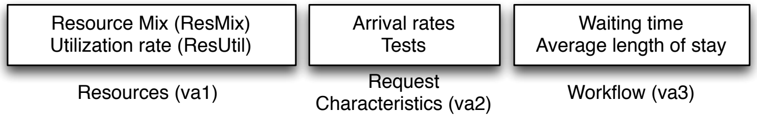

Figure 3: Relevant aspects and elements for the described hospital simulation. All DES aspects are relevant.

- Resource related model elements (va_1)
  - *ResMix* ($e^1_{3,1}$): The mix of resources or providers (how many doctors, nurses and registration clerks, beds, etc.)
  - *ResUtil* ($e^1_{4,1}$): The utilization rate of resources (beds, doctors, nurses)
- Request related model elements (model elements that causes resource usage in simulation) (va_2)
  - *Arrival* ($e^2_{1,1}$): The patient arrival rate
  - *Tests* ($e^2_{3,1}$): The relative makeup of tests and procedures performed on patients
- Workflow related model elements (va_3)
  - *WaitingTime* ($e^3_{2,1}$): Patient wait time for bed placement after registration (waiting room time)
  - *LOS* ($e^3_{3,1}$): The overall length of stay of patients

We next identify each element as either input data (*id*) or observable emerging information (*oei*) that could be validated. We identify ResMix ($e^1_{3,1}$), Arrival ($e^2_{1,1}$), and Tests ($e^2_{3,1}$) as *id*s, while ResUtil ($e^1_{4,1}$), WaitingTime ($e^3_{2,1}$), and LOS ($e^3_{3,1}$) are *oei*s.

Next we add relative weights for each of these model validation elements. In this particular case study, the focus of the simulation was to study the patient LOS and the resource utilization. Consequently, as simulation modeler and validator we placed a higher relative weight of 3 on $e^1_{4,1}$, $e^3_{2,1}$, and $e^3_{3,1}$ when compared to other elements that are given a relative weight of 1.

The fourth step is to perform validation and track which of the validation techniques were successfully performed on the simulation model for each *id* and *oei*. Table 2 shows applicable and used validation techniques, as well as the calculation of the validation coverage score for each element without including the element's weight. For this case study we use the validation techniques and weights from Table 1; otherwise, we would need to define the set of techniques and weights appropriately.

Table 2: Computation of Validation Technique Coverage by Element.

Model Elements	category	Weight	Applicable VTs	VTs Applied	VT Score
ResMix ($e^1_{3,1}$)	id	1	FV-id	FV-id	$\frac{2}{2} = 1$
ResUtil ($e^1_{4,1}$)	oei	3	CM, ET, FV-oei, RV, SA	CM, RV, ET, FV-oei	$\frac{3+4+2+2}{3+2+2+4+2} = .85$
Arrival ($e^2_{1,1}$)	id	1	FV-id, GoF	FV-id, GoF	$\frac{2+3}{2+3} = 1$
Tests ($e^2_{3,1}$)	id	1	FV-id, GoF,	FV-id	$\frac{2}{2+3} = .4$
WaitingTime ($e^3_{2,1}$)	oei	3	CM, ET, FV-oei, RV, SA	ET, FV-oei, RV	$\frac{2+2+4}{13} = .62$
LOS ($e^3_{3,1}$)	oei	3	CM, ET, FV-oei, RV, SA	CM, ET, FV-oei, RV	$\frac{3+2+2+4}{13} = .85$

We perform the fifth and final step to compute our validation coverage by combining the weighted validation application values (rightmost column of Table 2) with the element weights (third column of Table 2) and dividing by the total element weights across all aspects:

$$vc = \frac{1*1+3*.85+1*1+1*.4+3*.62+3*.85}{12} = .7769 = 77.69\% \qquad (2)$$

The result of this calculation demonstrates the interaction between weights and validation techniques. For instance, since Goodness-of-fit (GoF) is the higher ranked of the *id* validation techniques, model element Tests ($e^2_{3,1}$) has a much lower score by only using face validation; if it had only been validated through GoF instead, that element's score would be 0.6, and the overall validation would increase to 79%. The addition of CM for element WaitingTime ($e^3_{2,1}$) would cause an even greater change (up to 83%), even though CM also has a weight of 3, because the element WaitingTime ($e^3_{2,1}$) has been weighted with more importance than element Tests ($e^2_{3,1}$).

4 Summary and Discussion

Validation is primarily a confidence building activity for simulation models. Quantifying this activity and assigning a numeric value to the confidence about the accuracy of a simulation model is considered extremely difficult, if not impossible (Balci 2004). There is a general lack of rigor and standard practice in reporting V&V activities of simulation models (Raunak and Olsen 2014). The Modeling and Simulation (M&S) community has recently identified the need for increased scientific rigor and taken it as a challenge for simulation studies (Taylor et al. 2013).

We previously developed an initial framework for quantifying validation effort (Olsen and Raunak 2013). Our initial effort was simplified and with a focus on application to agent-based simulation models. The primary idea behind our initial validation coverage framework was to identify aspects of a simulation model, determine the model validation elements in those aspects, and validate them using validation techniques. It did not take into account the relative importance of different model elements requiring validation or the relative usefulness of each validation technique. Building on our earlier work, we have presented a detailed approach for developing and computing validation coverage that addresses the previous shortcomings. This research also shows how the coverage computation framework can be applied on DES models, the most commonly used modeling methodology in the community. We have presented a detailed case study using this validation coverage computation of a hospital ED simulation.

Our new validation coverage framework is more flexible and it allows simulation modelers to define relative importance of model validation elements. It also provides mechanism for capturing the level of confidence one can build by using various validation techniques. An additional contribution of our new framework is a detailed suggestion for relative coverage weight and classification of specific validation techniques. We believe these details show ways to reduce the subjectivity in simulation model validation coverage calculations.

General software engineering testing has long benefited from the establishment of test coverage criteria. These criteria have helped to answer such questions as "how much testing is sufficient?" and "when can we stop testing?" in regard to general software. These test coverage criteria also helped standardize the testing effort, and conveyed the amount of testing performed and the level of confidence one can place on a computer program. We have argued that V&V in simulation studies, especially in model validation, will also benefit from the development of validation coverage criteria that help to provide a numeric answer to questions such as "how much validation is sufficient?" or "how confident can we be on the accuracy of a simulation model?" The key insight we present is that just as a test coverage criterion is based on some program element such as a statement or a branch being exercised by a test suite, we can also identify and describe validatable elements in a simulation model that require validation. We have identified two primary categories of validatable elements in a simulation model: input data (id) and observable emerging information (oei). When validating their models, simulation modelers usually concentrate on a few key *id*s and *oei*s. Often the validation revolves around just one simulation output (oei) and using only one validation technique. We have presented a detailed process as to how all such *id*s and *oei*s can be identified for DES models and how multiple validation techniques should be considered while validating those elements.

One reason for a lack of numerical quantification of simulation validation could be the nature of the model validation task. Simulation model validation is similar to verifying software systems that are often called "non-testable" due to the absence of a test oracle. In both cases, it is difficult to know for sure whether the execution of the system, be it regular software such as a "machine learning" system or a simulation model, has resulted in correct behavior or not. Simulations thus fall under the category of software without a deterministic oracle to definitively answer whether it is behaving as expected or not. Consequently validation of simulation models carries some of the same challenges as verifying programs without test oracles.

Although we focus on validation of simulation models, we argue that this approach can be extended to other "non-testable" software. One can identify verification elements of software through the approach of identifying system aspects and element-types of any software without oracles. One can then determine the techniques to be applied for verification or validation of those elements, and compute the coverage in a similar manner as we have shown for simulation models. We believe that the key is to define the coverage on the software elements to be verified or validated, as opposed to focusing on the usual execution of program elements such as lines or branches in the source code.

Focusing only on simulation model validation, our approach provides a framework that enables simulation modelers to carefully think about all elements that may require validation and the validation activities that would best validate them. It also provides a standard approach in capturing the amount of validation performed on a simulation model, thus making the assessment on a simulation model independently verifiable. This contribution has the potential to increase not only the general confidence one usually places on a simulation study, but also may result in increased repeatability of simulation studies, which is something of a rarity in the field. Simulation modeling vendors can utilize this validation criterion calculation as a means of building measurable confidence in the minds of simulation customers. Through the standardization and use of this approach, the simulation industry can significantly benefit from the ability to specify clearer requirements both from the perspective of simulation customers and vendors.

ACKNOWLEDGMENTS

This work was partially funded by the Loyola University Maryland Summer Research Grant Program, and the Clare Boothe Luce Professorship program. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding sources.

REFERENCES

Balci, O. 2004. "Quality Assessment, Verification, and Validation of Modeling and Simulation Application". In *Proceedings of the 2004 Winter Simulation Conferences*, edited by R. Ingalls, M. Rossetti, J. Smith, and B. Peters, 122 – 129. Piscataway, New Jersey: IEEE.

Bertolino, A. 2007. "Software Testing Research: Achievements, Challenges, Dreams". In *FOSE '07 Future of Software Engineering*, 85–103: IEEE.

Chen, L., and A. Avizienis. 1978. "N-version programming: A fault tolerant approach to reliablity of software operation". In *FTCS-8 Proceedings of the 8th Symposium on Fault-Tolerance Computing*, 3–9: IEEE.

Chen, T. Y., T. H. Tse, and Z. Q. Zhou. 2003. "Fault-based testing without the need of oracles". *Information and Software Technology* 45:1–9.

Chillarege, R., I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray, and M. Wong. 1993, Nov. "Orthogonal defect classification-a concept for in-process measurements". *IEEE Transactions on Software Engineering* 18 (11): 943–956.

Ghezzi, C., M. Jazayeri, and D. Mandrioli. 2002. *Fundamentals of Software Engineering*. 2nd ed. NJ, USA: Prentice Hall PTR.

Kanewala, U., and J. M. Bieman. 2013. "Techniques for Testing Scientific Programs Without an Oracle".

Law, A. 2009. "How to build valid and credible simulation models". In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, 24–33. Piscataway, New Jersey: IEEE.

Law, A., and W. D. Kelton. 1991. *Simulation Modeling and Analysis, 2nd Ed.* New York: McGraw Hill.

Olsen, M., and M. Raunak. 2013. “A Framework for Simulation Validation Coverage”. In *Proceedings of the 2013 Winter Simulation Conference*, 1569–1580. Piscataway, New Jersey: IEEE.

Pfleeger, S. L. 2009. *Software Engineering: Theory and Practice*. 4th ed. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Raunak, M., and M. Olsen. 2014. “A Survey of Validation in Health Care Simulation Studies”. In *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller. Piscataway, New Jersey: IEEE.

Raunak, M., L. J. Osterweil, A. Wise, L. A. Clarke, and P. L. Henneman. 2009. “Simulating Patient Flow through an Emergency Department Using Process-Driven Discrete Event Simulation”. In *SEHC ’09 Proceedings of the 2009 ICSE Workshop on Software Engineering in Health Care*, 73–83. Washington, DC: IEEE Computer Society.

Raunak, M. S., and L. J. Osterweil. 2013. “Resource Management for Complex, Dynamic Environments”. *IEEE Transactions on Software Engineering (TSE)* 39:384–402.

Raunak, M. S., L. J. Osterweil, and A. Wise. 2011. “Developing Discrete Event Simulations from Rigorous Process Definitions”. In *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, TMS-DEVS ’11, 117–124. San Diego, CA, USA: Society for Computer Simulation International.

Rosenblum, D. S. 1995. “A Practical Approach to Programming With Assertions”. *IEEE Transactions on Software Engineering* 21 (1): 19–31.

Sargent, R. G. 2010. “Verification and Validation of Simulation Models”. In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 166–183. Piscataway, New Jersey: IEEE.

Taylor, S., S. Brailsford, S. Chick, P. Ecuyer, C. Macal, and B. Nelson. 2013. “Modeling and Simulation Grand Challenges: An OR/MS Perspective”. In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S.-H. Kim, and A. Tolk, 1269–1282. Piscataway, New Jersey: IEEE.

Wang, Z. 2011. “Towards a Measurement Tool for Verfication and Validation of Simulation Models”. In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 581–592. Piscataway, New Jersey: IEEE.

Wang, Z. 2013. “Selecting Verification and Validation Techniques For Simulation Projects: A Planning and Tailoring Strategy”. In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S.-H. Kim, and A. Tolk, 1233–1244. Piscataway, New Jersey: IEEE.

Weyuker, E. J. 1982. “On testing non-testable programs”. *The Computer Journal* 25 (4): 465–470.

Zhu, H., P. Hall, and J. May. 2009, Dec. “Software Unit Test Coverage and Adequacy”. *ACM Computing Surveys* 29 (4).

MOHAMMAD RAUNAK is an Assistant Professor in the Computer Science Department at Loyola University Maryland. His research interests are verification, validation, and analysis of complex software and simulation systems. He is also interested in modeling and anlyzing software development and other human centric processes. His email and web addresses are raunak@loyola.edu and http://www.cs.loyola.edu/~raunak.

MEGAN OLSEN is an Assistant Professor in the Computer Science Department at Loyola University Maryland. Her research interests are in simulation validation, complex systems, and stigmergy. Much of her work is in simulating biological complex systems, primarily of predator-prey systems and cancer cell growth. Her e-mail and web addresses are mmolsen@loyola.edu and http://www.cs.loyola.edu/~olsen.