Applying Statistical Debugging for Enhanced Trace Validation of Agent-Based Models

Ross J. Gore[1], Christopher J. Lynch[1,2*], and Hamdi Kavak[2, 3]

[1] Virginia Modeling, Analysis, and Simulation Center, Old Dominion University

[2] Department of Modeling, Simulation & Visualization Engineering, Old Dominion University

[3] Department of Industrial and Systems Engineering, Turkish Military Academy

Abstract

The verification and validation (V&V) of Agent-based Models (ABMs) is challenging. The underlying structure of the model and the agents can change over time. Furthermore, the theoretical context of the model is often very different from established models of the same phenomenon. In an effort to overcome these issues, Trace Validation is becoming a common V&V mechanism within the Agent-based Modeling community. In Trace Validation, characteristics of agents and the model are tracked over time and then analyzed by subject matter experts (SMEs) to gain insight into unexpected and potentially invalid output. Here, we present our tool, the V&V Calculator, which applies predicates employed in the field of software engineering. The result is a structured trace validation approach with quantifiable measures that facilitates SME exploration and insight into the causes of unexpected output within ABMs.

Keywords: statistical debugging, trace validation, agent-based models, emergent behavior

Submitted 06 September 2015 for the *Special Issue of Simulation: Modelling and Simulation in the Era of Big Data and Cloud Computing: Theory, Framework and Tools*

1. Introduction

The Agent-based Modeling paradigm allows for the direct representation of individual entities and their interactions within an environment to explore system level behaviors [1]. A plethora of work has shown that ABM outcomes can provide novel insight into the phenomenon that modelers did not originally anticipate [1—3]. Furthermore, ABMs also can include different stochastic processes and multiple options for communication. These features create opportunities for implementation errors that also lead to outputs that modelers did not originally anticipate [4].

[*] Corresponding author
Virginia Modeling, Analysis and Simulation Center, Old Dominion University
1030 University Blvd. Suffolk, VA 23435
Email: cjlynch@odu.edu – Phone number: 757-686-6248

Given these inherent complexities, it is important for unexpected ABM outputs to undergo validation.

Trace validation is a means to this end. Trace validation entails periodically collecting data from a simulation during execution (i.e. tracing) in a manner that facilitates the explanation of what agent or model characteristics cause the unexpected output [5]. Specifically, once a SME has observed an unexpected output and decided to employ trace validation, the SME must (1) run the simulation to create traces that generate expected and unexpected output, (2) analyze the traces to find the characteristics of the agents and the model that cause the unexpected output, and (3) determine if the output reflects a valid but previous unanticipated output or an invalid output resulting from an error [6, 7].

The chief drawback of trace validation is that the creation of traces results in large volumes of data that is difficult to analyze and too time-consuming for SMEs to interpret [5, 8]. Our approach for enhanced trace validation directly addresses this challenge by providing a tool for SMEs to automatically search for and quantify the extent to which traced agent and model characteristics within the ABM cause the unexpected output [9].

The novelty of our approach is the application of statistical debugging. Statistical debugging is an area of research within the field of computer science focused on the automatic localization of faults in software. We apply the same principles that localize faults in software to analyze and quantify the extent to which an agent and the model characteristic included in a trace cause an unexpected output. We automate this analysis and provide a graphical interface for SMEs to query the collected traces. The result is a more efficient and effective mechanism for trace validation because the burden of analysis and interpretation for SMEs is reduced.

The remainder of the paper proceeds as follows. Section 2 provides background information on trace validation of ABMs and statistical debugging. Section 3 provides a description of our enhanced trace validation approach and how it is realized in a tool called the V&V Calculator. Section 4 evaluates the effectiveness of our proposed approach relative to several other versions of our tool we implemented. In Section 5, we discuss the validity of evaluation and the assumptions and limitations of our work. Finally, we review our contributions and offer direction for future work.

2. Background

A large number of V&V techniques have been cataloged. These techniques can be classified as informal (rely on human reasoning and subjectivity [5]), formal (based on mathematical proofs of correctness [5, 10]), static (applied to the simulation’s source code while it is not being executed [10]), and dynamic (applied to the simulation during execution under different conditions [11, 12]).

In this paper, we combine two dynamic techniques that utilize simulation data collected during execution: (1) trace validation and (2) statistical debugging. In Sections 2.2 and 2.3, we present background material and work related to each. However, first we discuss work related to construction of ABMs and describe why our approach to enhanced trace validation is independent of ABM construction.

2.1 ABM Construction

The underlying structure of an ABM can change over time and the theoretical context of the model is often very different from established models of the same phenomenon that are constructed using other modeling paradigms. ABMs deal with human-like behaviors or human-like thinking that requires validating both the system level (macro-level) and individual level (micro-level components) of the model [4]. To consider an ABM as valid, agent behaviors, relationships, and interactions must correctly correspond to the individuals that they represent [13]. Windrum et al. [14] identify several challenges with Agent-based Modeling, including the lack of an accepted way to build and analyze ABMs that could contribute to the selection of the best sensitivity analysis approach to V&V the model. While we do not address the ability to build ABMs, our enhanced trace validation approach can be applied to any ABM regardless of the approach used for construction as long as the internal characteristics of the agents and the model can be traced and the outcome of interest can be quantified as an output.

2.2 Trace Validation

Trace validation entails periodically collecting data from a simulation during execution (i.e. tracing) in a manner that facilitates the identification of unexpected outputs and the agent or model characteristics that lead to these outputs. The traces enhance the internal validity of the ABM because they offer a means of comparing multiple simulation replications for the same inputs to identify any inconsistencies in the outputs [6, 8].

Trace Validation includes various forms of execution testing such as execution monitoring, execution profiling, and execution tracing all of which focus on examining micro-level occurrences within the model to reveal errors [5]. Traces track the characteristics of the agent and the model over time to determine if the logic is correct and the simulation produces believable values [11, 15—17].

However, traces require manual analysis by the SME, which is can be too burdensome to be useful [8]. As a result there have been several efforts to provide SMEs with an interface to collect traces. Xiang et al. [8] present a natural organic matter model and a process for conducting validation that includes traces, graphs and charts, and model-to-model comparisons. They use graphical charts to test if the produced data curves match with the expected normal distribution curve and they determine that the traces and the visual methods do increase confidence that the model is correct; however, they also state that statistical methods are needed to further validate the model.

Courdier et al. [18] use the Geamas Virtual Laboratory tool to collect traces of a Biomass ABM for analyzing the management of animal wastes to explore for unexpected outputs. These traces collect either (1) sets of messages exchanged between agents or (2) an accounting of simulation execution per agent or group of agents. They conducted an experiment yielding 20,000 traces of textual messages exchanged between agents. Visualization tools are applied to inspect these traces and identify interactions that lead to successful negotiations between agents. Their visualization features filter the traces based on specific groups of agents or specific characteristics. However, the Geamas Virtual Laboratory tool does not allow for the visual interpretation of traces or graphics for populations in excess of several dozen agents at the individual level [18].

While both of these approaches assist SMEs in the identification of characteristics within traces that cause unexpected outputs, they rely on visual inspection. Visualization can be a powerful mechanism to facilitate insight but its effectiveness, especially relative to other visual representations, is difficult to evaluate. Our enhanced trace validation method facilitates SME insight without visualization by leveraging existing work in the field of statistical debugging. It provides an automated means to search across multiple traces and quantify the extent to which different combinations of agent and/or model characteristics cause an unexpected output within an ABM. Since it does not rely on a visual interpretation for its analysis, it can be easily evaluated against other alternatives.

Next, we review background material needed to understand statistical debugging, then we present our enhanced trace validation approach, and finally we objectively evaluate its effectiveness.

2.3 Statistical Debugging

Our approach to trace validation employs predicates that are used in statistical debuggers. Statistical debuggers isolate the causes of software faults using a set of inputs, corresponding execution traces, and a labeling of the execution traces as passing or failing [19]. The execution traces typically reflect the coverage of individual statements. The debuggers assign suspiciousness scores to statements to guide developers in locating faults. Equations 1-3 define the suspiciousness of a statement *s*.

$$correlation_s = \frac{\#\ of\ failing\ execution\ traces\ including\ s}{\#\ of\ execution\ traces\ including\ s} \tag{1}$$

$$coverage_s = \frac{\#\ of\ failing\ execution\ traces\ including\ s}{\#\ of\ failing\ execution\ traces} \tag{2}$$

$$suspiciousness_s = \frac{2*correlation_s*coverage_s}{correlation_s+coverage_s} \tag{3}$$

The $correlation_s$ of a statement reflects the likelihood of a statement *s* appearing in a failing execution trace, while the $coverage_s$ of a statement reflects the likelihood that a failing execution trace includes statement *s*. The *suspiciousness*$_s$ of a statement balances these two rate measures via the harmonic mean. Developers examine the statements in decreasing order of suspiciousness until the fault is discovered. For the approach to be effective, faulty statements must generally have higher suspiciousness scores than non-faulty statements.

In addition to profiling program statements, most statistical debuggers employ conditional propositions, or predicates, to record the values assigned to variables in an execution trace. For example, three predicates can be instrumented for every assignment statement in a program to test if a value being assigned to a variable is greater than, less than, or equal to zero. The suspiciousness of these predicates is calculated using (1) the failing execution traces where the predicate is true, (2) the total number of execution traces where the predicate is true, and (3) the total number of failing execution traces.

The addition of predicates (including those that are more complex than the three described above) enables statistical debuggers to analyze relationships within and among variable values. In theory and in practice this has been shown to improve effectiveness of the statistical debugging [9, 19—21]. Next, we describe the different types of predicates and how these predicates can be combined.

2.3 Predicates

Statistical debuggers employ two different types of predicates (*single variable*, *scalar pairs*) at two different levels of specificity (*static* and *elastic*) to localize faults. The choice of type and the specificity-level defines a unique combination of conditions related to the variable(s) that the predicate captures. Two or more predicates can also be combined by generating *compound* predicates to gather insight about a variable's behavior at an additional level of granularity. Here we review predicate types, their specificity levels, and describe how they can be combined in a compound predicate.

2.3.1 Single Variable Predicates

A *single variable predicate* partitions the set of possible values that can be assigned to a variable *x*. Single variable predicates can be created at two levels of specificity: the *static* level and the *elastic* level. The most basic single variable predicates are static. *Static single variable predicates* are employed to partition the values for each variable *x* around the number zero: ($x > 0$), ($x \geq 0$), ($x = 0$), ($x \leq 0$) and ($x < 0$). These single variable predicates are referred to as static because the decision to compare the value of *x* to 0 is made before execution. In contrast, the *single variable elastic predicates* use summary statistics of the values given to variable *x* to create partitions that cluster together values which are a similar distance and direction from the mean. For the variable *x* with mean μ_x and standard deviation σ_x, the elastic single variables predicates created are: ($x > \mu_x + \sigma_x$), ($x \geq \mu_x + \sigma_x$), ($\mu_x + \sigma_x \geq x > \mu_x - \sigma_x$), ($\mu_x + \sigma_x \geq x \geq \mu_x - \sigma_x$), ($\mu_x + \sigma_x > x \geq \mu_x - \sigma_x$), and ($x \leq \mu_x - \sigma_x$), ($x < \mu_x - \sigma_x$). These predicates reflect values of variable *x* that are well above their normal value, within their normal range of values and well below their normal value.

2.3.2 Scalar Pair Predicates
Scalar pair predicates capture the important relationships between two variables that elude single variable predicates. The most basic scalar pair variables are static. *Static scalar pair predicates* are employed to partition the difference between a pair of variables, x and y, around the number zero: ($x - y > 0$), ($x - y \geq 0$), ($x - y = 0$) ($x - y \leq 0$) and ($x - y < 0$). These scalar pairs predicates are referred to as static because the decision to compare the difference between x and y to 0 is made before execution. In contrast, the *scalar pairs elastic predicates* use summary statistics of the difference between x and y to create partitions that cluster together values which are a similar distance and direction from the mean. For the pair of variables x and y with mean difference μ_{x-y} and standard deviation σ_{x-y}, the elastic scalar pairs predicates created are: ($x - y > \mu_{x-y} + \sigma_{x-y}$), ($x - y \geq \mu_{x-y} + \sigma_{x-y}$), ($\mu_{x-y} + \sigma_{x-y} > x - y > \mu_{x-y} - \sigma_{x-y}$,), ($\mu_{x-y} + \sigma_{x-y} \geq x - y > \mu_{x-y} - \sigma_{x-y}$), ($\mu_{x-y} + \sigma_{x-y} > x - y \geq \mu_{x-y} - \sigma_{x-y}$), ($\mu_{x-y} + \sigma_{x-y} \geq x - y \geq \mu_{x-y} - \sigma_{x-y}$), ($x-y \leq \mu_{x-y} - \sigma_{x-y}$) and ($x-y < \mu_{x-y} - \sigma_{x-y}$). These predicates reflect differences between the values of x and y that are well above the normal value, within the normal range of values and well below the normal value.

2.3.3 Compound Predicates
Compound predicates reflect any combination of single variable and scalar pair predicates that can be composed using the logical operators $\wedge$ (and) and $\vee$ (or). For any two predicates P and Q, two compound predicates are tested: (1) the conjunction of the predicates ($P \wedge Q$) and (2) the disjunction of the predicates ($P \vee Q$). Once created a compound predicate can be combined with another compound predicate. Previous work in the field of software engineering has shown that there is not a significant benefit to combining compound predicates together more than three times [22].

3. Enhanced Trace Validation with the V&V Calculator
In order to conduct a trace validation of an unexpected output from an ABM, the SME must identify the following: (1) the simulation output of interest; (2) the expected range of values for the output of interest; and (3) the agent and model characteristics relevant to the output of interest. These agent and model characteristics of interest reflect the entities within the simulation to trace during execution. These characteristics include any pieces of information relevant to the agents or the model that the SME thinks are directly dependent upon or that directly affect the output of interest. Next, the simulation must then be instrumented so that each time it is run it creates a trace of the characteristics of interest and records the value of the simulation output. Finally, the simulation is executed across a range of input conditions where the SME anticipates output within the expected range.

It is important to note that these steps are not unique to our trace validation approach. Instead, these steps are required by trace validation. In what follows we describe how our approach, realized in the V&V Calculator, facilitates SME analysis of the traces by quantifying extent to which traced agent and the model characteristics cause outputs falling outside of the specified range.

3.1 Using the V&V Calculator

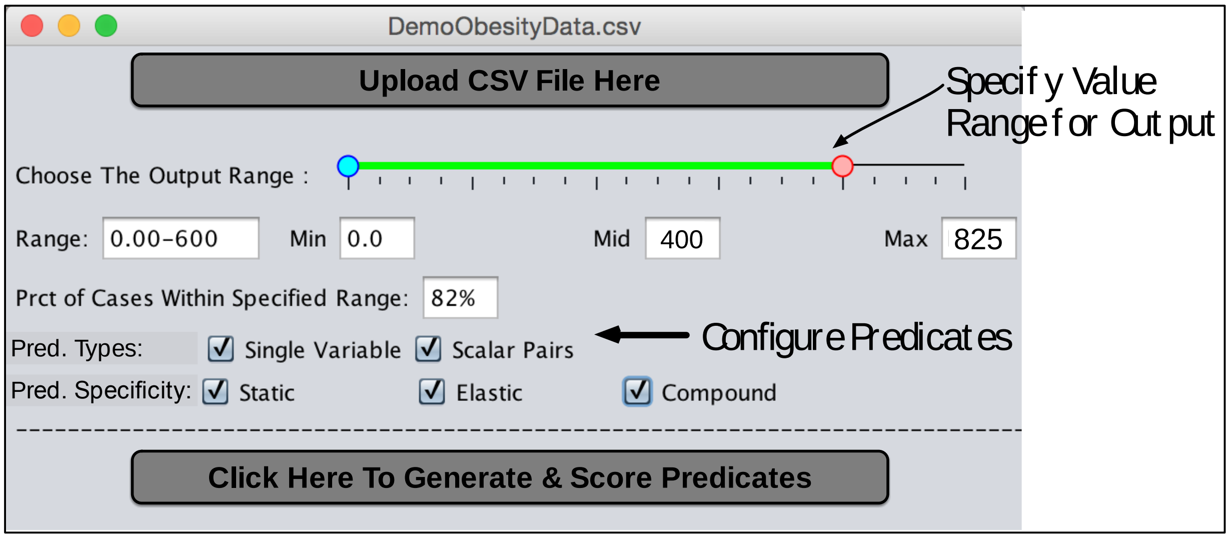
The traces produced by the simulation are aggregated together for use in the V&V Calculator. Once the traces are loaded into the calculator, the calculator notifies the SME if the output of each collected trace is within the specified range. If this is the case, the trace validation is over; there are no unexpected outputs for the simulation and the simulation output is considered valid under the input conditions that generated the traces [5, 10, 12].

However, the trace validation process continues if any output that is produced falls outside of the specified range. Recall, an output that falls outside the specified range does not necessarily reflect an invalid simulation exhibiting an error. Instead, it may reflect output and lower-level interactions that are valid but were not expected by the SME [4]. The V&V calculator facilitates the collection of insight need for SMEs to make this determination.

SMEs pursue insight into agent or model characteristics from the calculator by searching among the traced agent and model characteristics. The goal is to identify those with the strongest effect on the simulation outputs that fall outside the specified range. The search is parameterized by the SMEs specifying the types of statistical debugging predicates among the traced agent or model characteristics that they would like to explore. The specification includes the choice to query single variable, scalar pair, or both types of predicates, and if the predicates will be static, elastic, or compound. SMEs can also specify to exclude any individual or combinations entities of interest from the search.

The interface of the V&V Calculator is shown in Figure 1. In Figure 1 the user has loaded a single file containing all the aggregated traces using the “Upload CSV File Here” button. Furthermore, they have specified the valid range of values for the output using the “Specify Value Range for Output” slider. The calculator has already informed the SME that at least one of the traces produced unexpected output. As a result, the SME expressed predicates of interest for querying using the “Pred. Types” and “Pred. Specificity” checkboxes. The only action remaining before the SME receives analysis to facilitate insight is to click the “Click Here To Generate & Score Predicates” button.

The result of clicking this button is a list of the queried predicates in descending order of suspiciousness. Within the list, each predicate is a condition featuring at least one (often several) agent and model characteristics included in the trace. The application of using statistical debugging predicates to capture different conditions related to any traced model and agent characteristics is the novel contribution of our work. Specifically, it enables the extent to which each generated condition featuring at least one agent or model characteristic causes the unexpected output to be quantified via the suspiciousness measure.


DemoObesityData.csv
Upload CSV File Here
Specify Value Range for Output
Configure Predicates
Click Here To Generate & Score Predicates

Choose The Output Range :

Range: 0.00-600 Min 0.0 Mid 400 Max 825

Prct of Cases Within Specified Range: 82%

Pred. Types: ✓ Single Variable ✓ Scalar Pairs

Pred. Specificity: ✓ Static ✓ Elastic ✓ Compound

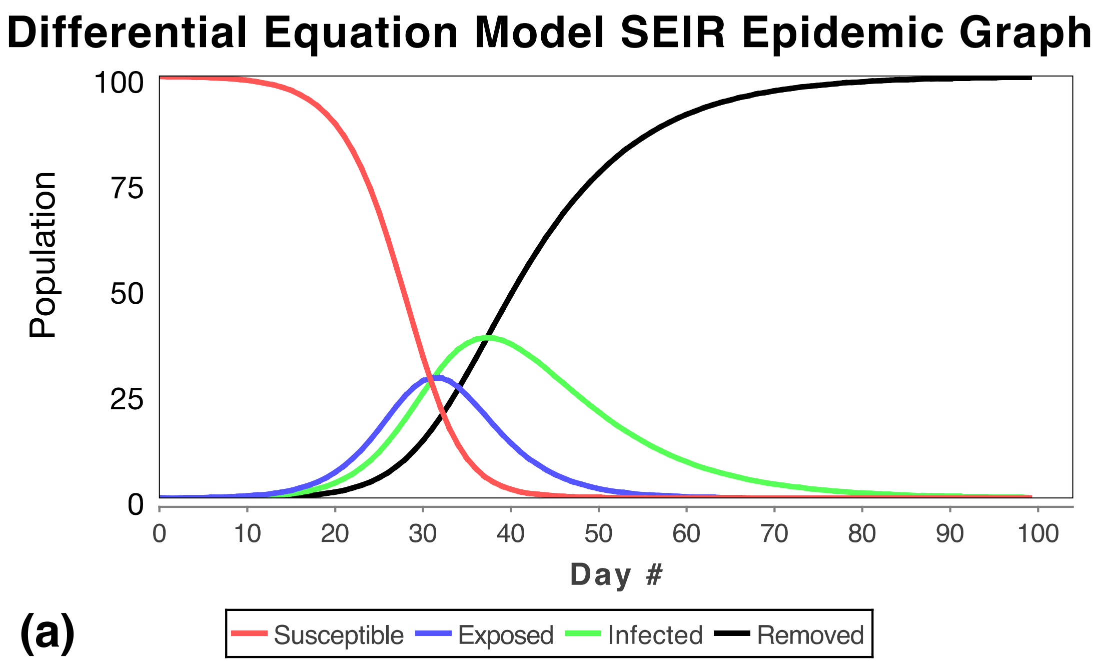
Figure 1. User interface of the V&V Calculator after loading collected traces into the tool.

Recall, the suspiciousness measure reflects two metrics that quantify the extent to which the predicate causes output outside the specified range. These metrics are: (1) *correlation* – the likelihood given the predicate that output outside of the specified range occurs and (2) *coverage* – the likelihood given output outside the specified range that the predicate occurs. Ranking the SME configured predicates in descending order of suspiciousness provides an effective interface for analysis that enables SME insight into the causes of unexpected output. To elucidate the utility of our proposed approach we apply the V&V Calculator to an ABM of epidemic disease spread.

3.2 Example: Epidemic Disease Spread

An example helps elucidate the application and benefits of our enhanced approach to trace validation with the V&V Calculator. Epidemics have been modeled mathematically for over a century. Traditionally, in epidemiology the spread of a SIER infectious disease is described by a system of differential equations that determine the fractions of the population that are susceptible (S), exposed (E), infectious (I), and recovered (R), respectively. Individuals are susceptible, then exposed (in the latent period), then infectious, and then recover with permanent immunity [23].

Recently, researchers have begun to study the spread of SEIR epidemics in a fundamentally different way using agent-based models [24]. Here agents interact on a 2-D landscape and infectious agents search for susceptible agents within a specified radius of their current position and push the infection onto the susceptible population within the radius with a certain probability. Agents travel to work and return home at specified time steps and birth new agents onto the landscape at a specified birth rate. For a number of parameterizations of the ABM the curves (susceptible, exposed, infected, and removed) are a qualitative match to the established differential equation model. Figure 2(a) and Figure 2(b) show these qualitative matches [25].



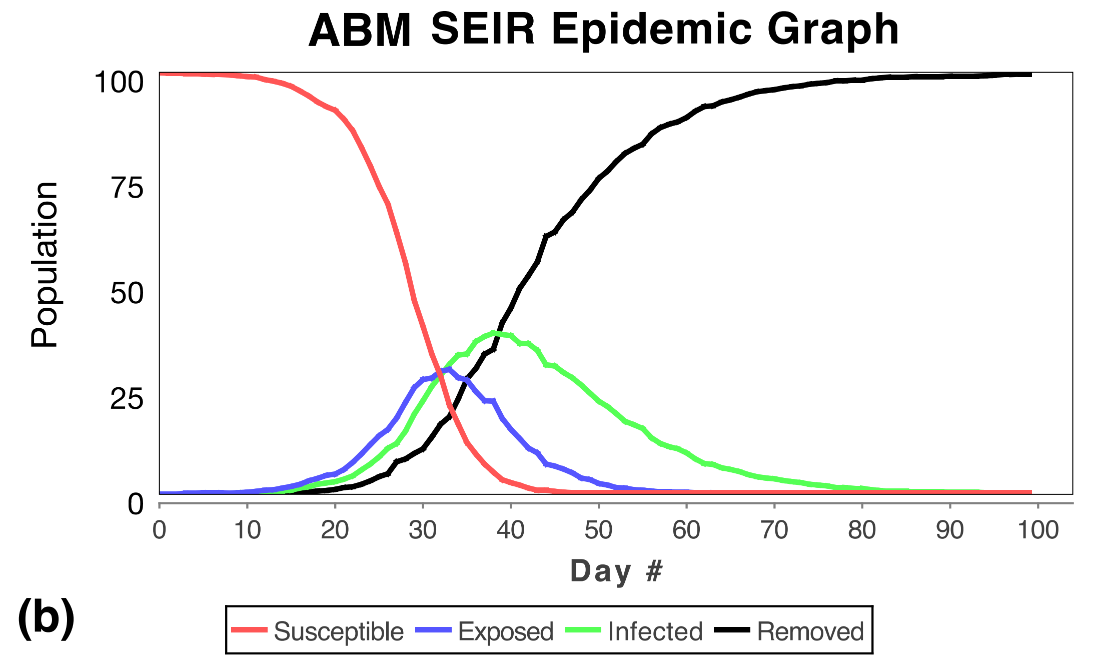


Figure 2. Epidemic graphs of susceptible, exposed, infected, and removed population curves from a differential equation model (a) and an Agent-based Model (b).

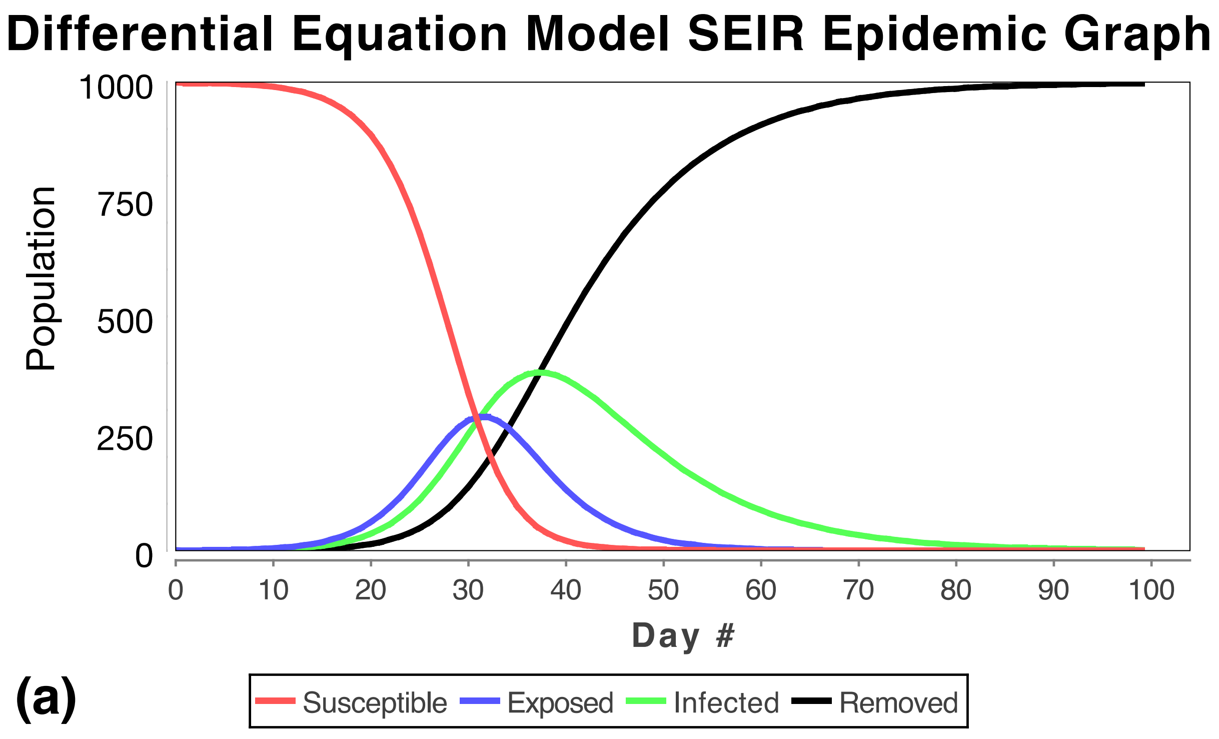
Given the similarity in curves between Figure 2(a) and Figure 2(b), instances where the agent-based model's prediction drastically differs from the differential equation based model's prediction should be validated. The SME needs to understand why the predictions are different and the SME needs to determine the validity of the interactions within the model that cause those predictions. One such circumstance occurs when all of the parameters in both models remain the same but the size of the population of interest is increased from 100 to 1,000 as shown in Figure

3. The difference in the prediction of the disease spread of the two models is shown in Figure 3(a) and Figure 3(b).

To determine if the ABM-generated predictions that drastically differ from differential equation model predictions are valid, we apply our enhanced approach to trace validation. First, we identify our output of interest: the mean standard deviation of the four epidemic curves (S, E, I, and R) produced by the agent-based model from the established differential based curves over 100 days. Then, we specify the valid range for this output: any mean deviation < 5% of the population will be considered a qualitative match of the differential equation model while any deviation > 5 % of the population deviation will not be considered a qualitative match and not an expected output. For context, the average mean standard deviation of the curves shown in Figure 2(b) is 1.5 % while the average mean standard deviation of the curves shown in Figure 3(b) is 14.7 % of the population.

Next, we trace the value of the population size and four characteristics of each agent: (1) *stay* - the number of days thus far the agent has stayed at home and not travel to work due to infection; (2) *othersInfed* - the number of other agents thus far that have been infected by the agent; (3) *expDays* - the number of days thus far the agent has spent in the exposed state (E); and (4) *infDays* - the number of days thus far that the agent has spent in the infected state (I). Then, we run the simulation under different parameterizations including the one shown in figures 2(b) and 3(b). Finally, we collect the traces we load them into the calculator, specify the predicates we want to generate, and score them for suspiciousness.

The most suspicious predicate for the unexpected model predictions, like the one shown in Figure 3(b), is: $othersInfed > \mu_{othersInfed} + \sigma_{othersInfed}$ AND $population > \mu_{population} + \sigma_{population}$. This compound elastic predicate shows that the most drastic differences in the prediction of the agent-based model are due to a significantly higher rate of new infections per individual occurring when the population size is increased. Based on this analysis the SME can determine that the higher rate of new infections per individual is caused by more interactions among the agents due to a denser ABM landscape. Given this explanation it is up to the SME to determine if these interactions were intended and if the prediction is valid. However, the existence of the explanation and the insight that it provides is due to the V&V Calculator's ability to generate, quantify, and rank predicates containing traced agent and model characteristics. Such insight would be challenging to capture without the use of (1) predicates to capture these conditions, and (2) the suspiciousness measure to quantify and rank the conditions. Next, we evaluate the effectiveness of our approach using five different published agent-based models each exhibiting an unexpected output. We compare the effectiveness of our approach to enhanced trace-validation to several other candidate approaches that we have implemented.



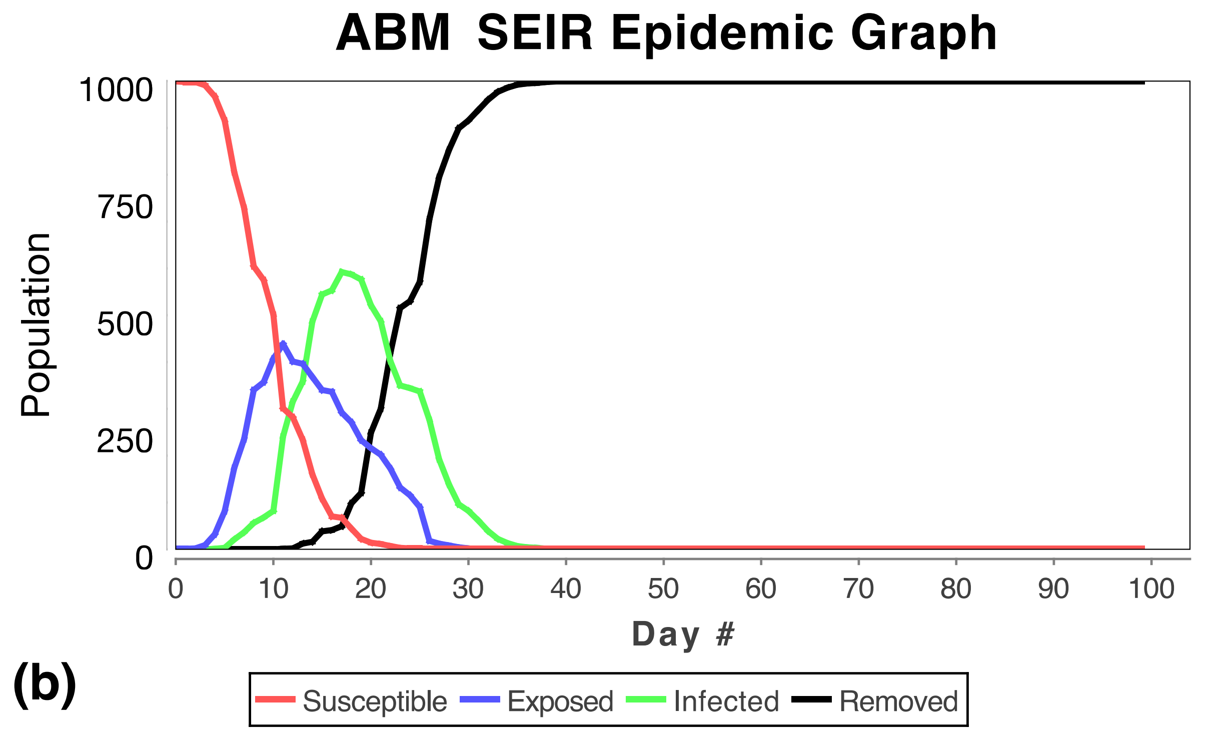


Figure 3. Epidemic graphs of susceptible, exposed, infected, and removed population curves from a differential equation model (a) and an Agent-based Model (b) when using 1,000 people.

4. Evaluation

4.1 Experimental Setup

The utility of a trace validation approach is determined through experimental evaluation. The five different agent-based models and the unexpected output used to conduct our evaluation for each are listed in Table 1. Also listed in Table 1 are the agent and model characteristics traced for each simulation and the subset of those characteristics that cause the unexpected output. These models because they reflect a broad spectrum of established agent-based examples. The traced characteristics and outputs are chosen for each ABM because they reflect published examples of behaviors that at one time did not match the expectation of SMEs. The characteristics responsible for causing the unexpected output of each simulation reflect our review of published explanations of the model's output. For each of the subject simulations we construct an experimental design by applying Latin hypercube sampling with 10,000 samples to the simulation's parameters. A Latin hypercube design yields a sample where each of the dimensions of each variable is divided into equal levels and that there is only one point (i.e. sample) at each level. We use an optimized random procedure to determine the point locations so that good coverage of the design space is ensured as recommended by Cioppa and Lucas [26]. Overall, this setup ensures that our experiments are objective and the evaluation is not biased towards favoring our approach.

Table 1. Subject simulations used in the evaluation of the enhanced trace validation process.

Name	Characteristics Traced	Unexpected Output	Characteristics Responsible
Boids	Height, Width, Separation, Cohesion, Alignment, Orientation Angle, Velocity, Acceleration	Flocking Index > n/2	Separation, Cohesion Alignment
Schelling	Height, Width, Tolerance, # of Neighbors	Mean Similarity < 60%	# of Neighbors, Tolerance
Obesity	Age, Height, BMR, BMI, Calories Per Week, Life Expectancy	Max Weight > 600	Age, Life Expectancy
Self-Driven Particle	Height, Width, Interaction Radius, Random Term, Orientation Angle	# of clusters > n/2	Interaction Radius, Random Term
Epidemic Disease Spread	Population, Stay, OthersInfed, expDays, infDays	Avg. Std. Dev. > 5	Population, Others Infected

Boids

Boids models the flight patterns of birds on a two-dimensional grid [27]. Height and width parameters control the area of the grid. Within the simulation, three parameters control the behavior of each bird: separation; alignment; and cohesion. Each parameter reflects the mean value of a normal distribution and each boid is instantiated by sampling values from these distributions. The degree of separation controls the extent to which the birds avoid crowding each other. Alignment controls the degree to which each bird steers to the average head of the other birds. The cohesion parameter controls the amount that each bird moves toward the average position of the other birds [28—30].

The output of the simulation is the mean flocking index of the birds [31]. Under most conditions the model produces a low flocking index reflecting birds flying along independent paths. However, under certain conditions the birds convene into a single flock flying along one path creating a high flocking index. This characteristic of the simulation has made it an exemplar of designing decentralized systems through an agent-based paradigm.

In the evaluation we track the height and width of the canvas along with the separation, alignment, and cohesion of each bird. Additionally, the trace tracks the following characteristics of each bird over time: the orientation angle of the bird, the current speed of the bird, and the change in acceleration from the previous time step of the bird. A trace producing an unexpected output occurs if there is a mean flocking index greater than half the size of the population occurring at any point after 100 time steps in the simulation. This implementation is described in Quera et al. [31].

We chose this output because most observers do not expect to observe the boids flocking since there is no centralized direction given to the boids. As a result when one first observes this behavior, an explanation is needed to ensure the model is valid [32].

Schelling's Model of Segregation

The Schelling model of segregation explores how the decisions of individuals to move based on their relative positions to their neighbors can lead to segregation. In the model, each agent belongs to one of two groups and aims to reside within a neighborhood populated by similar agents. Agents in the simulation continually relocate based on their neighbors until achieving a steady state. The degree of segregation of the steady state is referred to as the mean similarity. This reflects the average percentage of similar individuals within a neighborhood during the steady state. A steady state is achieved when the mean similarity of each neighborhood in the simulation does not increase or decrease by 1% in ten consecutive time steps. Under most parameterizations, a mean similarity of at least 75% occurs by the time that the simulation reaches a steady state. This implementation is described in Schelling [33].

How and when agents move are controlled by the following conditions: (1) the dimensions of the urban area; (2) agent-density of the urban area; and (3) the extent of each agent's tolerance – the agent's willingness to reside in a neighborhood with dissimilar agents [34, 35]. The tolerance of each agent is chosen from a uniform distribution with a specified mean. Along with the height and width of the canvas and the tolerance of an agent, the trace also tracks the number of neighbors the agent has over time. A trace producing an unexpected output occurs if the mean similarity is less than 60% after a steady state is achieved.

We chose this output because given the ability of agents to move to areas with similar neighbors, most observers expect segregation. As a result when one first observes an output reflecting integrated neighborhoods an explanation is needed to ensure the model is valid [36].

Obesity ABM

The agent-based obesity model included in our evaluation shows how the availability of different restaurant choices in an area affects obesity. The simulation contains four types of entities: people, homes, restaurants, and workplaces. Inputs include the number of people, the eating habits of the people, the starting age and weight levels of the people, the number of workplaces, the number of restaurants, and the types of restaurants. Everyday each person eats three meals, which are obtained from the restaurants. Each meal contributes a number of calories to the individual based on the restaurant's type. People choose restaurants based on their current location. At the end of every week, each person's weight adjusts based on the number of calories that they consumed during the week compared against the number of calories that they needed to maintain their weight level. Calorie levels are reset each week and the obesity of each individual is tracked over time. Agents are removed from the population as a stochastic function of their life expectancy, which is influenced by their level of obesity. This implementation is described in [37].

The trace tracks six characteristics of each agent in the simulation over time: (1) the age of the agent; (2) the height of the agent; (3) the BMR (basal metabolic rate) of the agent, which provides the number of calories that the agent needs each day to maintain its current weight; (4) the BMI (body mass index) of the agent, which is a screening measure for obesity; (5) the average number of calories that the agent consumes in a week; and (6) the life expectancy of the agent given the previously mentioned variables. The output of interest in the model is the maximum weight in pounds of any living person in the population. A trace producing an unexpected output features at least one person whose maximum weight is more than 700 pounds.

We chose this output because it reflects a liberal estimate for the maximum weight of an agent. When agents are produced that exceed 700 pounds the model is invalid. Explanation is required so the model can be modified to ensure that 700-pound agents are not produced [21].

Self-Driven Particle Model

In the self-driven particle model agents interact on a 2-dimensional torus according to a simple rule. Particles move at a constant speed, and their interaction radius and a random term control their orientation. The interaction radius reflects the maximum distance a neighbor must be from a particle to influence their orientation. The random term reflects the degrees of randomness added to the average orientation of all particles within the interaction radius [38]. The implementation of this model is provided in Wieman et al. [39].

Under most parameterizations particles form clusters. However, under some parameterizations the particles exhibit a different behavior. Rather than joining a distinct cluster, each particle roams in a random walk. This behavior is called Spontaneous Symmetry Breaking [40].

The trace tracks the height and width of the torus, the interaction radius, and the random term employed by each agent along with the orientation angle of the agent over time. Any simulation run where there are more clusters than half the number of agents in the population reflects Spontaneous Symmetry Breaking and is considered an unexpected output.

We chose this output because given the influence of neighboring particles on one another most observers expect coordinated movement among the particles. As a result when one first observes the particles moving independently along random walks an explanation is needed to ensure the model is valid [41].

Epidemic Disease Spread ABM

The epidemic disease spread model, its implementation, unexpected output, and agent entities included are the same as those described in the example presented in Section 3.2. Recall, under many parameterizations the ABM produces a prediction that matches the differential equation model of epidemic disease spread. However, under certain parameterizations the predictions of the two models drastically differ. In these cases the output of the ABM needs to be explained to understand why it differs from the differential equation model and under what conditions it is valid [25].

4.2 Competing Approaches

Four different versions of our enhanced trace validation approach are featured in the evaluation: Random-Walk (RW), Correlation-Only (Corr), Coverage-Only (Cov) and Suspiciousness (Susp). Each approach generates the same predicates for each simulation (static, elastic, single variable, scalar pairs and compound predicates). However, the versions differ based on the measure they use to rank predicates related to an unexpected output. In what follows, we summarize the similarities and the differences of the ranking strategies.

Random-Walk (RW)

The RW version of our enhanced trace validation approach puts generated predicates reflecting the traced agent and model characteristics in a random order and presents them to the SME. It is included within the evaluation as a baseline, which any version of our approach must outperform.

Correlation-Only (Corr)

The Corr version of our enhanced trace validation approach uses the correlation measure for predicates described in Equation 1. Recall, this measure reflects the likelihood that an execution trace exhibiting the predicate featuring the traced agent and model characteristic(s) will produce outputs outside of the specified range. Predicates from the Corr version are ranked in descending order of correlation and presented to the SME. This version is included in the evaluation to determine the extent to which using correlation alone enables effective trace validation.

Coverage-Only (Cov)

The Cov version of our enhanced trace validation approach uses the coverage measure for predicates described in Equation 2. Recall, this measure reflects the likelihood that an execution trace producing output outside of the specified range will exhibit the predicate featuring the traced agent and model characteristic(s). Predicates from the Cov version are ranked in descending order of coverage and presented to the SME. This version is included in the evaluation to determine the extent to which using coverage alone enables effective trace validation.

Suspiciousness (Susp)

The Susp version of our enhanced trace validation approach combines the correlation and coverage measure for predicates using the harmonic mean. This measure is described in Equation 3. Predicates from the Susp version are ranked in descending order of suspiciousness and presented to the SME. This is the version of our enhanced trace validation approach that is proposed and described in the preceding sections. It is included in the evaluation to determine the extent to which using a measure, which factors in correlation and coverage alone enables superior trace validation.

4.3 Effectiveness

Given a ranked set of predicates, *Cost* measures the percentage of predicates a developer must examine before encountering a predicate which: (1) includes all of the agent and model characteristics causing the unexpected output; and (2) includes only the agent and model characteristics that cause the unexpected output.

If there are ties, it is assumed that the developer must examine all of the tied predicates. For example, if there are *n* predicates scored by an approach and all *n* predicates have the same score, it is assumed that the developer must examine all *n* predicates. A lower *Cost* is preferable because it means that fewer of the predicates must be considered by the SME before the characteristics causing the unexpected output are found.

It is important to note that this evaluation strategy does not consider the relationship among the agent and model characteristics causing the unexpected output. This is a conscious choice made to reflect the variability surrounding how SMEs gain insight into unexpected outputs. Reasonable SMEs may disagree as to how an ideal predicate would capture the relationship among the agent and model characteristics creating the unexpected output. We avoid these concerns by employing an evaluation metric that does not take this into account.

The effectiveness results of employing the approaches included in the evaluation for the subject simulations are shown in Table 2. Each row in Table 2 shows a *Cost* range, and the number (and percentage) of subjects for each approach that incur the specified *Cost*. Figure 4 provides a graphical view of this data, where the x-axis represents the lower bound of each *Cost* range and the y-axis represents the percentage of subjects where a Cost less than or equal to the upper bound is incurred. This presentation of data follows the established convention in the statistical debugging community [42].

Table 2. Number and (percentage) of predicates that need to be searched through by the SME within the ranked list in each score range for all approaches.

***Cost* Range**	**RW**	**Corr**	**Cov**	**Susp**
< 1 %	0 (0 %)	0 (0 %)	0 (0 %)	0 (0 %)
< 2 %	0 (0 %)	1 (20 %)	1 (20 %)	1 (20 %)
< 4 %	0 (0 %)	1 (20 %)	1 (20 %)	2 (40 %)
< 8 %	0 (0 %)	2 (40 %)	2 (40 %)	3 (60 %)
< 16 %	1 (20 %)	3 (60 %)	3 (60 %)	5 (100%)
< 32 %	2 (40 %)	4 (80 %)	5 (100%)	5 (100%)
< 64 %	3 (60 %)	4 (80 %)	5 (100%)	5 (100%)
≤100 %	5 (100%)	5 (100%)	5 (100%)	5 (100%)

The Susp version of our approach outperforms each of the other alternatives in our evaluation. For each of the five subject simulations it lists the agent and/or model characteristics causing the unexpected output in the top 10% of the ranked predicates. Furthermore, for three of the five simulations it ranks the causing characteristics in the top 4% of the ranked predicates. In what follows, we discuss the difference between the performance of Susp and each of the other versions in more detail.

Susp Vs. RW

The Susp version of our evaluation approach significantly outperforms RW. This is expected. Recall, RW is included in the evaluation as a baseline that any version of our approach must exceed. However, the extent to which all of the other versions outperform RW is noteworthy. No other approach needed to examine more than 60% of the generated predicates for a subject ABM, while the RW needed to examine this many predicates for two different subject simulations (Boids and Self-Driven Particle).

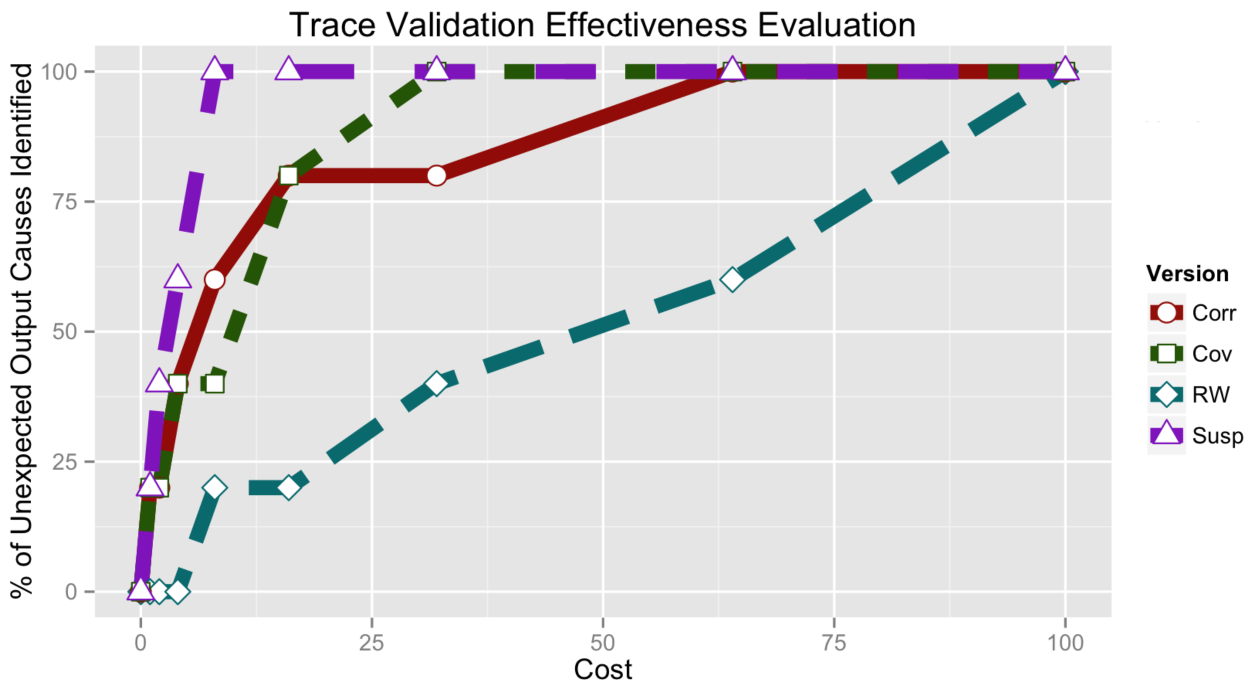


Figure 4. Evaluation of the effectiveness. Higher and further to the left is better.

Susp Vs. Corr

For two of the five subject simulations (Obesity and Epidemic) the Corr version of our approach performs extremely well. It ranks the agent and/or model characteristics causing the unexpected output of these simulations within the top 8% of the generated predicates. However, for the other three simulations its inability to factor in the number of times that an agent or any model characteristic appears in traces that produce unexpected output hampers its effectiveness. In each of these cases Corr over-ranks elastic predicates that only occur in one or two traces producing the unexpected output. These predicates do not capture the different ways in which the unexpected output can be produced. As a result, the Corr version of the approach pushes down more useful predicates that cover all the agent and/or model characteristics causing the unexpected output. The Susp version does not fall victim to this issue because it includes a measure that quantifies how often a trace, which produces unexpected output, includes each predicate. This ensures that only the agent and model characteristics that occur frequently and exclusively in traces that produce unexpected output are ranked at the top of the list.

Susp Vs. Cov

The Cov version of our approach performs well. It ranks the agent and model characteristics causing the unexpected output of four of the five simulations within the top 16% of the list. Furthermore, for the final simulation it ranks the predicate containing any agent or model characteristics causing the unexpected output in the top 32% of the list. Ultimately, what reduces the effectiveness of the Cov version is that it does not distinguish between a predicate in a trace that does produce expected output versus a predicate in a trace that does not. The result is an approach that ranks predicates containing the agent and model characteristics, which appear in all traces the same as predicates containing the agent and model characteristics that only appear in traces that produce unexpected output. The Susp version does not fall victim to this issue because it includes a measure that quantifies how often a predicate results in unexpected output. This ensures that only the agent and/or model characteristics that occur frequently and exclusively in traces that produce unexpected output are ranked at the top of the list.

Susp Vs. Others

The suspiciousness measure used in Susp facilitates SME inspection and analysis of ABM traces. This capability of Susp is similar to the Geamas Virtual Laboratory (GLV) tool [8] and the Trace Validation Interface (TVI) proposed by [18]. However, the two approaches are orthogonal. GLV and TVI facilitate SME inspection of trace data by efficiently creating on-demand graphics. In contrast, Susp employs automated analysis on the collected data to rank conditions possibly causing the unexpected output.

These different capabilities make the approaches hard to compare against one another. It is possible that some SMEs could use GLV or TVI to produce a graphic that provided an explanation of an unexpected output faster than if they used Susp to rank conditions causing the output. However, it is also possible that another SME could choose to use GLV or TVI to create different graphics and as a result of this choice would be unable to explain the unexpected output at all. In future work we will explore how the visualization capabilities provided in GLV and TVI can be incorporated into Susp to improve its effectiveness.

5. Discussion

5.1 Validity

Internal, external, and construct validity threats affect our evaluation. Internal validity threats arise when factors affect the dependent variables without evaluators' knowledge. It is possible that some implementation flaws could have affected the evaluation results. However, we conducted a code review on each of the trace validation approaches we implemented and the source code of the subject agent-based models were obtained from supplemental materials of published research. Threats to external validity occur when the results of our evaluation cannot be generalized. Although we performed our evaluations on five agent-based models and three different fault localization approaches, we cannot claim that the effectiveness observed in our evaluation can be generalized to other unexpected outputs in other agent-based models. Threats to construct validity concern the appropriateness of the metrics used in our evaluation. More studies into how SMEs gain insight from collected traces need to be performed. However, currently we are not aware of any evaluations. As a result, our evaluation and effectiveness metrics serve as a starting point for other researchers to improve upon.

5.2 Limitations

It is important to note that all of the causes of unexpected output in our evaluation reflect agent and model characteristics that are included in traces. A notable limitation of our work is that it is incapable of identifying any characteristic that causes an unexpected output, which is not included within the traces. Furthermore, our evaluation only includes models exhibiting a single type of unexpected output. This is relatively uncommon. During the initial stages of development, models often feature multiple types of unexpected outputs. Adapting our enhanced validation approach so that it is effective for models with multiple types of unexpected outputs is an opportunity for future work.

6. Conclusion and Future Work

Our goal is to develop an effective and structured approach for Trace Validation to facilitate SMEs insight into the causes of unexpected outputs. As a means to this end, we developed the V&V Calculator, which applies statistical debugging to enhance trace validation. Specifically, it enables SMEs to aggregate collected traces of agent and/or model characteristics together and quantifies the extent to which combinations of those characteristics cause an unexpected output. Our approach is evaluated using against several alternatives we implemented and is shown to be the most effective version for five different agent-based models each exhibiting an unexpected output. Ultimately, these contributions further the state of the art in trace validation for SMEs tasked with V&V. In the future we will explore how SMEs gain insight from the V&V Calculator and how to adapt its analysis to simulations with multiple types of unexpected output.

Notes

The V&V Calculator is freely available on GitHub at https://github.com/rossgore/IVandVLevelChecker/

References

1. Gilbert N. *Agent-Based Models*. SAGE Publications. Thousand Oaks, CA, USA: SAGE Publications, 2008.

2. Bonabeau E. Agent-based modeling: Methods and techniques for simulating human systems. *PNAS* 2002; 99(3): 7280–7287.

3. Macal CM, North MJ. Tutorial on agent-based modelling and simulation. *J Simul* 2010; 4(3): 151–162.

4. Takadama K, Kawai T, Koyama Y. Micro- and Macro-Level Validation in Agent-Based Simulation: Reproduction of Human-Like Behaviors and Thinking in a Sequential Bargaining Game. *J Artif Soc Soc Simul* 2008; 11(2).

5. Balci O. Verification, Validation and Testing. In: Banks J, editor. *Handbook of simulation: Principles, methodology, advances, applications, and practice*. Hoboken, NJ: John Wiley & Sons, Inc., 1998. pp.335–393.

6. Bharathy GK, Silverman BG. Validating agent based social systems models. In: *Proceedings of the Winter Simulation Conference* (eds B Johansson, S Jain, and J Montoya-Torres), Baltimore, MD, 2010, pp.441–453. IEEE.

7. Sargent RG. Verifying and validating simulation models. In: *Proceedings of the Winter Simulation Conference* (eds A Tolk, SY Diallo, IO Ryzhov, L Yilmaz, SJ Buckley, JA Miller). Savannah, GA, 2014, pp.118–131. IEEE.

8. Xiang X, Kennedy R, Madey G, Cabannis S. Verification and validation of agent-based scientific simulation models. *ADS'05*. San Diego, Ca, 2005, pp.47–55. SCS.

9. Gore R, Reynolds Jr. PF, Kamensky D. Statistical debugging with elastic predicates. In: *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering*, 2011, pp.492–495. IEEE.

10. Whitner RB, Balci O. Guidelines For Selecting And Using Simulation Model Verification Techniques. In: *Proceedings of the Winter Simulation Conference* (eds E MacNair, K Musselman, and P Heidelberger). Washington, D.C., 1989, pp.559–568. IEEE.

11. Sargent RG. An overview of verification and validation of simulation models. In: *Proceedings of the Winter Simulation Conference* (eds A Thesen, H Grant, and WD Kelton). New York, NY, 1987. pp.33–39. ACM.

12. Sargent RG. Verification, validation and accreditation of simulation models. In: *Proceedings of the Winter Simulation Conference* (eds P Fishwick, K Kang, J Joines, and R Barton). Orlando, FL, 2000. pp.50–59. SCS.

13. North MJ, Macal CM. *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*. New York, NY: Oxford University Press, 2007.

14. Windrum P, Fagiolo G, Moneta A. Empirical Validation of Agent-Based Models: Alternatives and Prospects. *J Artif Soc Soc Simul* 2007; 10(2).

15. Bagni R, Berchi R, Cariello P. A Comparison of Simulation Models Applied to Epidemics. *J Artif Soc Soc Simul* 2002; 5(3).

16. Xu C, He HS, Hu Y, Chang Y, Li X, Bu R. Latin hypercube sampling and geostatistical modeling of spatial uncertainty in a spatially explicit forest landscape model simulation. *Ecol Modell* 2005; 185(2-4): 255–69.

17. Epstein JM, Axtell RL. *Growing artificial societies: social science from the bottom up*. Washington D.C., USA: Brookings Institution Press, 1996.

18. Courdier R, Guerrin F, Andriamasinoro FH, Paillat J-M. Agent-based simulation of complex systems: Application to collective management of animal wastes. *J Artif Soc Soc Simul* 2002; 5(3).

19. Liblit B, Naik M, Zheng AX, Aiken A, Jordan MI. Scalable statistical bug isolation. *ACM SIGPLAN Not* 2005 ;40(6): 15.

20. Gore R, Diallo S. The need for usable formal methods in verification and validation. In: *Proceedings of the Winter Simulation Conference* (eds R Pasupathy, S Kim, A Tolk, R Hill, ME Kuhl). Washington D.C., 2013. pp.1257–1268. IEEE.

21. Gore R, Reynolds Jr. PF, Kamensky D, Diallo S, Padilla J. Statistical Debugging for Simulations. *ACM Trans Model Comput Simul* 2015; 25(3): 1–26.

22. Nainar PA, Chen T, Rosin J, Liblit B. Statistical debugging using compound boolean predicates. In: *Proceedings of the 2007 international symposium on Software testing and analysis*. New York, NY, 2007. pp.5–15. ACM.

23. Li MY, Muldowney JS. Global stability for the SEIR model in epidemiology. *Math Biosci* 1995; 125(2): 155–64.

24. Dunham JB. An agent-based spatially explicit epidemiological model in MASON. *J Artif Soc Soc Simul* 2005; 9(1): 231–44.

25. Gore R, Reynolds PF. Applying causal inference to understand emergent behavior. In: *Proceedings of the Winter Simulation Conference* (eds S Mason, R Hill, L Monch, O Rose). Miami, FL, 2008. pp.712–721. ACM.

26. Cioppa TM, Lucas TW. Efficient Nearly Orthogonal and Space-Filling Latin Hypercubes. *Technometrics* 2007; 49(1): 45–55.

27. Reynolds CW. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Comput Graph* 1987; 21(4): 25–34.

28. Gilbert N. Agent-based social simulation: dealing with complexity. *Complex Syst Netw Excell* 2004; 9(25): 1–14.

29. Epstein JM. *Generative Social Science: Studies in Agent-Based Computational Modeling*. Princeton, NJ: Princeton University Press, 2006.

30. Salge C, Polani D. Digested information as an information theoretic motivation for social interaction. *J Artif Soc Soc Simul* 2011; 14(1): 5.

31. Quera V, Herrando S, Beltran FS, Salas L, Miñano M. An index for quantifying flocking behavior. *Percept Mot Skills* 2007; 105(3): 977–87.

32. Bajec IL, Zimic N, Mraz M. The computational beauty of flocking: boids revisited. *Math Comput Model Dyn Syst* 2007; 13(4): 331–47.

33. Schelling TC. *Micromotives and macrobehavior*. New York, NY: WW Norton & Company, 2006.

34. Hatna E, Benenson I. The schelling model of ethnic residential dynamics: Beyond the integrated - segregated dichotomy of patterns. *J Artif Soc Soc Simul* 2012; 15(1).

35. Hegselmann R. Thomas C. schelling and the computer: Some notes on schelling's essay "on letting a computer help with the work." *J Artif Soc Soc Simul* 2012; 15(4).

36. Rossiter S, Noble J, Bell KRW. Social Simulations: Improving Interdisciplinary Understanding of Scientific Positioning and Validity. *J Artif Soc Soc Simul* 2010; 13(1).

37. Hammond RA. Complex systems modeling for obesity research. *Prev Chronic Dis* 2009; 6(3): 1–10.

38. Nishinari K, Sugawara K, Kazama T, Schadschneider A, Chowdhury D. Modelling of self-driven particles: Foraging ants and pedestrians. *Phys A Stat Mech its Appl* 2006; 372(1): 132–41.

39. Wieman CE, Adams WK, Perkins KK. PhET: simulations that enhance learning. *Science* 2008; 322(5902): 682–3.

40. Vicsek T, Czirk A, Ben-Jacob E, Cohen I, Shochet O. Novel type of phase transition in a system of self-driven particles. *Phys Rev Lett* 1995; 75(6): 1226–9.

41. Huepe C, Aldana M. Intermittency and clustering in a system of self-driven particles. *Phys Rev Lett* 2004; 92(16): 1–4.

42. Jones A, Harrold MJ. Empirical evaluation of the tarantula automatic fault-localization technique. In: *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. New York, NY, 2005. pp.273–282. ACM.