Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

APPLYING CAUSAL INFERENCE TO UNDERSTAND EMERGENT BEHAVIOR

Ross Gore
Paul F. Reynolds, Jr.

Modeling and Simulation Technology Research Initiative
151 Engineer's Way, P.O. Box 40070
Department of Computer Science, University of Virginia
Charlottesville, VA 22904, USA

ABSTRACT

Emergent behaviors in simulations require explanation, so that valid behaviors can be separated from design or coding errors. Validation of emergent behavior requires accumulation of insight into the behavior and the conditions under which it arises. Previously, we have introduced an approach, Explanation Exploration (EE), to gather insight into emergent behaviors using semi-automatic model adaptation. We improve our previous work by iteratively applying causal inference procedures to samples gathered from the semi-automatic model adaptation. Iterative application of causal inference procedures reveals the interactions of identified abstractions within the model that cause the emergent behavior. Uncovering these interactions gives the subject matter expert new insight into the emergent behavior and facilitates the validation process.

1 INTRODUCTION

In the past half century quantitative methods of analysis have been developed to take advantage of computational resources. Simulation is gaining prominence as the proper tool of scientific analysis under circumstances where it is infeasible or impractical to study a system directly. As models have become more complex, our capacity to understand their behavior and determine their validity has suffered. Poorly understood model behaviors which compromise overall model validity are dangerous.

Understanding *emergent* model behavior is challenging. Emergent behavior can arise from seemingly unrelated phenomena, or it can reflect an error in a model or its implementation. Behavior is emergent if it is unexpected and stems from the interactions of the underlying components of the model (Johnson 2006).

The challenge of establishing validity of emergent behavior is evident in the results of recent epidemiology studies. Epidemiologists have explored government policy for controlling the spread of infectious diseases such as smallpox and bird flu. Should a comprehensive vaccination program be initiated? How, and for how long should infected individuals be isolated? Answers to these questions are full of conflict. Recently Elderd, (Elderd 2006) has shown analytically that just four of the potentially hundreds of independent variables in these studies induce extreme sensitivity in model predictions, leading to serious conflict regarding remedial approaches involving billions of dollars and millions of people. Subject matter experts must be given additional capabilities to understand the behavior of their models so that model results can be used effectively and with confidence.

Our goal is to design and develop a novel approach to observed simulation behavior hypothesis testing that allows users to validate or reject emergent model behaviors efficiently, and with confidence. Validating an emergent behavior, is different from validating a simulation, which is a demonstration that a simulation meets expected behaviors. Emergent behavior validation is a demonstration that an unexpected behavior is valid (or not) for a given set of conditions, or experimental frames (Zeigler 2000). Methods to validate simulations exist (Balaci 1997). Emergent behavior validation is an active area of research. We have introduced "Explanation Exploration" (EE) (Gore 2007) for demonstrating that a given emergent behavior is valid. EE allows a subject matter expert (SME) to test hypotheses about the emergent behavior as a simulated phenomenon is driven towards conditions of interest.

In this paper, we take EE a significant step further by offering SMEs additional insight into the interactions of abstractions causing emergent behavior in a model. By applying causal inference procedures we can reveal the interactions of identified abstractions in the model which cause the emergent behavior. Uncovering these interactions gives the subject matter expert new insight into the emergent behavior and facilitates the validation process.

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

2 PREVIOUS WORK

Software engineers have addressed exploration of unexpected behaviors. We review research in static and dynamic program analysis, Delta Debugging, and causal inference, and its applications.

Program slicing is a decomposition that extracts the program statements that affect the value of a target variable (Tip 1995). Static program slicing extracts these program statements for all possible runs of the program. Dynamic program slicing extracts them for one run of the program. The program understanding community has successfully used dynamic and static analysis to help programmers gather insight into program behavior (Eisenbarth 2001). These techniques have limited their focus to gathering insight into deterministic programs on the source code level.

Delta Debugging is an automated approach to program debugging that isolates the causes of failing test cases systematically. Delta Debugging requires a program run that passes the test case, and one that fails it. The cause of a failure is isolated by assessing outcomes of altered executions to determine whether a change in the program state makes a difference in the test outcome (Zeller 2002). Delta debugging does not assume changes are ordered, but requires user choice of an instrumentation point and does not preserve validity of execution traces (Groce 2004).

Causal inferencing finds cause-effect relationships among observed variables to explain a set of observations. Causal inference theory unites two pieces of mathematics and one piece of philosophy. The mathematical pieces are directed acyclic graphs (DAGs) and probability theory (with the focus on conditional independence), and the philosophy involves causation among variables (Spirtes 2001, Pearl 2000). In causal inference theory, DAGs are given two distinct functions: to represent sets of probability distributions and to represent causal structures (Spirtes 2001).

DAGs represent probability distributions through the graphical relation "d-separation" (Spirtes 2001), which is a relation among three disjoint sets of vertices in a directed graph. If a set of vertices *Z,* blocks all connections between a set of vertices *X* and a set of vertices *Y,* in a graph *G*, then *X* and *Y* are d-separated by *Z* in *G*. In the DAG on the left side of Figure 1, X_2 blocks the only directed path connecting X_1 and X_3, so X_1 and X_3 are d-separated by X_2. Applying d-separation to connect a DAG G to the probability distribution *P* means that if two sets of vertices *X* and *Y* are d-separated by a set of vertices *Z* in *G*, then *X* and *Y* are independent conditional on *Z* in the distribution *P*. In Figure 1 X_1 and X_3 are d-separated by X_2. Note, in all distributions the DAG can represent, X_1 is independent of X_3 conditional on X_2. The notation for independence is:

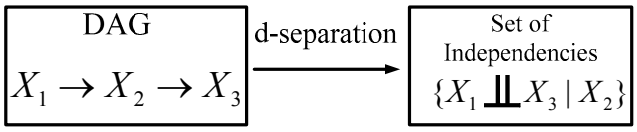

Figure 1: D-separation. (Schienes 2003).

$X_1 \perp\!\!\!\perp X_2 \mid X_3$ which means: X_1 and X_3 are independent conditional on X_2 (Dawid 1979).

Often, distinct DAGs can represent the same set of independence relations, and thus the same set of distributions. Inference algorithms have been developed to compute the DAGs that represent a given set of independence relations. The relationship between a set of independencies, inference algorithms and DAGs is shown in Figure 2. We will be using these inference algorithms to compute DAGs for a given set of independence relations in our enhanced version of EE.

Causal inference procedures have been used to determine causal relationships among system events. These procedures are tailored to assist users faced with forensic questions. Users range from forensics experts needing to analyze the origins of a trojan horse program ("Who was responsible for the creation of this file?") to the system administrator needing to analyze intrusions ("How did the intruder break in?"). These techniques have been shown to assist security experts required to answer questions about the cause-effect relationships between various events that occur in a computer system (Jeyaraman 2006).

Ranking and Selection procedures have been designed to select the best system from a number of alternatives, where the best system is defined by the given problem. Causal inference algorithms can be used to rank the factors or variables which affect a specified output. However, ranking and selection is a fundamentally different problem from revealing the interactions of identified abstractions within a model causing emergent behavior. Improvements in ranking and selection procedures will improve the ability of causal inference procedures to rank the influence of variables on a specified output.

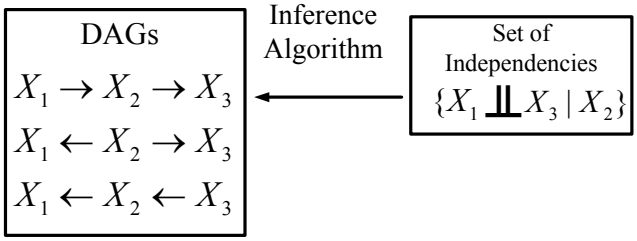

Figure 2: The relationship between sets of independencies and inference algorithms. (Schienes 2003).

3 EXPLANATION EXPLORATION AND CAUSAL INFERENCE

Previously, we have introduced "Explanation Exploration" (EE), for gathering insight into emergent behaviors using semi-automatic model adaptation (Gore 2007). EE can be improved by iteratively applying causal inference procedures to samples gathered from semi-automatic model adaptation, as we discuss here.

3.1 Explanation Exploration

EE is a method for increasing insight into unexpected – emergent– behaviors, and providing a path to validation of valid behaviors. EE incorporates semi-automated exploration of conditions in which a user can test hypotheses about emergent behaviors, and increase confidence about hypothesized meaning of the emergent behaviors. Model validation is a goal, but not necessarily an outcome of EE.

When testing a hypothesis about an emergent behavior, a SME may want to observe the emergent behavior under a specified set of target behaviors. However, when there are non-linearities in simulated behaviors, the SME may not know how to adapt the simulation to achieve the desired target behaviors directly.

We advocate the application of semi-automated model adaptation for efficient exploration of emergent behavior. When constructing a model, abstractions inevitably must be selected in order to reduce complexity, improve performance, or provide estimations for unknown information. We call those places where a SME can choose among abstractions *abstraction opportunities*. We have developed a language and supporting tools for a SME to identify abstraction opportunities and alternatives for model abstractions. The supporting tools allow alternatives for each model abstraction to be reflected in the simulation source code as possible alternate bindings (Carnahan 2006).

With alternate bindings present in the source code, a model adaptation strategy employing optimization becomes possible. EE uses optimization-based adaptation to test user hypotheses about an emergent behavior by efficiently creating user specified conditions of interest. A user gathers insight by observing an emergent behavior under the conditions of interest. If the observed behavior matches the user's hypothesis it passes the hypothesis test, otherwise it fails (Gore 2007).

As an example, we use the concept of *apparent wind* to elucidate differences between EE and its improved version employing causal inference. It is a little known fact that some sailing craft can attain a forward velocity that exceeds true wind speed. Such craft are capable of exploiting *apparent wind*, which is a combination of true wind speed and the craft's own forward velocity (Colgate 2001). Apparent wind creates emergent behavior.

Given an emergent behavior E, a user must establish if expectations regarding simulation behaviors need to be modified to include the emergent behavior. Alternatively the user may decide the emergent behavior E, is an error and not valid. EE facilitates this decision process. The user generally needs to formulate a hypothesis, H, about how emergent behavior E will be manifested under a condition of interest, C. The user must identify possible model abstraction alternatives to search to create C. The user can test hypothesis H by observing the emergent behavior E, under condition of interest C.

In the sailing example, C may be the condition "the sail is *full of wind*" –wind flow is smooth over both surfaces of the sail. As the boat accelerates the sail must be oriented increasingly towards the boat's forward direction. If the user can identify model abstraction opportunities and possible alternative bindings that can be explored to achieve sail fullness for a given set of conditions, then s/he can form and test the hypothesis H: the boat speed will be maximized when the sail is full of wind.

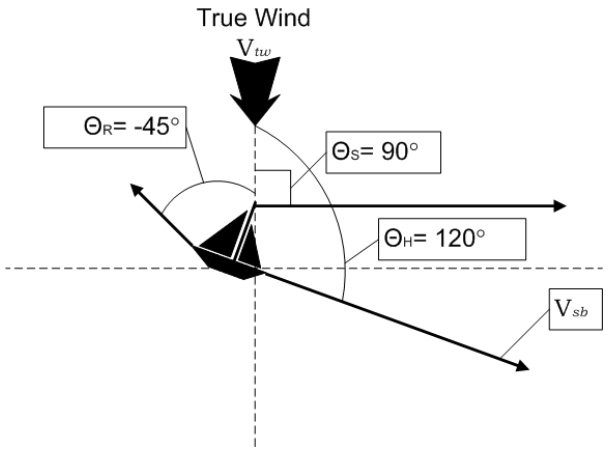

Figure 3: SME Hypothesis $H_{indirect}$

V_{sb}= velocity of the sailboat.

V_{tw}= velocity of the true wind.

Θ_R= angle between true wind and the sailboat rudder.

Θ_H= angle between true wind and the sailboat hull.

Θ_S = angle between true wind and the sailboat's sail.

The EE Process is applied as follows :

1. The SME identifies emergent behavior, E, which occurs when $V_{sb} > V_{tw}$.
2. The SME speculates E will exhibit behavior E_{ic} when condition C arises: "Boat speed will be maximized when the sail is full of wind." The SME speculates that Θ_R, Θ_H, and Θ_S are alternatives that can be searched to adapt the model to meet C. This hypothesis is H, which is shown in Figure 3. Θ_R, Θ_H, and Θ_S are chosen based on the SME's understanding of sailing and wind. If E is valid, it must possess the characteristics described in E_{ic}, when condition C arises.
3. The SME identifies model abstraction opportunities related to Θ_R, Θ_H, and Θ_S.
4. The SME identifies model abstraction alternatives for each model abstraction opportunity.
5. The developer incorporates the model abstraction alternatives into the simulation source code.
6. The SME believes E_{ic} will be observed when C is true, and that C will be true for some combination of the model abstraction alternatives.
7. The developer and SME use optimization and/or manual modification to discover when C is true.
8. The SME observes the simulation instance produced through model adaptation. The SME confirms or refutes H.

The SME gathers insight from the hypothesis test in Step 8. S/he repeats the EE process until s/he has gathered sufficient insight to validate or refute the emergent behavior, E, in the simulation.

The sailing example illustrates how EE is applied to emergent behavior occurring in a fictional model. Next we explore how EE is improved by adding causal inference.

3.2 Improving EE with Causal Inference Procedures

Causal inference procedures create an explanation for a given set of observations. They require an unbiased set of observations of the variables of interest and they produce a causal theory explaining the relationship of the variables. A causal theory includes a causal model and parameters which specify how each variable is influenced in the causal model. A causal model is a directed acyclic graph, with a 1-1 mapping between vertices in the graph and variables of interest. Variable *X* has a causal influence on variable *Y* if and only if a directed path exists from vertex *X* to vertex *Y* in the causal model. The model serves as the basis for the causal theory. Each edge in the causal model is a 1-1 mapping with a parameter associated with the causal theory. Each parameter specifies the strength of the causal influence (the probability that *X* has a causal influence on *Y*) induced by the corresponding edge (Pearl 2000).

We improve EE by iteratively applying causal inference procedures to samples gathered from semi-automatic model adaptation, Step 7 of the EE process. In this step the developer either chooses manual modification or optimization to adapt the model to meet the SME's specified condition of interest. Each time optimization is used multiple runs of the simulation with alternate abstraction bindings are performed. Simulation results are collected and treated as sample runs from the model space. We improve EE by applying causal inference procedures to this set of samples to reveal the interactions of the identified abstractions in the model which cause an emergent behavior of interest. Each condition of interest the user creates results in more samples of the model space. In general, a larger sample size is expected to create a more accurate causal theory.

Samples collected from the optimization process cannot be used immediately by causal inference procedures. Optimization samples regions in the model space with high probability of meeting the SME's condition of interest more frequently. This bias is expected: densely sampling certain regions of a search space makes optimization effective. Consequently, the samples are not uniform randomly distributed as required by causal inference procedures.

We use importance sampling to produce a uniform random sample from the biased optimization sample. Importance sampling supports estimation of the properties of a particular distribution, while only having samples generated from a different distribution (Srinivasan 2002). In order to create a sample with properties of a uniform random distribution we use the Uniform-Divide Algorithm and the Random Sampling Algorithm, which are described next.

3.2.1 Uniform-Divide Algorithm

The Uniform-Divide Algorithm (UDA) takes as input a biased set of samples *B*, and a search space *S*. It outputs a set *R* and an integer *N*. Each element of *R* is a bucket containing arbitrarily many unordered samples. *N* represents the number of buckets in *R*. UDA assumes the existence of the following functions:

- *divide(S, N)* - given a search space *S*, and an integer *N*, *divide(S,N)* outputs a set of buckets *R*. Each bucket in *R* represents a non-overlapping, *1/N* size area of search space *S*.
- *map(B,R)* - given a set of samples *B*, and a set of buckets *R* representing a search space, *map(B,R)* places each B_i into bucket R_j such that bucket R_j covers the search area containing sample B_i.
- *noEmptyBuckets(R)* – given a set of buckets *R* *noEmptyBuckets(R)* outputs true if and only if each bucket in *R* contains at least one element.

Algorithm 1 (Uniform-Divide Algorithm):
Input: B, S Output: R, N
Step 1: N := 0;
Q := NULL;
R := NULL;
Step 2: while noEmptyBuckets(Q) OR Q = NULL
R := Q;
N := N+1;
Q := divide(S,N);
map(B,Q);
end;

UDA discretizes the search space into the maximum number of buckets such that each bucket contains at least one sample, and each covers a uniform, non-overlapping area of the search space. Each bucket contains all of the samples taken from the area of the search space the bucket covers. *R* is organized so that each biased sample is weighted with the probability that the sample was produced from a uniform random distribution. We use the Random Sampling Algorithm to sample *R*, producing *N* uniform random samples from the search space.

3.2.2 The Random Sampling Algorithm

The Random Sampling Algorithm (RSA) takes as input a set of buckets *R* where each bucket contains a set of samples, and an integer *N* which represents the number of buckets in *R*. RSA outputs a set *O* containing *N* random samples from the search space covered by the input set *R*. RSA assumes the existence of the following functions:

- *rand(I,N)* - given two integers *I* and *N*, where $I \leq N$, *rand(I,N)* outputs a random integer between *I* and *N* inclusive, with each integer between *I* and *N* inclusive being equally likely.
- *sample(*R_k*)* – given a bucket R_k, *sample(*R_k*)* returns one element from the bucket chosen randomly with all elements in the bucket being equally likely.

Algorithm 2 (Random Sampling Algorithm):
Input: R, N Output: O
Step 1: O := NULL;
I := 1;
Step 2: while $I \leq N$
K := rand(1,N);
add sample(R_k*) to O;*
I = I+1;
end;

When used immediately after UDA, RSA outputs *N* uniform random samples from the search space which are suitable for use with causal inference procedures. Use of these algorithms to produce a uniform random sample is straight forward, but an important capability of improved EE. Any optimization process can be used to adapt a model to meet new conditions of interest (Step 7 of the EE process) and improved EE will produce causal theories without any change to UDA or RSA.

3.2.3 The Sailing Example and Improved EE

To elucidate how causal inference procedures are applied to and improve EE we revisit the sailing example. Recall user identified emergent behavior E: the sailboat is traveling faster than the speed of wind, $V_{sb} > V_{tw}$. The user is interested in creating condition of interest C: the sail is *full of wind*. The user wants to test the hypothesis H: boat speed, V_{sb}, will be maximized when the sail is full of wind. The improved EE process is applied as follows:

1-6 remain the same.
7. The developer and SME use optimization to discover cases where C is true.
8. If optimization is used the developer records the samples from the model space explored by the optimization process in set *B*. Each of these samples is a 5-tuple: (Θ_R, Θ_H, Θ_S, *SailFullness*, V_{sb}). The developer also records the dimensions of the model space searched in data structure *S*.
9. The developer applies UDA to *B,S*. UDA outputs *R* and *N*, the set of filled buckets covering the search space *S* and the number of buckets.
10. The developer applies RSA to *R, N*. RSA outputs *N* uniform random samples in set *O*.
11. The developer applies the causal inference procedure *P*, to set *O*. *P* outputs causal theory *T*.
12. The SME views and queries causal theory *T* for insight into the interactions of abstraction opportunities within the model that cause E. The causal theory, *T*, for the sailing example is shown in Figure 4. *T* shows that changes in Θ_R, Θ_H, and Θ_S cause the sail to be full of wind, which in turn causes the sailboat to change velocity.
13. The SME observes the simulation instance produced through model adaptation. The SME confirms or refutes H.

The SME gathers insight from the causal theory in Step 12 and the hypothesis test in Step 13. S/he repeats the improved EE process until she has gathered sufficient insight to validate or refute the emergent behavior, E, in the simulation. Each time the SME repeats the EE process more samples from the model space are recorded and the causal theory is expected to become more accurate.

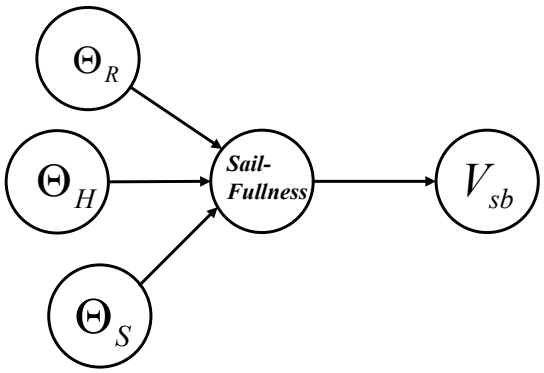

Figure 4: Causal Theory *T* for the sailing example.

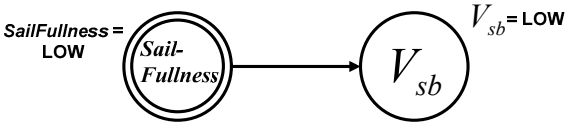

Figure 5: Intervening on *SailFullness*. The double circle denotes where the user specified change occurs.

3.3 New Capabilities of Improved EE

Improved EE creates new opportunities. Through its use of causal inference procedures it produces a causal theory that reveals the interactions of abstraction opportunities that cause emergent behavior. The causal theory supports ordering of abstraction opportunities based on their effect on emergent behavior. The theory can be queried as well to predict user specified interventions to the model. Each causal theory provides explicit formulas for estimating how abstraction bindings will be affected by changes in others (Spirtes 2001). For example, consider setting *SailFullness* in Figure 4 to "low". The result is shown in Figure 5. By setting *SailFullness* "low" the effects of Θ_R, Θ_H, and Θ_S are screened off. The remaining cause, *SailFullness* = LOW causes sailboat velocity, V_{sb}, to also be low.

The ability to predict the results of interventions within given confidence intervals provides a complimentary exploration capability to model adaptation. Predicting interventions offers users a lightweight, probabilistic method to explore how changes to one abstraction opportunity affect others in the model. However, predicting intervention effects is not as powerful as model adaptation because predicting interventions cannot be used to test user hypotheses for conditions of interest.

4 IMPROVED EE AND EPIDEMIOLOGY

To evaluate improved EE we conducted a study using an agent based epidemic model (Dunham 2005). We compare our results with results from an established differential equation model used to model SEIR diseases (Li 1995).

4.1 Epidemic Model Overview

Epidemics have been modeled mathematically for over a century. The well established SEIR model of infectious disease spread is described by the following system of differential equations where p, q, γ, ε, λ and u are positive parameters and S, E, I, and R denote the fractions of the population that are susceptible, exposed, infectious, and removed, respectively. Individuals are susceptible, then exposed (in the latent period), then infectious, then removed from the studied population (Li 1995).

$$\begin{aligned} S' &= -\lambda I^p S^q + u - uS \\ E' &= \lambda I^p S^q - (\varepsilon + u)E \\ I' &= \varepsilon E - (\gamma + u)I \\ R' &= \gamma I - uR \end{aligned} \qquad (1)$$

We compare the results of the established differential equation model with Dunham's (Dunham 2005) agent based SEIR epidemic model (ABM). The Dunham model predicts disease spread by modeling interactions on a 2-D torus. Infectious individuals search for susceptible individuals within a specified radius and spread their infection with a given probability. Dunham claims, "the curves (susceptible, exposed, infected, removed) created are a qualitative match to real-world epidemic data. With proper parameterization, this model could be used for realistic simulations" (Dunham 2005). Figures 6 and 7 show results for Dunham's ABM and the differential equation model for an SEIR epidemic for 100 days for a population of 100. Results match closely: standard deviations of corresponding curves differ by less than 2% of the population size.

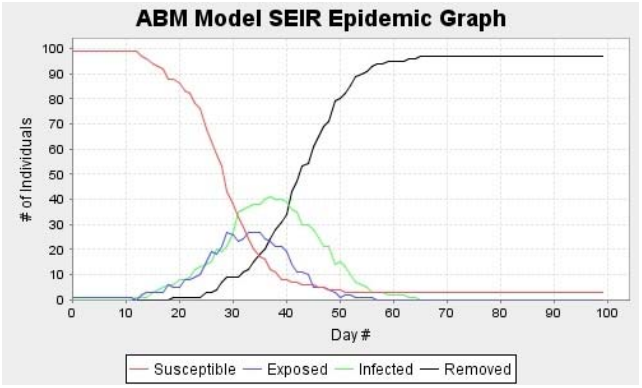

Figure 6: Dunham's ABM with the author's suggested parameterization for a population of 100 over 100 days.

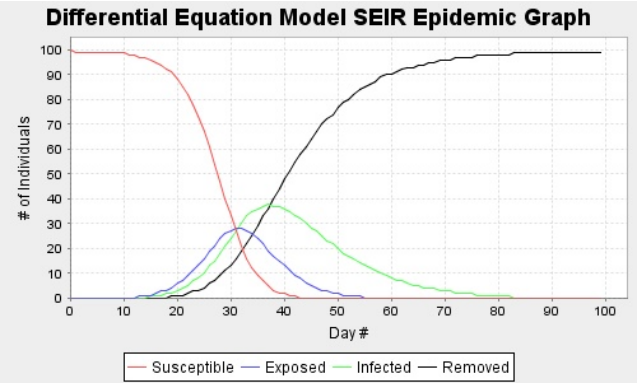

Figure 7: Differential equation model with new infection rate = 8.0 for a population of 100 over 100 days.

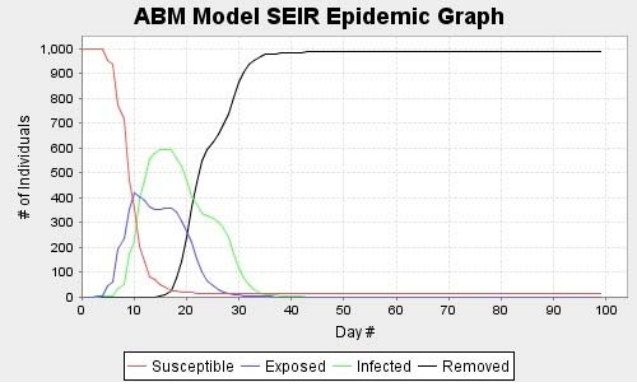

Figure 8: Dunham's ABM with the author's suggested parameterization for a population of 1,000 over 100 days.

Dunham's ABM appears to produce valid predictions. Differences between it and accepted data or models should be investigated to determine the ABM's general validity.

4.2 Emergent Behavior within the Epidemic Model

We have observed an unexpected –emergent– behavior in Dunham's ABM for a population size of 1,000. By only changing the population parameter from size 100 in Figure 6 and Figure 7 to size 1,000 significantly different predictions for disease spread emerge. As shown in Figure 8 Dunham's ABM predicts a shorter, heightened infectious period where no infected individuals remain after day 35. In contrast, the differential equation model shown in Figure 9 predicts a longer infectious period with infectious individuals still present at day 80. The standard deviation of the ABM's curves from the accepted differential equation model's curves are (S) 173.2 individuals or 17.32% of the population, (E) 132.6 individuals, 13.26%, (I) 181.6 individuals, 18.16%, and (R) 110.6 individuals, 11.06%. Standard deviations, compared to those of the population of size 100 are an order of magnitude greater. Based on examples in (Dunham 2005) and Figures 6 and 7 we expected the ABM to predict results similar to the differential equation based model. The differences represent an emergent behavior.

4.3 Improved EE and New Infection Rate

To understand why Dunham's ABM predictions differ so significantly for a population of size 1,000, and to determine if the model is valid we applied improved EE. Epidemic models often include parameter(s) which represent the rate of new infections. In Equation 1 this rate is governed by the term $\lambda I^p S^q$. New infections occur in Dunham's ABM over time but there is no input parameter for infection rate, and we found none published.

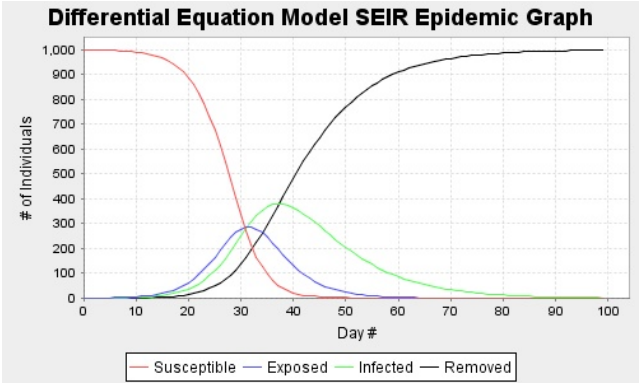

Figure 9: Differential equation based model with new infection rate = 8.0 for a population of 1,000 over 100 days.

We instrumented Dunham's ABM to capture the rate of new infections and used improved EE to create the condition of interest C: the rate of new infections is 8.0 per infected individual. Using C as a target behavior, we tested our hypothesis H: when the rate of new infections is 8.0 in both models, their predictions will be similar. Also, we used improved EE to produce a causal theory describing the abstraction opportunities that causally influence the rate of new infections in Dunham's ABM. The insight gained from observing Dunham's ABM under the condition of interest C, and the insight provided by the causal theory helped us determine if Dunham's ABM is valid for a population of size 1,000.

We adapted Dunham's ABM by searching alternative bindings for six different abstraction opportunities to create the condition of interest C: infection rate = 8.0. For each model run searched using improved EE we recorded:

- the explored alternative bindings for the six model abstraction opportunities:
  - the width, W, of the 2D torus
  - the height, H, of the 2D torus
  - the probability, B, an infected individual stays at home and does not travel to work.

  - the probability, X, an infected individual pushes the infection onto a susceptible individual within the specified radius.
  - the mean number of days, $\overline{EI}$, an individual remains in the exposed state (E) before transitioning to the infected state (I).
  - the mean number of days, $\overline{IR}$, an individual remains in the infected state (I) before transitioning to the removed state (R)
- the rate of new infection, ρ
- the number of individuals susceptible at day 25, S
- the number of individuals exposed at day 25, E
- the number of individuals infected at day 25, I
- the number of individuals removed at day 25, R
- the median standard deviation of the four agents based curves, σ

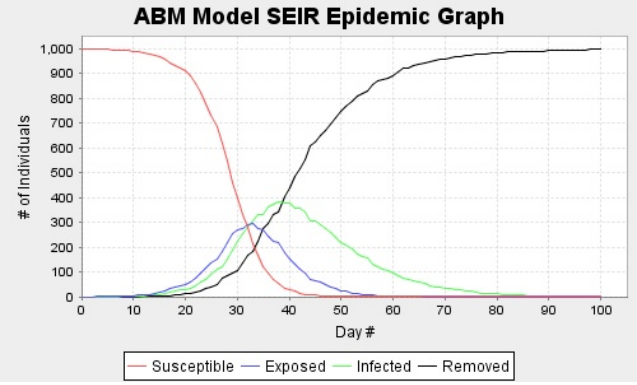

Figure 10: Dunham's ABM for condition C: new infection rate = 8.0 for a population of 1,000 over 100 days.

When the condition of interest, C, was achieved where the two models' infection rates where 8.0, Dunham's ABM predictions closely match those of the differential equation based model. This is shown in Figure 10. The standard deviation of Dunham's ABM curves from the accepted differential equation model's curves are (S) 18.9 individuals or 1.89% of the population, (E) 11.6 individuals, 1.16%, (I) 12.4 individuals, 1.24%, and (R) 12.7 individuals, 1.27%. Our hypothesis is correct: when the rate of new infections is 8.0 in both models their predictions are similar. Thus we gain confidence that Dunham's ABM can be used to correctly predict the spread of epidemic SEIR diseases. However, we still do not understand how the rate of new infections is controlled in the agent based model. Without fully understanding how the interactions of abstraction opportunities cause the rate of new infections to change, we cannot explain the emergent behavior of Dunham's ABM.

The causal theory produced by improved EE enables us to understand how the rate of new infections is controlled in Dunham's ABM. The causal theory for the model adaptation performed to create the condition of interest C is shown in Figure 11. W and H represent the width and height respectively of the 2D torus on which individuals interact. X represents the probability an infected individual infects another individual within the specified radius. S, E, I, and R represent the number of susceptible, exposed, infected, and removed individuals at day 25. ρ is the rate of new infections per infected individual and σ is the median standard deviation of the four curves of Dunham's ABM from the curves of the differential equation based model.

The causal theory reveals that the width and height of the 2D torus, along with the probability of infection have a causal influence on the infection rate. Also, the infection rate has a causal influence on the median standard deviation of the four curves in Dunham's ABM. These insights help us understand how the rate of new infections is controlled in the agent based model.

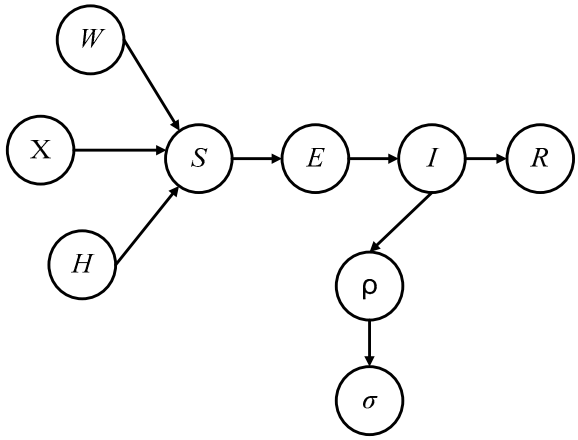

Figure 11: The causal theory output by Improved EE for the agent based SEIR epidemic simulation (Dunham 2005).

Increasing the width and height of the 2D torus causes the population to become less dense, ultimately causing the rate of new infections to decrease. A decrease in the probability that an infected individual infects a susceptible individual within the specified radius also causes a decrease in the rate of new infection. These interactions make sense. The significantly different prediction of Dunham's ABM in Figure 8 is due to a too-high rate of new infections per individual. If the size of the torus is increased to maintain a constant population density as population is increased then Dunham's ABM does track the predictions of the differential equation model. A more dense torus increased the number of exposed and infected individuals causing an increase in the rate of new infections per individual. This explanation is a product of applying the causal theory portion of improved EE, revealing how abstraction opportunities cause the emergent behavior in the model.

5 CONCLUSIONS AND FUTURE WORK

Computational power has reached a state where we can model and explore research topics as complex as the

spread of epidemic diseases individual by individual. However, methods for analyzing and testing hypotheses about emergent behaviors that result when developing such complex models are lacking. The disparity between our ability to build complex models and our ability to understand emergent behaviors exhibited by these models adversely affects the advancement of scientific understanding.

Our goal is to design and develop an effective approach to support user validation or rejection of unexpected model behaviors efficiently, and with confidence. We have improved the capabilities of our previously published approach, EE, by offering SMEs additional insight into the interactions of identified abstractions causing emergent behavior in a model. This insight is achieved by iteratively applying causal inference procedures to samples from different runs of the model. In our future work we expect to improve our causal insight by correctly combining causal analysis data (probabilistic) with static analysis data (conservative). Furthermore, using uncertainty representation we plan to extend EE to allow users to observe two new characteristics of the emergent behavior at conditions of interest: 1) the likelihood of the given output for the model and 2) the range of possible outputs.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the National Science Foundation (ITR 0426971), as well as from our colleagues in the Modeling and Simulation Technology Research Initiative (MaSTRI) at the University of Virginia.

REFERENCES

Balaci, O. 1997. Principles of simulation model validation, verification, and testing. *Transactions of the Society for Computer Simulation International,* 14:3-12.

Carnahan, J. C. 2006. Language Support for the Coercible Software Domain, *A Dissertation Proposal*, University of Virginia: School of Engineering and Applied Science, Charlottesville, VA.

Colgate, S. *Fundamentals of Sailing, Cruising, and Racing*, W.W. Norton & Company, New York City, NY, 1996.

Davis, P. K. 2005. New Paradigms and Challenges. *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong and J. A. Joines, 1067-1076. Piscataway, NJ: Institute of Electrical and Electronic Engineering.

Dawid, A. 1979. Conditional independence in statistical theory (with discussion). *Journal of the Royal Statistical Society,* 41:1-31.

Dunham, J. 2005. An Agent-Based Spatially Explicit Epidemiological Model in MASON. *Journal of Artificial Societies and Social Simulation.* 9:1-3.

Eisenbarth, T., R. Koschke, and D. Simon. 2001. Aiding program comprehension by static and dynamic feature analysis. *Proceedings of the 17th International Conference on Software Maintenance*, 602–611. IEEE Computer Society, Washington, D.C.

Elderd, B.D., V. M. Dukic, and G. Dwyer. 2006. Uncertainty in predictions of disease spread and public health responses to bioterrorism and emerging diseases. *Proceedings of the National Academy of Sciences*. 103:42: 15693-15697.

Gore, R., P. F. Reynolds, L. Tang, and D. Brogan. 2007. Explanation Exploration: Exploring Emergent Behavior. *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation*, 113-122. San Diego, CA: Society for Modeling and Simulation International.

Groce, A. 2004. Error explanation with distance metrics. *Tools and Algorithms for the Construction and Analysis of Systems*, 108-122.

Jeyaraman, S and M. J. Atallah. 2006. An empirical study of automatic event reconstruction systems. *Digital Investigation,* 3:108-115.

Johnson, C.W. 2005. What are emergent properties and how do they affect the engineering of complex systems? *Reliability Engineering and System Safety,* 91: 1475-1481.

Li, M. Y. and J. Muldowney. 1995. Global stability for the SEIRmodel in epidemiology, *Mathematical Biosciences* , 125(2):155-164.

Liu, X., P. F. Reynolds, and D. Brogan. 2006., Using abstraction verification of simulation coercion., *Proceedings of the 2006 Conference on Principles of Advanced and Distributed Simulation*, 119-128. IEEE Computer Society, Los Alamitos, CA.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*, Cambridge Univ. Press.

Schienes, R. 2003. AnIntroduction to Causal Inference. Available via <http://www.cs.cmu.edu/~tom/10-702/causal1.pdf> [accessed March 27, 2008]

Spirtes, P., C. Glymour, and R. Scheines. 2001. *Causation, Prediction, and Search*, Springer-Verlag, New York.

Srinivasan, R. 2002. *Importance Sampling: Applications in Communication and Detection*, Springer-Verlag, New York.

Tip, F. 1995. A survey of program slicing techniques. *Journal of Programming Languages*, 3:3:121-189.

Zeigler, B. P., T. G. Kim and H. Praehofer. 2000. *Theory of Modeling and Simulation*. 2nd Edition. Burlington: Academic Press

Zeller, A. 2002. Isolating Cause-Effect Chains from Computer Programs. *Proceedings of ACM SIGSOFT 10th International Symposium on the Foundations of*

Software Engineering (FSE-10), 1-10. Charleston, South Carolina.

AUTHOR BIOGRAPHIES

ROSS GORE is a Ph.D. Candidate in Computer Science and a member of MaSTRI at the University of Virginia. His email address is <rjg7v@virginia.edu>.

PAUL F. REYNOLDS, JR. is a Professor of Computer Science and member of MaSTRI at the University of Virginia. He has conducted research in Modeling and Simulation for over 25 years and has published on a variety of M&S topics. His email address is <reynolds@virginia.edu>.