Digital Caricature: Stochastic M&S of Complex System of Systems

Ronald Deiotte, Jarrid Carroll-Frey

ISSAC LLC

Colorado Springs, CO
ron.deiotte@issaccorp.com, jarrid.carroll-frey@issaccorp.com

ABSTRACT

Physics-based algorithms are the gold standard for Modeling and Simulation (M&S) of a System of Systems (SoS); however, high-fidelity physics M&Ss can be cost prohibitive, slow, and/or exceedingly complex. Further, physics-based algorithms are often not considered useful until they pass a threshold of "realness," this adds to cost and delays the readiness of M&S. Given the costs, development delays, and run speed of physics-based M&S they are often used sub-optimally, introduced too late to be used to aid design, or simply not implemented at all.

Stochastic algorithms are inexpensive as the same algorithm can be used for each component in an SoS. These algorithms use a random number generator making them faster and less hardware intensive than physics-based algorithms. Finally, the parameters used in a stochastic model or simulation can be changed and refined over time, while still providing actionable insights early on in the design process. However, one drawback of stochastic models is that they do not have the fidelity of their physics-based counterparts. With this understanding, we assert that the M&S environment could greatly benefit from more extensive use of stochastic M&S. This is particularly true for aiding design, as fault identification, and as a guide for better utilization of higher-fidelity M&S tools.

This paper describes the use and benefits of stochastic M&S for SoSs. We compare the capabilities of stochastic M&S to physics-based M&S using a simplified threat/intercept scenario. We explore how stochastic simulations can aid in optimization of physics-based M&S by constraining the input space for physics-based M&S. We introduce the concept of a digital caricature (a stochastic digital twin) and how this can be used in conjunction with real-world data and existing digital twins. Finally, we describe other avenues of interest for stochastic M&S to aid design, training, and operations of real-world systems.

ABOUT THE AUTHORS

Ronald Deiotte is the chief scientist at ISSAC. Ronald has been developing data analytics, data management systems, and novel databases for flexible application to M&S, data analysis, and model-based systems engineering with ISSAC for 15 years. In that time, he has worked on projects that span from missile defense to cancer research to overland shipping logistics. This multi-disciplinary environment has led him to several new approaches and unique solutions to problems that span all data environments.

Jarrid Carroll-Frey is a data scientist/software developer at ISSAC. He holds a bachelor's degree in mathematics and a master's degree in applied mathematics. He has primarily worked on the development of data conversion software focused on converting JSON, CSV, and PDF files into a graph database to expose hidden relationships in the data that may not be immediately obvious.

Digital Caricature: Stochastic M&S of Complex System of Systems

Ronald Deiotte, Jarrid Carroll-Frey

ISSAC LLC

Colorado Springs, CO
ron.deiotte@issaccorp.com, jarrid.carroll-frey@issaccorp.com

INTRODUCTION

In this paper, we are focused on the application of stochastic models and their use in simulations. This approach is distinct from randomly generating input for simulations (i.e. Monte Carlo simulation) in that the models themselves are probabilistic rather than deterministic (Rubinstein & Kroese, 2011). This also differs from propagation of errors into a probabilistic cloud based on physics simulations (Mooney & Swift, 1999).

A stochastic model is a model in which an event is represented by a Probability Distribution Function (PDF) (Pinsky & Karlin, 2010). The exact definition of this PDF is determined by input into the model. The outcome of the event being modeled in a simulation is then determined by a random draw on the PDF. The expectation and behaviors of different events can be modeled and refined in the PDF to reflect real world (or physics-based) outcomes.

By converting a problem space from physics-based to event-based and then modeling those events using PDFs, we decouple the simulation from absolute time steps and reduce the complexity of calculations. This drastically reduces the necessary compute power for the simulations. Moreover, combining reduced compute power with a parallel approach to running the simulations allows for substantially higher run frequency and a better understanding of the possible outcomes of the problem space. It has been noted that some of the problems with physics-based M&S has led to abstraction of such models and the divergence of digital twins from their real-world counterparts (Skinner, 2023). For this reason, we propose stochastic event-based models as an alternative to low resolution, abstract, physics-based digital twins. This type of model could be seen as a digital caricature as it is not a perfect representation of its real-world counterpart but is recognizable and has the prominent features of the real-world counterpart.

Stochastic models are not intended as a replacement for physics-based models but as an overlooked tool that can aid in design and development of new systems, discovery of the best practices/use of a system, optimizing the effective use of physics-based high-fidelity M&S, and as a training aid. Further, stochastic M&S is not intended as an assessment of capability but as a survey of what is possible.

EXPERIMENTAL SET-UP

For this experiment, the intent is to compare a stochastic M&S to a physics-based M&S. In the past, ISSAC has developed a stochastic M&S to do large system of systems analysis with the Missile Defense Agency (MDA). For obvious reasons, we cannot use that data for this publication. Further, we do not have use rights to the physics-based models and simulators used in that work. For this effort, we developed a simple two-dimensional scenario that can be injected into a simplified two-dimensional physics-like simulator and into a stochastic simulator. In this way, we are able to do an apples-to-apples comparison between the two approaches and properly assess the costs and benefits of both approaches and better understand where each approach is best used in the life cycle of a system-of-systems.

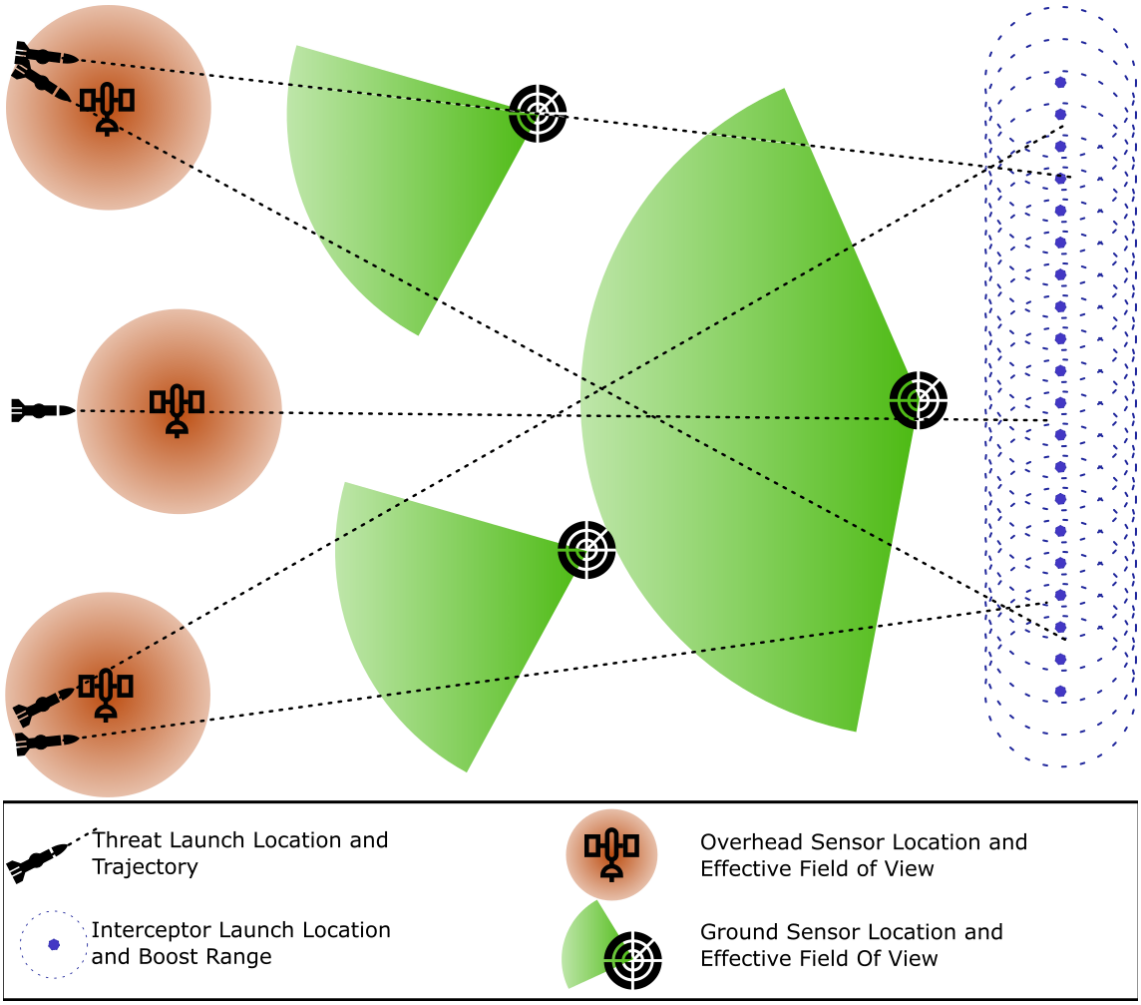

Figure 1: Test scenario layout.

For the scenario we developed for this effort, the red force consists of 5 simple threat models that accelerate from a launch location to a max speed at which point they "coast" in a straight line to their target. The threats launch at times $t = 0, 1000, 100, 1001, 200$ respectively, from the top left corner to the bottom left corner of the layout. The blue force consists of two different types of sensors – 3 overhead and 3 ground –, a command-and-control node (not pictured), and 20 interceptors that only travel in straight lines. These simplified physics components allowed us to develop simple physics M&S to compare to our event-based stochastic M&S.

Threat Model

Both simulations use the same threat models. These models are physics based and generated using physics-based modeling independent of the simulation. Modeling the threats separate from the simulation is true to industry practices and it gives us a simple format upon which to base both simulations. The threat model consists of a simple table with location, shape, and boost values for each simulation clock tick from the launch to impact.

The two different simulations handle the threat model differently when running. In the physics simulation, the simulation steps through each tick presenting the instance of the threat at that tick to each sensor model. In the stochastic simulation, the threat model is used to generate a sequence of events, each with a PDF based on the features of the threat.

Physics Based Implementation

In the physics simulation, there are three main model types: sensors, interceptors, and Command-and-Control (C2). The simulation runner presents the actual location and shape of each threat to the sensor and interceptor models at each clock tick of the simulation. The models then calculate a view and, if they can, a track for the threat and use a message bus to pass the information to the C2 model. The view generated by sensor and interceptors contains introduced errors to replicate resolution limitation and real-world environments. The C2 model predicts a future state for each track based on all the reports about that track and generates a plan for cueing future sensors and deploying interceptors. The sensor cue and interceptor deployments are transmitted to the respective models using the messaging bus.

Sensor Models

In the test scenario, there are 2 types of sensors: overhead and ground. Overhead sensors use the boost value to identify and generate a view and track of a threat. This means that if the boost value is below the detection threshold, then the overhead sensor does not see it. Overhead sensors view the whole area of their field-of-view at each tick. Overhead sensors send tracks to the C2 model using the message bus but does not accept any tasking or mode changes.

In contrast, the ground sensor uses the shape to generate a view and track of a threat. In this case, the sensor determines if the relative perspective size of the threat is above the resolution threshold of the sensor. The perspective size of the threat also determines the amount of error added by the sensor to the final generated view and track. The field-of-view of each ground sensor is partitioned into resolution areas. The sensor can only resolve threats that are in the currently active resolution area and can only have one active resolution area for each tick. The way the sensor transitions between active resolution areas depends on the mode the sensor is in. If the sensor is in “scan” mode, then it visits each resolution area in a minimal number of ticks. If the sensor is in “track” mode, then the sensor has a list of resolution areas that it is visiting. The track list is based on cues from C2 and from the sensor’s own predictions of existing tracks.

Interceptor Models

The interceptor is modeled only after it has entered its coast phase. Interceptors have a boost range, which indicates the distance away from the launch location the interceptor must travel before it coasts and starts looking for a threat to intercept, the boost time is the time it takes for the interceptor to get from the launch location to the boost range. Constrained to the boost range and boost time, the C2 creates an intercept plan that tells an interceptor when and where it should enter that coast phase. At that time, the interceptor model starts acting on the threat information that is passed to it from the simulation. The interceptor acts similarly to a ground sensor, but it does not communicate with C2 at this point and only has one resolution area. If the threat is within one tick of the interceptor and the threat interceptor has a view of the threat, then the interceptor is considered to have hit and destroyed the threat.

C2 Model

The C2 model collects tracks from the different sensors to generate a physics model of each threat. This model consists of a prediction of the current location of the threat and the velocity of the threat. The C2 model uses this information to generate cues for the ground sensors and to develop an intercept plan. On the appropriate tick, the C2 model sends these messages on the appropriate channel of the message bus.

Parallel Runs in the Physics Simulation

In the physics simulation, there is only so much parallelization we can do. This is because each track and prediction of the C2 model is based on the cumulative knowledge gathered by all the models throughout the simulation. The accuracy of C2’s prediction leads to putting the interceptors in the right position to intercept the threat and we must

step through each tick of the simulation to determine if the threat and the interceptor hit. This means that, each time we run the simulation, we must fully execute each tick of the entire simulation in order. The only parallelization that can be gained is by running each model independently (either asynchronous, on separate threads/cores, or on separate hardware). However, the simulation must still pass each tick to each model and the models will synchronize at some points due to the messages that are passed between models.

If we want to run a statistically significant number of runs of a single scenario (to completely account for intrinsic error), then we would have to run many instances of the simulation where each instance must iterate through the thousands of ticks of the simulation clock.

Stochastic Based Implementation

For the stochastic simulation, we must first identify the meaningful events that are to be simulated. This can be done at several different levels of abstraction and is similar to the concepts used to build Activity Diagrams and/or Sequence Diagrams from Model Based Systems Engineering (MBSE) (OMG Systems Modeling Language (OMG SysML™) Version 1.6, 2019) (Jamshidi, 2008). Figure 2 shows the high-level sequence diagram for a single threat in the SoS.

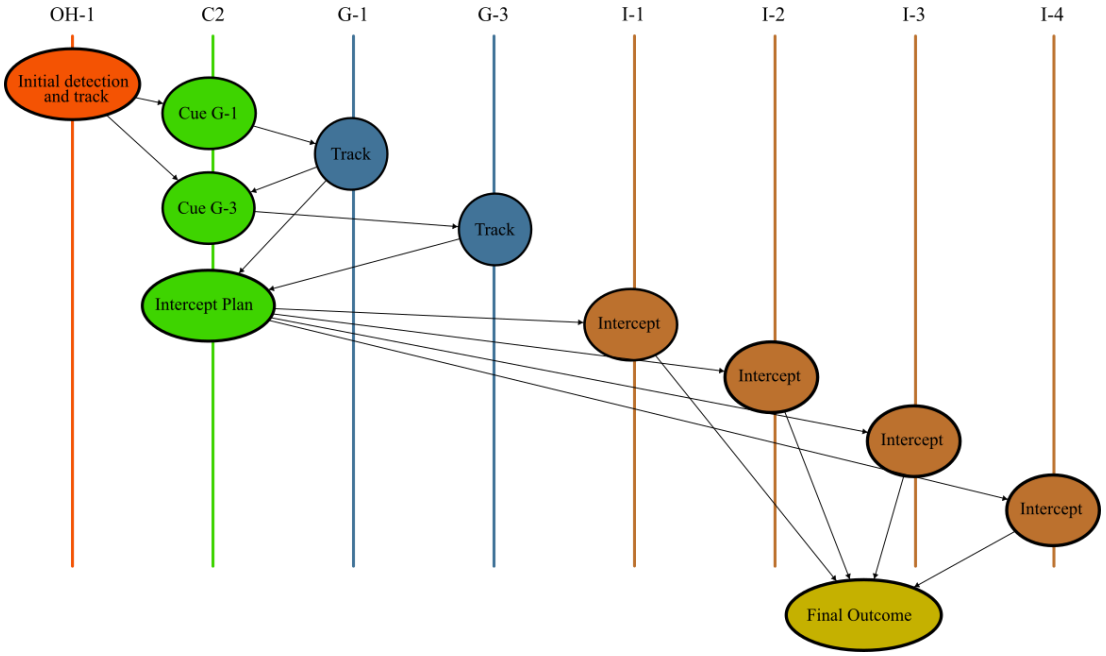

Figure 2: A simplified Sequence Diagram for a single threat in the system of systems

For this investigation, we are interested in the basic behavior of the system of systems. To this end, we will ignore all internal events that occur within individual systems and condense all the interactions between the sensors, command-and-control, and the interceptors into single interactions.

We start with the simulation parameters, which consist of a list of threats and a blue force description, which consists of an overhead sensor, ground-based sensor, the command-and-control, and interceptor parameters as described at the beginning of this section. Using these, we create an event stack for each threat. Using these event stacks, we can then run a simulation that combines all events in order. Each event in the event stack is a stochastic model of the probability of the event being successfully executed. This means that each event model does the same thing – sets up a PDF based on some parameters and draws from that PDF – and we do not need to make different models for each event.

The ARGUS Distribution

The first step in building such a stochastic model is to decide on a PDF to use. For consistency and simplicity of implementation, we want the same PDF for each event that we shift based upon several factors, such as the previous probability, number of threats present, number of interceptors left, etc. For this experiment, we used the ARGUS distribution, a probability distribution that originated with the ARGUS particle physics experiment which ran from 1982 until 1992. The ARGUS distribution is the probability distribution of the invariant mass of a decayed particle in a continuum background (Albrecht, et al., 1990). The ARGUS distribution depends upon a single parameter, χ, which controls the curvature of the distribution. See Figure 3 for a graph of the ARGUS PDF for different values of χ.

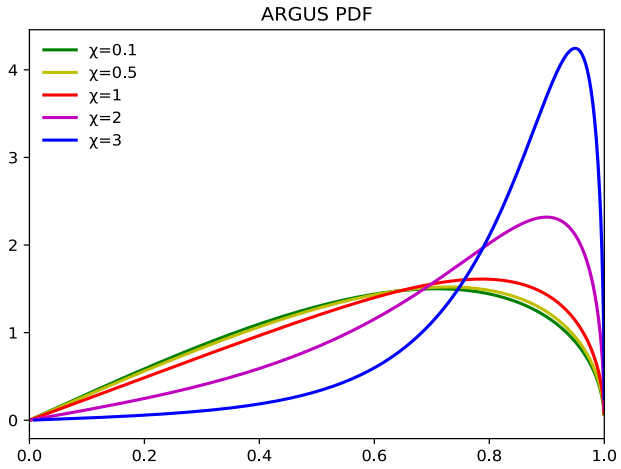

Figure 3: Argus PDF for a variety of χ values. By selecting a χ value based on the geometry of the threat and the real-world sensor or interceptor, we can tailor each event model for each threat.

The ARGUS distribution was chosen for this experiment for several reasons. The most important is the fact that the ARGUS PDF is always skewed toward 1, this reflects the fact that each event is designed to succeed and has fewer failure states than success states. The other main reason is that the ARGUS PDF only requires one input value, χ. This simplifies the creation of the PDF because we can account for all the potential factors with a single equation rather than several for a PDF that requires multiple inputs. Any PDF could be used to model the likelihood of an event being successful. The selection of the PDF depends on the information available to influence the shape of that PDF. In this case, we are interested in building the PDF from the geometry of the scenario.

Calculating χ for Event Models

In general, we will build the event stack by first doing a pairwise comparison between sensors and threats. Using these pairwise comparisons, it is possible to map the geometry of the scenario to a list of characteristics of particular events, such as: the amount of time that the threat was in view of the sensor, the average angle of resolution of the threat with respect to the sensor, the variance of the angle of resolution, etc. This way, we keep track of all the important information for each sensor event. After all these events have been created, we then order them by the first tick in which the sensor would, theoretically, detect the threat. Finally, we add the intercept plan event and interceptor events using the probabilities from all previous sensor events.

For the overhead sensor events, we know that the success of a track is primarily based on the amount of time a threat is boosting in the field of view of the sensor (Δt), because this time can be quite large, we want to scale it down for the calculation of χ. The second criteria for a successful overhead track are the number of threats visible to the sensor when that threat is in view (n). So, we get the following function for χ.

$$\chi_{OH}(\Delta t, n) = \frac{1}{n^2} + 6 \cdot \frac{\Delta t^2}{\Delta t^2 + 300} \quad (1)$$

The ground-based sensors' track probabilities are by far the most complicated. The probability of a success is directly proportional to the amount of time the threat is in the field of view (Δt), how successful the cue event was that initiated

the track (i), the average angular size of the threat from the perspective of the sensor ($\bar{\varphi}$), and the variability (standard deviation) of the angular size of the threat (σ_φ). The probability of success is inversely proportional to the number of threats being tracked by the sensor (n). The $\bar{\varphi}$ accounts for the range of the threat from the sensor and σ_φ accounts for how the angular size changed due to the rotation of the threat and the change of range of the threat.

$$\chi_G(\Delta t, i, \bar{\varphi}, \sigma_\varphi, n) = \frac{1}{n} + \frac{(2.3 \cdot \Delta t^2)}{\Delta t^2 + 300} + i + 100 \cdot \overline{\varphi} + \frac{\left(\sigma_\varphi^{.125} - 1\right)}{\sigma_\varphi^2 - 1} \quad (2)$$

The event successes for C2 are a little more abstract but have to do primarily with the number of threats being handled (n), and the success of prior events (i) (like initial detection, or track). We also introduce a reliability factor, (f). In this case, the C2 is highly reliable so we randomly select f such that $4.75 \leq f \leq 5$.

$$\chi_{C2}(n, i, f) = \frac{1}{n} + i + f \quad (3)$$

Finally, the event success for the interceptor is purely dependent on the success of the intercept plan (i), and a reliability factor (f). However, if there are no interceptors left the success probability drops to 0 so the PDF becomes a delta function on 0. For the intercept event, the reliability factor accounts for hardware reliability and environmental factors like weather. This means that this value can range widely so we randomly select the value of f such that $0 \leq f \leq 5$.

$$\chi_I(i, f) = i + f \quad (4)$$

Building the Event Stack

In general, our event stack will consist of sensor events, C2 events, an intercept plan event, and interceptor events (Figure 4). Each event will have associated with it a function that returns χ. Each χ function will depend upon a variety of different variables depending upon the event at hand but will almost always include the output(s) of the previous event(s). This means that the probability of any individual event is dependent on both the parameters of the scenario and the likelihood that previous events were successful. This χ value will then be used to define the PDF, from which we will take a random draw to feed into the next event.

We will define a sensor event as the interaction between a particular threat and a particular sensor, so it is useful to think of an event as a sensor-threat pair. A C2 event will accompany each sensor event since each sensor will need to cue C2 after observing the threat. Note that, for the overhead sensor events, we will combine the probabilities for each into a single probability to pass to command-and-control, as we are assuming that they are independent of each other. The intercept plan event will take draws from each sensor event and combine them into a single probability for input into its χ function. The interceptor events will take as input a draw from the interceptor plan event, after which, a final draw is done to get the overall probability of a successful interception. See Figure 4 for an illustration of an example event stack.

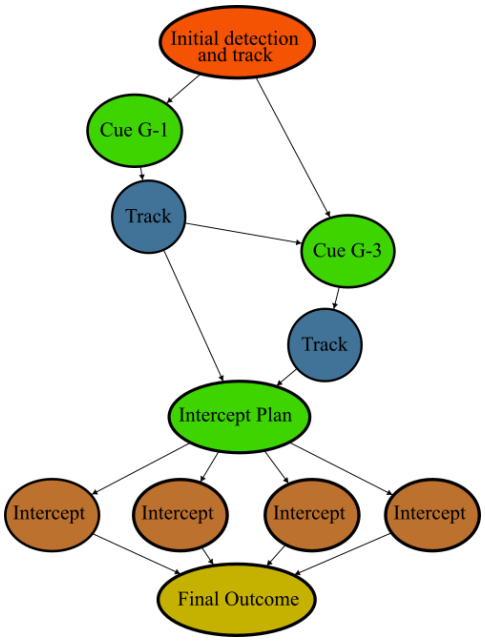

Figure 4: The event stack for threat-1 in the test scenario.

Combining Event Stacks

Based on the timing of the threats, there is a possibility of resources being tasked by multiple threats at the same time. This is particularly true of the sensors. In the physics simulation, this is accounted for by the resolution area in the ground sensor. In the event model, we need to recognize that some events will be degraded because, while they are independent events when it comes to their outcomes, the probability of each event being successful decreases simply because both events are using the same resources. We represent this in the event stack by connecting these events with a dotted line (see Figure 5). In the stochastic simulation, we identify when this happens and have an extra parameter for each of the χ values where this interaction occurs.

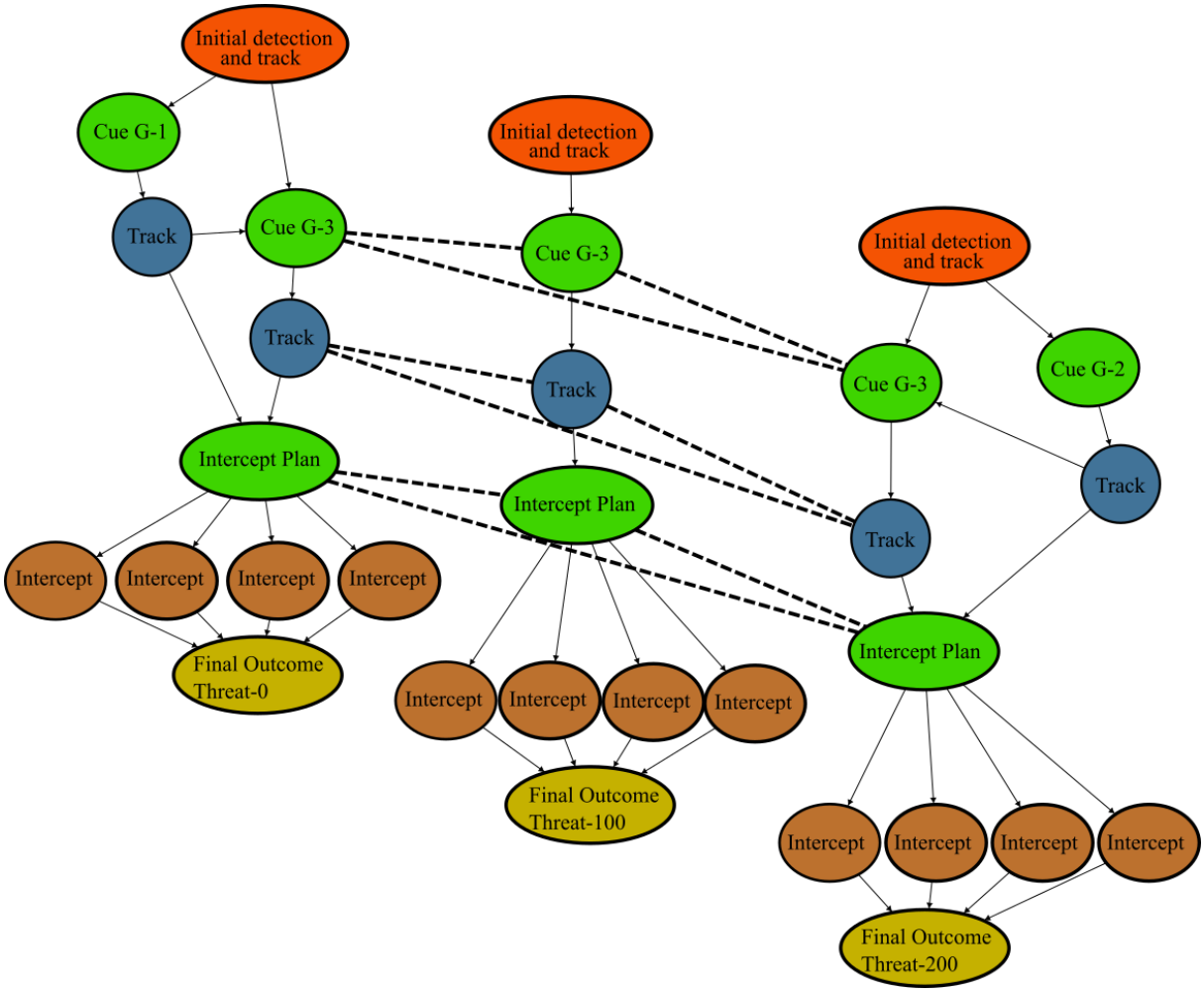

Figure 5: The combined event stacks for threat-1, -2, and -3 where concurrent events take place on the same hardware.

Parallel Runs in the Stochastic Simulation

By converting from a physical domain to an event-based domain, the stochastic simulation no longer depends on iterating through every tick of the physics-based domain. Instead, we just need to iterate over the layers of the dependencies detailed in the event stack. This means that each run of the simulation only needs to iterate through a few dozen events rather than thousands of ticks. For parallelization, this means that we can have more instances of the simulation running on the same hardware footprint.

RESULTS AND COMPARISONS

There are several ways to compare these two approaches to modeling and simulation for this effort. We primarily looked at factors that affect the lifecycle of a simulation tool.

Code and Algorithm Complexity, Creation, and Maintenance

For this effort, we developed algorithms and code from scratch for both simulation domains. This allows us to easily compare and quantify the complexity of developed code using the same coding language, similar design principles, and shared implementations of threat operations.

As discussed previously, the physics approach required four different types of models. These were instantiated into 27 unique models for the simulation that needed to be run. The last impact of a threat occurred at tick 5499. This means that, for each run of the simulation, each of the 27 models needs to act 5499 times. For the sensors and the C2 models, each of these actions requires more and more effort over time because of the accumulation of history that must be accounted for to generate a track.

The stochastic approach uses the parameters for 26 of those models (but not the models themselves) to generate a combined event stack of 57 events. The combined event stack generation is effectively a preprocessing of the scenario. Once the preprocessing is complete, the combined event stack can be passed to the simulation to be run. The simulation needs to iterate through the 57 events only once to complete. Further, the stochastic simulation architecture is simpler than that of the physics simulation because there is no need for a messaging bus or history storage.

This difference in complexity is magnified substantially if we are running simulations in parallel. Each parallel instance of the physics simulation requires iterating through each of the 5499 ticks, its own message bus, and history storage, but the stochastic simulation only needs to iterate through the 57 events. If we are looking at hundreds or thousands of parallel runs, this is a substantial difference in the number of operations needed to complete the task.

The algorithms in the physics approach require an understanding of physics, the interdependencies of the different systems, and the messaging infrastructure required to implement those interdependencies. This requires a unique model for each type of system being simulated. In the stochastic approach, there is only one type of model which is effectively a PDF. The set up of the PDF requires some customization to allow for different inputs but, in the end, the only operation that is required is that the model accepts a previous probability, generates a final PDF, and makes a draw from that PDF. This is a simple and well understood set of operations making the amount of code required to implement the stochastic simulation less than 500 lines of code. On the other hand, the implementation of the physics simulation was more than 2,000 lines of code. This implies that the creation and maintenance of the stochastic models and simulation is substantially easier than that of the physics models.

Runtime and Accuracy

In the physics-based model, we introduce some errors. This is intended to mimic a real-world environment. This is common practice with simulations, but it does mean that a single run of the physics-based simulation is not sufficient to determine the behavior of the SoS. In a similar way, we need to run the stochastic simulations of the event stack multiple times. To keep the comparison between the two methods as similar as possible, we wanted to run 1,000 instances of each simulation, further research is needed to determine how many runs of the event-based stochastic simulation are statistically significant. To get a good idea of how both simulations behave, we run several instances of the simulations in parallel. In the case of the physics-based simulation, this means running the full simulation in

each parallel instance. For the stochastic simulation, this means just running the event stack in each parallel instance. The results are summarized in Table 1.

Table 1: Results of running 1,000 runs of both simulation types.

Simulation Type	Average Run Time (μs)	Threat Destruction Rate	Standard Deviation
Physics-Based	224,413.544	Threat 0: 0.9501 Threat 100: 0.9293 Threat 200: 0.9450 Threat 1000: 0.9492 Threat 1001: 0.9489	The standard deviation does not apply here because it is a two state result; the threat was intercepted or not.
Stochastic-Based	373.134	Threat 0: 0.9433 Threat 100: 0.9388 Threat 200: 0.9419 Threat 1000: 0.9441 Threat 1001: 0.9460	Threat 0: 0.0236 Threat 100: 0.0251 Threat 200: 0.0247 Threat 1000: 0.0289 Threat 1001: 0.0260

As we can see from the results of 1,000 simulations, the physics-based simulation takes three orders of magnitude longer to run than the stochastic-based simulation, while showing similar trends in the threat destruction rates. For instance, the threats launched at $t = 1{,}000$ and $t = 1{,}001$ cross the scenario area and spend particularly large amounts of time in the field of view of the third ground sensor. This is reflected in both simulations with slightly higher threat destruction rates than the threats launched at $t = 100$ and $t = 200$. Similarly, the threat launched at $t = 100$ has the lowest threat destruction rate in both simulations. This is due to only being tracked by one ground sensor.

The formulas for χ, described above, could be further refined to better reflect the physics-based simulation. This could be done in several different ways; from in-depth investigation into the quality of the messages being sent between elements in the physics-based simulation, to using a neural network to determine higher quality definitions for χ.

One of the further benefits of the stochastic-based simulation is that we can collect the draws for each event in the event stack. Allowing us to identify the events that contribute the most to the overall success (or failure) of the stack. This can also be done in the physics-based model, but it requires interpretation of the messages sent between the elements of the SoS. As each message set is different there must be a unique set of interpreters for each message set, making the analysis of the root cause of a low destruction rate much more difficult and complicated. Such analysis is beyond the scope of this document but worthy of further discussion for stochastic-, physics-, and hardware-in-the-loop-based simulations.

STOCHASTIC M&S IN THE LARGER M&S ECOSYSTEM

While the outcomes of the stochastic M&S are not always the same as those of the physics M&S, the stochastic approach does provide some insights into the behavior of a system of systems. The shorter run times, the rapid implementation based on generic models, and the ability to make reasonable predictions about the outcome of an SoS, indicates that there is a place for stochastic M&S in the M&S ecosystem.

High Fidelity M&S Use Optimization

Currently, physics-based M&S are the only accepted approach for understanding how an SoS will respond to different initial conditions. This is a problem because the cost and speed of such M&S makes it difficult to fully characterize the SoS response for all situations and edge cases.

We propose introducing stochastic M&S to the SoS testing environment in order to better utilize the physics-based tools. This approach would use an existing physics M&S to refine a stochastic M&S based on the results of a handful

of runs of the physics M&S. Then the stochastic M&S would use a Monte Carlo approach to vary the inputs to the simulation (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953) (Hastings, 1970) (Liu, Liang, & Wong, 2000). This would provide a map of how the SoS might respond to the full range of possible inputs. The results of such a map should be spot checked by the physics M&S. Any problem areas that might have more variability in response should be the focus for further physics M&S testing. The goal being to focus the physics-based testing in the areas where a high level of detail is needed, while establishing confidence in the SoS behavior in the broader domain of possible inputs. This would optimize the use of the physics M&S, ensuring that the time and cost of running such simulations (particularly hardware in the loop simulations) are being done for meaningful and consequential scenarios and inputs.

Early Design Testing

The simplicity and common implementation of the models for stochastic M&S means that a library of event models could be created. The library of stochastic event models would be completely independent of any system design or operation. In event-based stochastic M&S, a system's design is expressed in the way that events connect in the event stack. The operation of a system's components is captured in the parameters that feed the PDFs. This means that stochastic M&S could be used as an early check to ensure that a design will satisfy system requirements and expectations particularly about reliability and redundancy. Some of the original concepts for event-based stochastic M&S came from MBSE design tools like activity diagrams and sequence diagrams. The information in these two diagrams can be extended with information about PDF choices and parameters to generate the event stack used by the stochastic M&S. This allows M&S tools to be introduced in the development phase of a system's life cycle, rather than waiting to build M&S only after fully designing the system.

CONCLUSION AND FURTHER EFFORTS

While stochastic M&S will never fully replace physics M&S, it does have a role to play in the overall M&S environment and in the systems engineering process. Stochastic M&S is much simpler to set up, more universal to implement, and faster to run, making it a good tool for a low resolution first look at the behavior of a complex system of systems.

The goals of this effort were to build a testbed where we could do some initial comparisons between physics M&S and stochastic M&S approaches. Now that the basic concepts have been developed and implemented, there is a wide range of possible efforts that can be explored.

In this effort, we intentionally kept the level of abstraction of event stacks high with the aim of replicating the overall behavior of the system of systems. The level of abstraction could be lowered to include events that are internal to a component (i.e. an interceptor probability to make it to a target location, or the timeliness of C2 developing an intercept solution) and more interstitial events (i.e. a sensor being in a state where it can be cued when it receives a cue, or refining an intercept location after interceptor launch). This lower level of event abstraction could provide insights into how individual systems behave within the context of the larger SoS.

There is still a large amount of experimentation to be done with different PDFs and different approaches to adjust those PDFs to better reflect the physics-based and real-world results. The event based stochastic approach may also have unique capabilities to model and simulate possible human interactions with a system or SoS.

REFERENCES

Albrecht, H., Gläser, R., Harder, G., Krüger, A., Nilsson, A., Nippe, A., . . . Has, C. (1990). Search for hadronic b→u decays. *Physics Letters B, 241*(2), 278-282. doi:10.1016/0370-2693(90)91293-K

Hastings, W. K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika, 57*, 97-109. Retrieved from https://api.semanticscholar.org/CorpusID:21204149

Jamshidi, M. (Ed.). (2008). *System of Systems Engineering: Innovations for the 21st Century* (1 ed.). Wiley.

Liu, J., Liang, F., & Wong, W. H. (2000). The Multiple-Try Method and Local Optimization in Metropolis Sampling. *Journal of the American Statistical Association, 95*, 121 - 134. Retrieved from https://api.semanticscholar.org/CorpusID:123468109

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953, June). Equation of State Calculations by Fast Computing Machines. *jcp, 21*(6), 1087-1092. doi:10.1063/1.1699114

Mooney, D., & Swift, R. (1999). *A Course in Mathematical Modeling.* The Mathematical Association of America.

OMG Systems Modeling Language (OMG SysML™) Version 1.6. (2019). OMG Systems Modeling Language. Retrieved from SysML.org: https://sysml.org/.res/docs/specs/OMGSysML-v1.6-19-11-01.pdf

Pinsky, M., & Karlin, S. (2010). *An Introduction to Stochastic Modeling.* Elsevier Science. Retrieved from https://books.google.com/books?id=RL74jxXd-zQC

Rubinstein, R. Y., & Kroese, D. P. (2011). *Simulation and the Monte Carlo Method* (Second ed.). Wiley. Retrieved from https://books.google.com/books?id=yWcvT80gQK4C

Skinner, S. (2023). Simulation model abstraction issues for Digital Twins; Separated at Birth? *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*. Retrieved from https://www.xcdsystem.com/iitsec/proceedings/index.cfm?Year=2023&AbID=120868&CID=1001