Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

SIMULATION OF MAINTENANCE PROCESSES IN THE BIG DATA ERA

Vitali Volovoi

Independent Consultant
Alpharetta, GA 30022, USA

ABSTRACT

Maintenance processes of repairable systems have been extensively studied in the past. The resulting simple solutions have proven to be remarkably effective. It requires complex and time-consuming simulations to improve on those simple solutions, and reliable input data is even harder to get. However, new technologies, epitomized by Big Data and the Internet of Things, change the data-availability part of the equation. As a result, there are new exciting possibilities for modeling more subtle effects, and developing processes for easily (and therefore frequently) updated inputs. Modeling decisions can be repeatedly tested on the data, and the models can be quickly adjusted to better reflect reality and even to compensate for missing pieces of the data. In this context, the transparency and simplicity of models becomes a larger virtue. Several examples of the insights based on real-world large-scale applications of predictive analytics using simulation are discussed.

1 INTRODUCTION

In abstract terms, maintenance processes can be viewed as two coupled loops, each centered around matching supply and demand. The first is an operational loop that represents a match between available engineering systems and their customers' needs. For example, an airline can have a fleet of various aircraft utilized to carry passengers. Here, each aircraft supplies the needed service for its customers, but must be serviced from time to time (maintained) in order to operate properly, which constitutes the second supply-demand loop. In the second loop, the system itself provides the demand for a maintenance signal, which is matched by the supply of maintenance support.

Maintenance can be related to system repair, but does not need to be: changing oil and filling up the fuel tank of a car also can be considered maintenance. Since the French Academy of Science stopped accepting applications for *Perpetuum Mobile* in 1775, it has been unrealistic to expect an engineering system to be completely maintenance-free. Even cell phones must be recharged (and more often as their batteries age), which can also be considered to be a type of maintenance. However, the subject of this paper is engineering systems that require more involved maintenance, such as that required for aircraft or gas turbines. This type of maintenance usually requires replacement parts and subsystems as well as specialized personnel and facilities. In the past, two factors contributed to each loop's being analyzed separately:

- The prevalence of the stove-pipe approach to engineering that heavily relied on a divide-and-conquer means of dealing with complexity. The latter was a necessity due to limited computational capabilities, but also was often a blessing—these limitations brought the principle of Occam's razor to the fore and forced a focus on essential matters, while abstracting away the rest;
- Separation of economical interests, with the engineering system acquisition on the one hand and operation and support on the other. In this traditional setting, an Original Equipment Manufacturer (OEM) would design and produce the engineering system (e.g., an aircraft) and sell it to an operator (e.g., an airline). As a result, an OEM that produces spare parts and conducts repairs would act on

an individual transactional basis with the operator, and therefore would not have full visibility in the operational loop.

Both of these factors become increasingly less relevant in the modern environment (Volovoi 2016). Nevertheless, the consideration of both loops simultaneously represents a considerable modeling challenge: each system can consist of thousands of components, while an operator might have hundreds or even thousands of such systems. Here the word "component" refers to an individually tracked entity. Depending on the context, a component can represent a different level (so-called "indenture"): e.g., it can be a line-replaceable unit (LRU) or a shop-replaceable unit (SRU). As a result, a combined model can have on the order of a million distinct entities that need to be tracked simultaneously. Fortunately, in the case of spare parts, a meaningful approximation can be made by considering a component-specific view of the problem. Specifically, the flow of all components of a given type can be modeled independently of the flows of other types of components. This independence assumption has led to the classical readiness-based sparing theory that is based on queueing analytical models, is widely accepted worldwide, and provides a means of establishing optimal stock levels of spares at various locations. In this context, the optimality is assured within the constraints of the model (Sherbrooke 2004).

There are several sources of dependency that can limit the accuracy of these analytical models:

- Capacity (resource) constraints during repairs, as explored in (Díaz and Fu 1997). Since multiple types of components can be repaired at the same facilities, understanding capacity constraints requires understanding the combined effect of the parts flowing through the repair facilities.
- The varying importance of individual components on the operation of the "end item," i.e., the system. An obvious example is the presence of redundancies in the system (so that the system can still operate when some components are down). Elaborate discrete event simulations [such as TIGER (Castillo 1989)] have been developed to account for the redundancies. However, from the practical perspective, the very presence of redundancies indicates the importance of provided functionality (e.g. for safety-critical systems such as aircraft engine controls). As a result, operations with depleted redundancies are often restricted and require consideration of additional complex scenarios (such as those associated with deferred maintenance, or what is referred in the Aerospace industry as the time-limited dispatch). As a result, developing sparing policies without considerations of the intra-system redundancies makes a lot of sense in most cases. However, there is another less appreciated coupling effect among components related to the fact that varying numbers of parts can be required to fix a given failure.
- The varying importance of individual systems in delivering overall functionality. This is essentially a system-of-system effect that is related to inter-system redundancy, as some of the systems can be more critical than others; for example, a ground station might be more critical than the availability of a remotely piloted aircraft.
- The closed-loop effect on the demand for spare parts. Let us consider a wind farm: if some windmills are down and fewer of them operate, fewer of them will break as well (there is negative feedback). In this case, the total number of operating windmills is needed to estimate the demand for an individual type of part. This situation can be contrasted with a fixed demand for system utilization that is provided at the fleet level (for example, if more aircraft are down, it implies that the aircraft remaining operational have to fly more).

When analytical solutions (most of which have been developed for over fifty years) are deemed to be insufficient, discrete event simulations (DES) are resorted to. In particular, this is the case when maintenance resources are considered (such as personnel, equipment, and facilities). The original models often have venerable pedigrees, especially in military applications, and often still have the good old-fashioned look and feel of the punch-card era. When those programs are updated, priorities are often given to introducing more features at the expense of user-friendliness; unfortunately, as the list of features (i.e., specific scenarios

and options that can be modeled) grows, very little attention is paid to the overall impact on modeling fidelity. As a result, a model can be excruciatingly specific in terms of some details, while making sweeping assumptions in other areas. Most importantly, the internal "business logic" of those models is often hidden from the user, with a significant impact on the results. Since modeling scenarios are quite complex, a gut check might not be sufficient to verify the model; at the same time, while modern computers have no problems crunching a Monte Carlo simulation with half a million moving parts, whether the internal logic of the model actually reflects the intentions of the user is an altogether different matter.

Until recently, inputs to DES were a precious commodity: their painstaking collection has often provided the major bottleneck of the modeling. As a result, simulation is still predominantly an "offline" activity: some experts extract the needed data, process it in order to create simulation inputs, and then proceed with the simulation. This extraction and processing data is labor-intensive and time-consuming, and so it is done once or very infrequently (for example when major redesign to the process is contemplated).

The era of Big Data fundamentally changes this balance, as abundant data becomes readily available: the modeling itself can become a bottleneck. The prevalent "black-box" tools in current Big Data applications (machine learning) are mainly agnostic to the type of problems they are modeling. The "one-size-fits-all" approach has an obvious advantage in terms of scalability; however, maintenance processes are structured and obey certain business rules, and if a model fails to capture those rules explicitly, the predictive power is drastically reduced. This creates an opening for DES to be efficiently used in the role of predictive analytics in conjunction with the technologies of Big Data and the Internet of Things; to achieve this, however, the affinity to the principle of Occam's razor needs to be rediscovered. A big part of the associated challenge is establishing the "just right" level of abstraction and filling the data gaps.

The rest of the paper is organized as follows: first, Abridged Petri Nets (APN) (Volovoi 2013, Volovoi 2015) is reviewed as a means to construct minimalistic DES; then a simple example of the operation of a fleet of cars is described, with the results demonstrating the challenges in understanding the stochastic behavior of even relatively simple systems; next, a more realistic two-part example of aircraft maintenance is discussed; finally, some conclusions are provided.

2 APN AS A MINIMALISTIC DES

As discussed in detail in (Volovoi 2015) a process-interaction view underpins the vast majority of modern DES software tools. Reflecting the manufacturing applications' heritage, they mainly model flows of entities through a network of servers that transform those entities (transactions) by changing their attributes. This view "implies system events by describing what happens to an entity as it encounters the system" (Nelson 2002). A certain degree of hierarchical (top-down) perspective is inherent: there are transient, low-level entities (jobs or transactions) and persistent higher, system-level, entities (such as resources) (Roeder 2004).

In contrast, in Stochastic Petri Nets (SPNs), the system is comprised of individual components (entities) that can change their states, thus triggering or preventing the state changes of other entities. In other words, SPNs are based on local (peer-to-peer, or bottom-up) interactions among entities, and the concepts of resources and queues are derived, rather than being fundamental properties, as they represent specific examples of coordination (interaction) among individual entities. Using the language of agent-based simulation, models utilize the point of view of "agent activities" rather than business processes (Bonabeau 2002), and as such can be considered a particular restricted type of agent-based models. Here, an activity is interpreted as a time delay between state transitions for a given entity (agent). Petri nets model the states of individual components rather than the explicit states of the entire system.

In Petri nets, Markov chain-state diagrams (that consist of states and transitions) are complemented by two additional types of objects: first, small filled circles (called tokens) denoting individual components are introduced, each placed inside of one of the larger hollow circles that denote the potential states of those components (the latter entities are named "places" as opposed to "states" in Markov diagrams). Second, in order to model interactions among components, the tokens are routed among places via intermediate stops or junctures, called transitions, which are denoted with solid rectangles. Two places cannot be connected

by an arc directly; instead they must be connected through a transition. The number of input and output arcs does not need to coincide, enabling the merging and splitting of tokens and their routes. The timing of state changes can be modeled by specifying time delays for transition "firing," an atomic (i.e., indivisible) action that removes tokens from all input places for the transition and deposits tokens into its output places. Such Petri nets are timed Petri nets, or, more specifically, SPNs (Balbo 2007, Haas 2002), when delays can be nondeterministic and follow a specified distribution. In simulation, no limitations on the associated types of distributions are needed, but historically SPNs referred to models with exponentially distributed delays only, so that they could be converted to Markov chains and solved using appropriate techniques for the underlying differential equations. The APN modeling framework is a streamlined version of SPNs (Volovoi 2015, Volovoi 2013), and its properties are reviewed next.

2.1 Properties of APNs

1. An APN is defined as a network of places (denoted as hollow large circles) that are connected by transitions. Changes in the system's state are modeled by transition firing: i.e., the moving tokens go from the transition's input places to its output places. The combined position of APN tokens at any given moment represents the net "marking" that fully specifies the modeled system.
2. A regular transition has a single input and a single output place.
3. A token has a discrete label (color) that can change when the token is fired in accordance with the policy specified by the firing transition. Tokens also have continuous labels (ages) that can change both when tokens move, and while they stay in the same place with the progression of time [the latter property is specified by the aging transition for the place, which is not necessarily the same as the firing transition (Volovoi 2004)].
4. A transition is enabled or disabled based on the combined marking of input places of the associated triggers (inhibitors and enablers). An inhibitor is depicted as an arc originating at a place and terminating at a transition with a hollow circle. An inhibitor of multiplicity k disables the transition at which it terminates if the number of tokens in its input place is at least k. An enabler is the opposite of an inhibitor (depicted similar to an inhibitor, but terminated with a filled circle): a transition is disabled unless an enabler of multiplicity k has at least k tokens in its input place.
5. Transitions have color- and age-dependent policies that specify the delay between the moment when the token is enabled and when it is fired (for example, one can specify separate distributions for distinct colors, while age can accumulate as the cumulative distribution function of the aging transition). If a token-transition pair is enabled, a firing delay is specified based on the combination of token and transition properties. If the token stays enabled throughout the delay, the token is fired after this delay expires. If there are multiple enabled tokens in the same place, they all can participate in the firing "race" in parallel. Similarly, the same token can be involved in a race with several transitions. If a token-transition pair is disabled, the firing is preempted (however, the aging label of the token can change as a result of being enabled for a finite amount of time).
6. Delays can be deterministic (including zero delay) or follow specified distributions. The firing after a specified delay is "atomic": a single action of moving a token from an input place to an output place (tokens don't dwell between places, in contrast to the case with some SPNs (Bowden 2000)).
7. Joints (token junctures depicted as triangles) connect three places, and are a specific subset of immediate transitions in regular SPNs. These entities are analogous to the batching and duplicating building blocks in process-interaction frameworks for discrete-event simulations (Law and Kelton 2000). A split joint has a single input place. A token from this place is duplicated, and two identical copies (i.e., having the same color, age, and token ID) are immediately deposited into the joint's two output places. The merge joint provides the opposite functionality by merging two tokens from its two input places and depositing a single token into the output place. A dominant input can be specified (and denoted with a thicker arc line), stipulating which of the two tokens provides all the attributes to the merged token. One of the four merging policies can be selected (the choice is indicated by

a capital letter inside the joint) stipulating which tokens can be merged: A = any two tokens; C = only two tokens of the same color; I = only tokens with the same ID; R = a token's previous ID matches the current ID of the other token.

8. The performance of the system is based on the statistical properties of the net marking. "Sensors" or "listeners" assigned to a place provide a means for evaluating relevant statistics about the number of tokens at that place (e.g., the chances of crossing a threshold for a specified period of time, the number of times a threshold is crossed, the mean and the variance of the number of tokens, or the correlation with similar quantities in another place). The presence of a sensor is denoted with a solid rectangle next to a place, and multiple sensors can be assigned to the same place.
9. Fusing places, commonly used in hierarchical Petri nets [see, for example, (Jensen 1993)] is employed to connect different parts of the model. Fused places appear as distinct graphical entities during model construction, but represent the same entity in simulation.

3 CHALLENGES OF MODELING EVEN SIMPLE STOCHASTIC PROCESSES

Let us consider a simple example (Volovoi 2013) of a household consisting of two family members (later referred to as customers) and a car. Each customer can be in one of three states: not needing a car (N), driving a car, or waiting for a car; we assume that being a passenger in the car qualifies as state N. The inputs to this model could include the usage pattern of each customer, e.g., the frequency and duration of trips, as well as the car properties, e.g., the frequency of breakage and the duration of the repairs. The outputs would be the "performance" measures or metrics of this "system," such as the frequency and duration of unsatisfied demand. These measures can help to make educated decisions about changes to the system, e.g., whether it makes sense to get a second car.

The corresponding APN diagram is depicted in Figure 1A. Each family member is denoted by a token, and can be in three possible states (denoted with places): the car is not needed ("Not needed"), waiting for the car ("Waiting"), and driving the car ("Driving"). In this model there is no token representing a car; instead, an inhibitor of multiplicity one is used to prevent more than one token appearing in the "Driving" place. This inhibitor stipulates that only one car can be driven at a time. There are three transitions in the model, numbered as shown in Figure 1A. Next, we introduce the possibility of the car breaking from time to time. If a trip is interrupted by the car's breaking, the family member has to wait until the car is fixed, and then attempt the interrupted trip again (from the beginning).

Instead of creating two separate places for both possible states of the car (broken or not), we employ token colors (i.e., integer labels) to create a more compact model, as depicted in Figure 1B. Here we take advantage of the fact that a customer might not be able to drive a car (the car is unavailable) for two distinct reasons—either when another family member is already driving the car, or when the car is broken. We combine both possibilities into a single place, "Unavailable," and introduce an immediate transition, 4, that "pushes" the token representing a family member to the "Waiting" place when the car breaks (due to the enabler of multiplicity two). To ensure that the tokens representing cars and people do not get mixed up, we differentiate them by color: when the car token gets to the "Unavailable" place, it has color 1, as opposed to the people tokens that have color 0; and outgoing transitions from that place are color-dependent (transitions 3 and 4 are only sensitive to color 0 [people], while transition 6 is sensitive only to color 1 [car]).

The model is set up for easy scalability: we simply need to add more tokens and adjust the trigger multiplicity to change the number of cars and customers. E.g., for a fleet of $k = 10$ cars with $n = 20$ customers, the model is shown in Figure 1A (note that the enabler multiplicity is $k+1 = 11$, and the inhibitor multiplicity is $k = 10$). At the time of the snapshot, we have two cars broken, eight cars driven, 11 customers not needing a car, and one customer waiting for a car. In this model, if a car breaks but there is another car available, the customer simply switches to another available car.

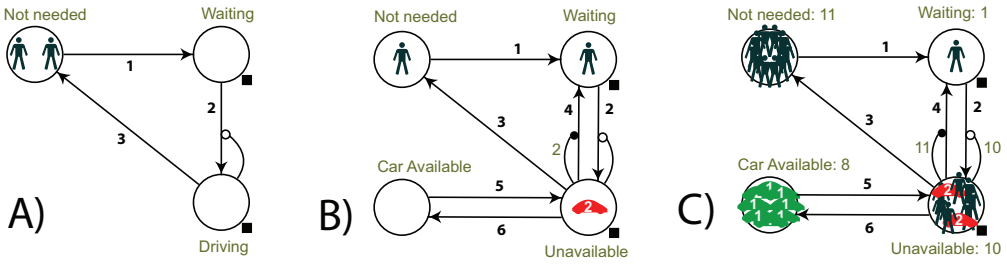

Figure 1: A) Two customers and one car, B) with car breaking; C) 20 customers and 10 cars.

3.1 Simple Excursions

While this model appears to be very simple, as shown next, the behavior of the model defies conventional wisdom in terms of performance dependence on the parameters of the model and in terms of the pace of conversion to a steady state.

Let us consider the duration of simulation to be $T = 12$ hours, and consider a situation where each family member does not need the car for two hours, and then would like to take a two-hour trip. We can assign transitions 1 and 3 a fixed duration of two hours, and make transition 2 immediate (in the simplest implementation we can assign a small fixed value ε to this transition; in the considered example $\varepsilon = 1 \times 10^{-6}$ hours). The result is a fully deterministic model (if we don't make a distinction between the family members and therefore do not need to know which of the family members gets the car first). Family members will be driving the car in turns, never waiting for the car (except during the first two hours) and ensuring that the car is effectively driven all the time (the car's utilization is 100%).

Let us now introduce randomness into the model and assign both transitions 1 and 3 exponential delays that, on average, result in a two-hour delay, so that the cumulative distribution function (CDF) has the following form $F(t) = 1 - e^{-\lambda t}$, where $\lambda = 0.5$ per hour. The resulting APN model is equivalent to a simple Markov chain [see further discussion in (Volovoi 2013)], with a steady-state solution indicating that the car utilization is 80%, and a customer would have to wait for a car 40% of the time (or 20% per customer). Figure 2A shows the results using APN simulation (10 million Monte Carlo replications are used here and for the following examples, unless stated otherwise).

Next, let us consider a "mixed" situation where demand delay (transition 1 in Figure 1) is fixed, while the trip duration is exponential (while keeping the mean values of both at two hours). As shown in Figure 2B, while the transient part of the results are distinct, the steady state values are the same as in the Markovian case. However, if we reverse the choice of distributions and use exponential distribution for the demand (transition 1 in Figure 1) and fixed-trip duration (transition 3 in Figure 1), the results are more interesting (see Figure 2C): First, the expected values are oscillating, with the amplitudes decaying relatively slowly; second, the converged values appear to be different from those we observed in the previous two cases. Running the simulation for a longer period of time, as shown in Figure 2, D) confirms this observation. The converged values are 84.5% for utilization and 31% for waiting.

3.2 Family Car That Does Not Always Work

In this example we consider the same problem of a family car as before, but this time we allow for the possibility that there are times when the car can periodically break. While breaks in reality occur relatively rarely (and it takes days to fix), for illustrative purposes we consider a situation where we have more frequent breaks that are also fixed relatively fast. If a trip is interrupted by a break, the family member has to wait until the car is fixed, and then attempt the interrupted trip again (from the beginning).

To model this situation we introduce a car token, and the possibility of the car breaking. Instead of creating two separate places for the two possible states of the car (broken or not), we employ tokens colors (i.e., integer labels) for a more compact model, as depicted in Figure 1B. Here we utilize the

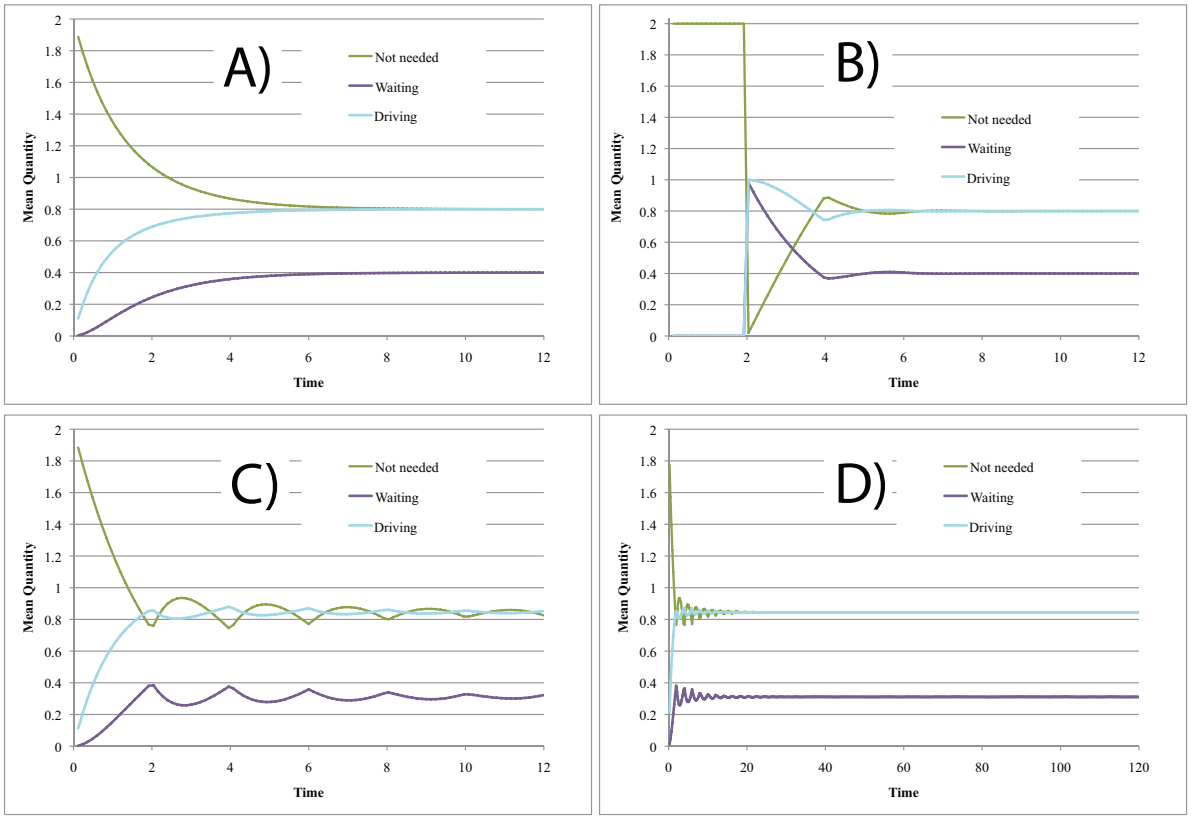

Figure 2: APN car model results using: A) exponential distributions, B) fixed demand and exponential trip duration); C) exponential demand and fixed trip duration; D) the same as C, but 120 hours simulated.

fact that a family member (customer) might not be able to drive a car (the car is unavailable) for two distinct reasons—either when another family member is already driving the car, or when the car is broken. We combine those two possibilities into a single place, "Unavailable", and also introduce an immediate transition, 4, that "pushes" a token representing a family member to the "Waiting" place when the car breaks (due to the enabler of multiplicity two). In order to make sure that tokens representing cars and people don't get mixed up, we differentiate them by color: when the car token gets to the "Unavailable" place, it has color 1, as opposed to the people tokens that have color 0; and outgoing transitions from that place are color-dependent (transitions 2 and 3 are sensitive only to color 0 [people], while transition 6 is sensitive only to color 1 [car]).

Let us consider a specific example where the car breaks in accordance with an exponential distribution with the rate 0.2 per hour, and is restored after a fixed delay of 0.2 hours. This implies that the car is available 96.15% of the time. The results of this scenario are shown in Figure 3A, where the car's availability corresponds to the "Car OK" curve. First, we note that the introduction of the possibility of the car breaking provides damping (or mixing) to the model, so that convergence to a steady state occurs faster. Second, while car utilization (the amount of driving) actually goes up by about 1.5% to approximately 86%, the amount of waiting significantly increases from 31% to 44.4% (or 22.2% per family member/customer). One of the reasons for such a significant increase is the fact that interrupted trips start from scratch. Even more remarkable is the importance of "mixing" that eliminates the effects of coordination, so that now whether we have fixed trip duration or the equivalent exponential delay does not make any difference (in other words, for the Markov model, waiting increases from 40% to 44.4% due to the presence of breaks).

Finally, let us consider the effects of pooling resources, given a fleet of $k = 10$ cars with 20 customers. The model is shown in Figure 1C—note that the multiplicity of the enabler now is $k+1 = 11$, and the

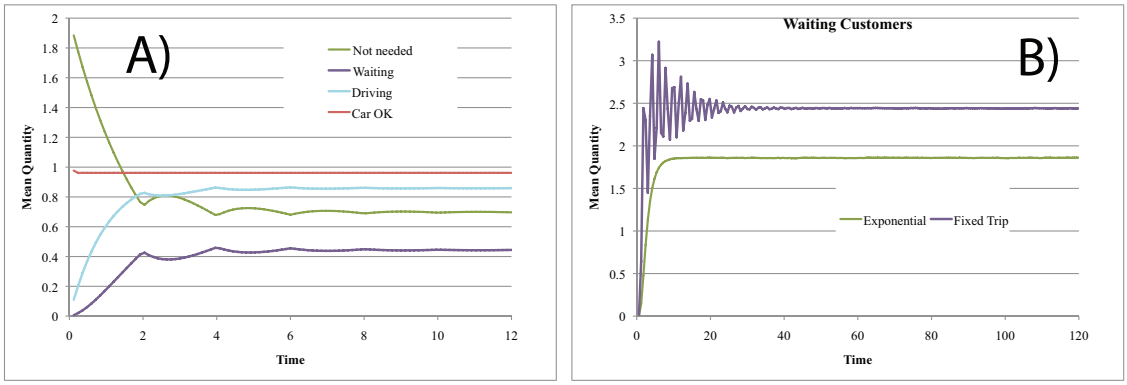

Figure 3: APN car model results using exponential demand and including the possibility of car breaking: A) fixed trip duration for a single carB number of waiting customers with 20 customers and 10 cars.

multiplicity of the inhibitor is $k = 10$. At the time of the snapshot we have two cars broken, eight cars driven, eleven customers not needing a car, and one customer waiting for a car. In this model, if a car breaks but there is another car available, the customer switches to the available car. The benefits of pooling the resources can be easily observed by evaluating the amount of time the customers are waiting for a car (Figure 3B—for fixed trip duration the percentage of waiting time is almost halved (from 22.2% to 12.2% per customer). Increasing the number of customers decreases the damping in the system, and noticeable oscillations are observed for some time. Furthermore, for exponential demand, comparing exponential and fixed-trip durations leads to a reverse trend, as compared to what we observed for a single-car problem without breaks (cf. Figure 2A vs. D): using exponential durations for trips instead of fixed ones decreases the waiting to 9.3% per customer!

3.3 A Simple Depot Process

Figure 4 illustrates the APN model of a simple two-stage depot (repair and overhaul) process. Two types of aircraft are simulated: C-130H, which is processed faster, and other versions of C-130 that are known to have a larger work scope and therefore require more time.

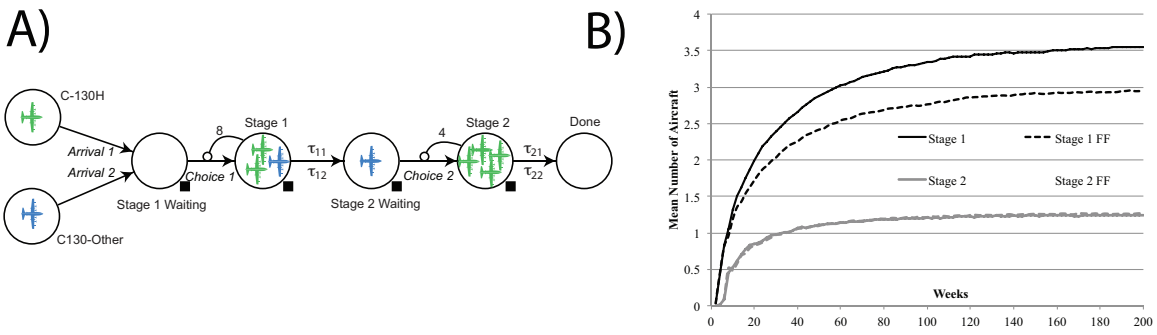

Figure 4: A) Abridged Petri Net (APN) Model of a simple two-stage Depot Process B) Sample results: benefits of an "express lane."

Tokens can be distinguished by color (integer attribute), and transition delays can be color-specific, e.g., so that the processing times τ_{ij} for each the stage and aircraft type can be specified. In Figure 4A, the color green represents C-130H, while blue represents other C-130 aircraft. Transitions "Arrival 1" and "Arrival 2" specify the frequency of aircraft arrival. Inhibitors enforce capacity constraints by specifying

the inhibitors' multiplicity: Stage 1 has 8 spots and Stage 2 has 4 spots. By selecting different color-specific policies for transitions Choice 1 and Choice 2 we can implement different scheduling priorities.

Figure 4C shows simulation-based results. The service times are fixed for all aircraft; C-130H are served twice as fast at each stage. We simulate Poisson arrivals of the once-per-week intensity for both types of aircraft that create service intensity $\eta = 0.9$ for both stages. (As known from the queuing theory, η must be less than unity for a stable system; η close to unity indicates a high-utilization system). Two different policies are simulated: the baseline is a simple First-In-First-Out (FIFO) policy that ignores the aircraft type. The second policy is FIFO within each aircraft type, but selects C-130H first. We refer to this policy as Fast First (FF). Figure 4 B) shows the comparison of the average waiting times obtained in one million Monte Carlo replications of 200-week periods. The waiting figures for Stage 2 are close for both policies. The number of aircraft waiting for Stage 1 shows a significant difference: the FF policy of prioritizing C-130H results in an average of less than three aircraft waiting, compared to the 3.5 for the baseline. This simulation illustrates the difference that can be made by the improved scheduling.

In general, the interactions between "intelligent" selection policies and their compact statistical representation presents one of the great challenges of effective DES modeling. For example, while this simple depot process model assumes Poissonian arrivals, in reality the depots have certain flexibility on when to induct the aircraft for maintenance. In essence there is a window of a certain width (based on aircraft usage) when the aircraft can be inducted. The planners therefore can delay the induction if there is already a queue forming, or induct early if there is an available spot. This pacing of the arrivals cannot be judged based on the traditional variability of the arrival process (based on the second moment of the inter arrival distribution), and the impact of that pacing can be quite significant. However, in this case, there is a simple means representing this pacing: one can introduce an additional place that represents a buffer where the aircraft arrives at the beginning of its window and only moves into the queue after the window ends.

3.4 A Manpower Maintenance Model

Finally, let us consider a more realistic model that explicitly models the two supply demand loops discussed in the introduction. We consider a unit of 10 airplanes that flies two sorties in the morning and two in the afternoon (see Figure 5). Delays for immediate transitions are shown (e.g., 2ε or 3ε), and they are selected to ensure coordination (handshake), where a single action in one part of the network causes a single action in another. Parameters of the model in this example do not correspond to any specific aircraft, but the magnitude of the parameters is selected to be representative of the field data. The flight duration is represented with lognormal distribution with mean a value of two hours with the Squared Coefficient of Variation (SCV) 0.5. While the aircraft flies, it accumulates discrepancies (gripes) at the constant rate of two per hour. When an aircraft, lands there are two possibilities: if no gripes were discovered, an aircraft returns to the "Available" place; otherwise, the aircraft is down until all the gripes are fixed. Here, no sparing delays are modeled and only a single maintenance specialty is considered, but the model is easily expandable to represent multiple specialties, as explained below.

For a complex system such as aircraft, there are thousands of different failure modes (usually distinguished by a specific work unit code). Different tasks require varying amounts of time and crew sizes. A detailed approach would provide a separate network for each failure mode; instead, we group the different failure modes together based on the crew size required for a task. Given the first two moments (mean and variance) of task duration along with the relative frequency of this task, a fairly straightforward calculation allows one to obtain the two moments for the combined distribution. Importantly, tasks that require a different numbers of people cannot be combined in a similar fashion, so those tasks are modeled separately; token colors are probabilistically assigned based on the resulting frequency of the crew size required. In this example, blue and purple colors represent two-man and three-man tasks, respectively.

The model allows maintainers to work on some unrelated tasks, so they are only available for part of a shift. One can make an analogy with lunchtime in a restaurant: there is a certain window of time when customers (in this case gripes) show up, and so the schedule is adjusted to ensure peak availability

of the service when needed. Here we consider three shifts, with 30 maintainers available during the first two shifts and 15 in the third shift. The maintainers work on the aircraft only two hours per shift, and the timing of that two-hour window is selected to match peak demand. Here we consider a scenario where this productive time starts 1.5 hours after the take-off time for each of the two day shifts. (The analogy with a restaurant is imperfect, as one hopes a customer does not need to wait until dinner [or the next day] to get fed if she fails to get served during lunchtime.) There is a 30-minute overtime limit: if the productive part of the shift is over, the maintainer can work for up to 30 minutes more on an open task.

A portion of the maintenance view is shown in Figure 6. There are two additional views (pages) not shown for brevity's sake: the "stalls" page, where each down aircraft is waiting for all its discrepancies to be fixed, and the "shifts" page, where the shifts and productive subshifts are handled. (For the maintenance view, the portions enforcing the overtime limit and shifts are also not shown.) Different pages of the model are connected by fused places that are present on different tabs and are akin to elevators connecting floors of a building. Fused places are shown with thicker circles. If the tasks are distinguished by maintainer specialty (i.e. structures vs. avionics, etc.) the tokens representing tasks would have additional colors, which would allow routing the token to an appropriate view for each specialty. The overall model remains relatively compact as, for example, five specialties would have a model with eight tabs. Let us consider a scenario where 60% of the tasks require a two-man crew and 40% require a three-man crew. The respective means for those jobs are $m_2 = 3$ and $m_3 = 2.5$ hours, and the SCVs are $c_{s2} = 0.2$ and $c_{s3} = 0.15$.

Results are shown for ten days with 100,000 replications (no activities occur on the weekends). We compare two different policies: when multiple tasks arrive at the same time, the Large Task First (LTF) policy selects a task with the larger crew size. In contrast, the Small Task First (STF) policy prefers the tasks with the smaller crew size. The LTF model is shown in in Figure 6. For STF, one needs to remove an inhibitor from P1 to T4, and replace it with an inhibitor from P2 to T3, and also reverse the order of the durations for the T4 and T3 transition, so that T3 and T4 and will have 5ε and 2ε, respectively. The results are shown for the last day of the simulation (after the warm-up). Figure 7A shows down aircraft. Here STF provides slightly better results: time-averaged down aircraft is 11.7% vs. 12.6%. Figure 7B shows the number of waiting tasks. If we weigh the waiting time with respect to the crew size, then the time averaged waiting man hours will be smaller for LTF as compared to STF (13.19 vs. 13.95). The resulting model can be used for different operational units and by pulling the data from different time periods.

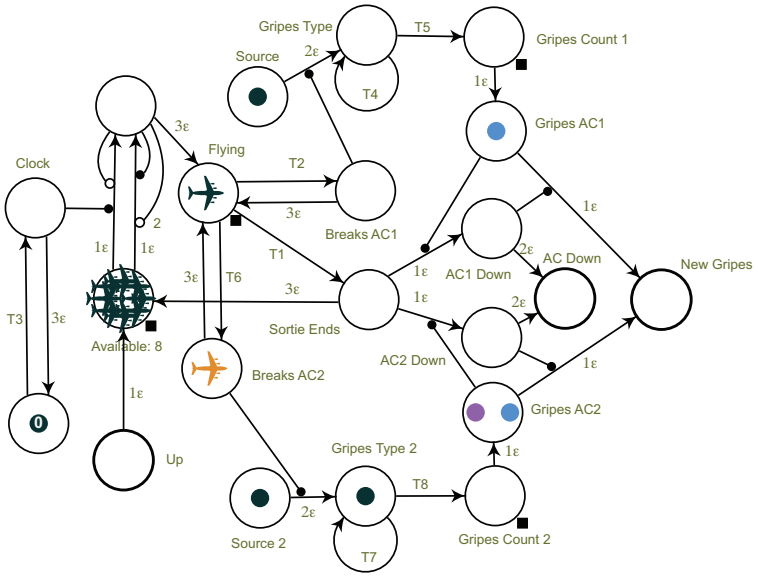

Figure 5: APN manpower maintenance model, operational view.

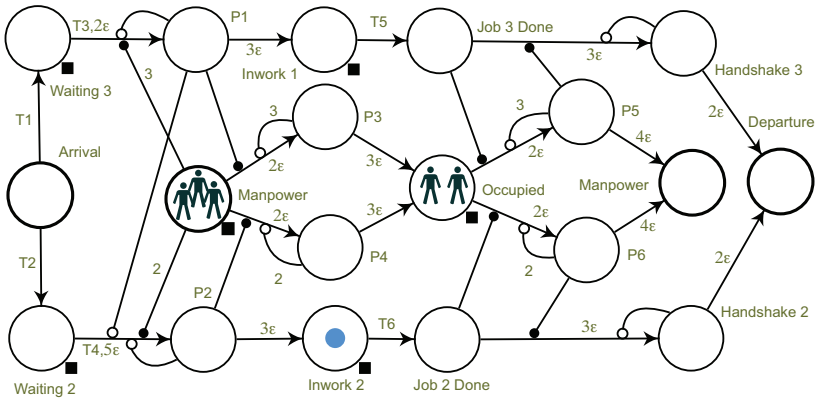

Figure 6: APN manpower maintenance model, mantenance view.

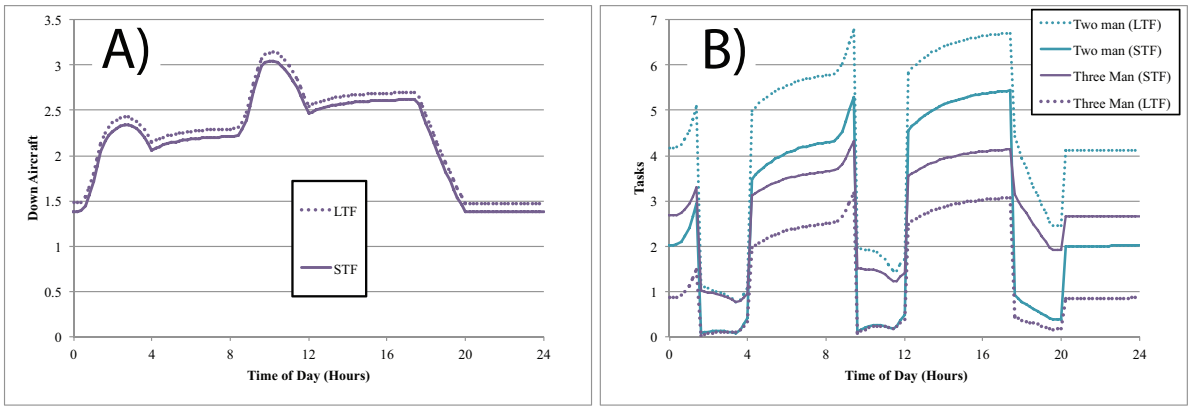

Figure 7: APN manpower model results: A) Down aircraft B) Number of waiting tasks.

4 CONCLUSIONS

The changes that currently occur in the modeling of the maintenance of complex systems are discussed, along with the resulting implications for the role of discrete event simulations in this modeling. The changes stem from the previously unimagined wealth of transactional data, and the increased attention to improving maintenance and operation support due to the increasing trend of transferring the financial burden of those activities onto the entities that have access to that data. The former is driven by Big Data and the Internet of Things technologies (Volovoi 2016); the latter is driven by the increasing trend of selling a complex engineering system as a service rather than a product, and the resulting popularity of long-term service agreements and similar arrangements, such as Performance-Based Logistics (PBL). As a result, there are both an opportunities and incentives to improve on the existing sparing and maintenance models, and to consider effects that previously were neglected. Discrete-event simulation is a legitimate means of achieving this goal in combination with analytical means (such as queueing networks). DES can become a vital tool for predictive analytics for well-structured processes, such as maintenance. However, in order to be successful in this endeavor, there has to be a drastic departure from the current trend of considering DES as a fancy black box with its inner workings hidden from decision makers. The lack of modeling transparency and "the more complex and detailed, the better" mentality can seem to be advantageous from the competitive perspective; however, in the long run, this practice is hurting the use of DES, as its role and usability in decision making falls short of what it could be. To this end, the role of visual representation of modeling processes and dependencies comes to the fore. Abridged Petri Nets, a recent version of Stochastic Petri Nets, can serve as a useful compromise between traditional DES and the more

flexible (yet lacking standard building blocks) agent-based simulation. The graphical nature of the models and the small number of standard building blocks facilitate easy and fast model creation. The models are self-documented, and therefore uniquely suited for tool-independent auditing and for conveying the insights from simulation to decision makers. These features can be useful for lowering the effective level-of-effort barrier for simulations and fully utilizing DES for predictive and prescriptive analytics.

REFERENCES

Balbo, G. 2007. "Introduction to Generalized Stochastic Petri Nets". In *Formal Methods for Performance Evaluation*, edited by M. Bernardo and J. Hillston, Volume 4486 of *Lecture Notes in Computer Science*, 83–131. Springer-Verlag.

Bonabeau, E. 2002, May. "Agent-Based Modeling: Methods and Techniques for Simulating Human Systems". *Proceedings of the National Academy of Sciences* 99 (3): 7280–7287.

Bowden, F. 2000. "A Brief Survey and Synthesis of the Roles of Time in Petri Nets". *Mathematical and Computer Modelling* 21:55–68.

Castillo, S. A. 1989. "Construction of a Formal Methodology to Refine a Spares Suite using TIGER". Master's thesis, Naval Postgraduate School.

Díaz, A., and M. Fu. 1997. "Models for Multi-Echelon Repairable Item Inventory Systems with Limited Repair Capacity". *European Journal of Operational Research* 97:480–492.

Haas, P. J. 2002. *Stochastic Petri Nets. Modelling, Stability, Simulation*. New York: Springer.

Jensen, K. 1993. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, Volume 1. Berlin: Springer.

Law, A., and W. Kelton. 2000. *Simulation Modeling and Analysis*. 3rd ed. New York, NY: McGraw-Hill.

Nelson, B. 2002. *Stochastic Modeling: Analysis & Simulation*. Mineola, NY: Dover Publications.

Roeder, T. M. K. 2004. *An Information Taxonomy for Discrete Event Simulations*. Ph. D. thesis, University of California, Berkeley.

Sherbrooke, C. C. 2004. *Optimal Inventory Modeling of Systems. Multi-Echelon Technigues*. Second ed. New York, NY: Springer.

Volovoi, V. 2013. "Abridged Petri Nets". *ArXiv*:arXiv:1312.2865.

Volovoi, V. 2015. "Simulation with Stochastic Petri Nets. Tutorial". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 88–102. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Volovoi, V. 2016, February. "Big Data for Reliability Engineering: Threat and Opportunity". *IEEE Reliability Magazine*.

Volovoi, V. V. 2004. "Modeling of System Reliability Using Petri Nets with Aging Tokens". *Reliability Engineering and System Safety* 84 (2): 149–161.

AUTHOR BIOGRAPHY

VITALI VOLOVOI is an independent consultant in the area of the Industrial Internet, risk, reliability, and the dynamic interactions of complex systems. He collaborates with Mitek Analytics, consults for Logistics Management Institute, and serves as a member of the NASA Statistical Engineering team. He has led many projects, including turbine analysis for the Air Force, quantitative risk assessment of the Space Shuttle wiring for NASA, air transportation safety analysis for NASA, condition-based maintenance of gas turbines for Siemens, and the reliability of the Hubble Space Telescope gyros for NASA. He has received two NASA Engineering and Safety Group Achievement Awards, a NASA Engineering and Safety Technical Excellence Award, and the Best Tutorial award at the Reliability and Maintenance Symposium (RAMS). He has a Ph.D. in Aerospace Engineering from Georgia Institute of Technology, and a University Diploma in Mechanics and Mathematics from Moscow State University. His e-mail address is vitali@volovoi.com.