Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds.

SUPPORTING A MODELING CONTINUUM IN SCALATION: FROM PREDICTIVE ANALYTICS TO SIMULATION MODELING

John A. Miller
Michael E. Cotterell

Department of Computer Science
University of Georgia
Athens, GA 30602, USA

Stephen J. Buckley

IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA

ABSTRACT

Predictive analytics and simulation modeling are two complementary disciplines that will increasingly be used together in the future. They share in common a focus on predicting how systems, existing or proposed, will function. The predictions may be values of quantifiable metrics or classification of outcomes. Both require collection of data to increase their validity and accuracy. The coming era of big data will be a boon to both and will accelerate the need to use them in conjunction. This paper discusses ways in which the two disciplines have been used together as well as how they can be viewed as belonging to the same modeling continuum. Various modeling techniques from both disciplines are reviewed using a common notation. Finally, examples are given to illustrate these notions.

1 INTRODUCTION

Two disciplines, predictive analytics and simulation modeling, are currently expanding their scopes and are likely to increase their commonalities in the near future. On one hand, predictive analytics attempts to make sense of data by finding patterns or fitting statistical models. On the other hand, simulation modeling attempts to mimic reality. Simulation requires data for fitting distributions and estimating parameters. One may view the two disciplines as two ends of the same continuum. Although somewhat of an overstatement, one end could be described as data-rich and knowledge-poor, while the other could be viewed as knowledge-rich and data-poor. If one looks at the emerging revolution of big data analytics (LaValle et al. 2011), it is only natural to suppose that simulation models will increase their data richness and that models used for deep analytics will increase in their sophistication.

Although analytics is similar to data mining, it often has a stronger emphasis on modeling. Mining has the basic philosophy to examine data with a variety of techniques in order to gain insight. Modeling pushes toward understanding the phenomena, systems or processes involved. There are some internal structures and mechanisms, which may only partially be known, but which are worthwhile to try to capture. Over time, understanding as well as model accuracy (e.g., weather forecasting or biochemical pathway analysis) should improve. Among other things, the refinement and validation of simulation models involves adjusting parameters so that the predictions of the models come in closer agreement with observed data (analogous to what occurs in machine learning). Optimization algorithms are used for both simulation model calibration (parameter adjustment) and machine learning.

In this paper, we examine both predictive analytics and simulation modeling. The characteristics and commonalities of both disciplines are illustrated with example problems. ScalaTion (Miller et al. 2010), an integrated environment supporting predictive analytics, simulation modeling and optimization is used as a testbed in this paper for studying the modeling continuum.

Techniques used in predictive analytics have been used in simulation for metamodeling (Barton 1998; Beers 2005) including techniques such as polynomial regression, neural networks, splines, radial basis functions and kriging. Metamodeling is especially useful when complex and time-consuming simulation models need to be run repeatedly, e.g., for sensitivity analysis or simulation optimization. Such techniques have been used to a lesser degree in simulation for input modeling (Leemis 2004). Analytics can also be used to make simulation optimization more efficient by determining the more important features/factors to consider while optimizing (Better et al. 2007). Finally, analytics can also be used for output analysis. Complex, large-scale simulations can produce result data that are difficult to digest, so use of analytics techniques post simulation can be quite useful.

The rest of the paper is organized as follows: Using a common notation, sections 2 and 3 provide some necessary background on predictive analytics and simulation modeling, respectively. Section 4 makes the case for a modeling continuum based on the richness of data and knowledge utilized by various modeling techniques. Example problems illustrating commonalities and trade-offs between the various techniques are discussed in sections 5 and 6. Finally, conclusions and future work are given in section 7. Due to space limitations, all figures and many code listings are provided in the on-line supplement (see http://www.cs.uga.edu/~jam/scalation/apps/simopt). The supplement also contains an Appendix that provides an overview of ScalaTion.

2 PREDICTIVE ANALYTICS

As the name predictive analytics indicates, the purpose of techniques that fall in this category is to develop models to predict outcomes. For example, the distance a golf ball travels, y, when hit by a driver depends on several factors or inputs, $\mathbf{x}$, such as club head speed, barometric pressure, and smash factor (how square the impact is). The models can be developed using a combination of empirical data and knowledge (e.g., Newton's Second Law). The modeling techniques discussed in this section tend to emphasize the use of data more than knowledge.

Abstractly, a predictive model can generally be formulated using a prediction function, $y = f(\mathbf{x}, t; \mathbf{b})$, where y is a scalar output, $\mathbf{x}$ is an input vector, t is time, and $\mathbf{b}$ is the vector of parameters of the function that can be adjusted so that the predictive model matches available data. Of course, the formulation could be generalized by turning the output into a vector $\mathbf{y}$, the parameters into a matrix, and allowing feedback in the function f. Here, we do not initially consider these generalizations, but introduce them only when necessary.

In ScalaTion, data are passed to the `train` function to train the model/fit the parameters, $\mathbf{b}$. In the case of prediction, the `predict` function is used to predict values for the scalar response y, while the `predictAll` function is used when the response, $\mathbf{y}$, is multidimensional. In the case of classification, the `train` function is still used, but the `classify` and `classifyAll` functions replace the prediction functions. A key question to address is the possible functional forms that f may take, such as the importance of time, the linearity of the function, the domains for y and $\mathbf{x}$, etc. We consider several cases in the subsections below.

2.1 Time-Independent Models

In time-independent models, the time argument, t, is removed from the prediction function. Although these techniques are thought of as time-independent, it is still possible to interpret one of the values in the input vector, $\mathbf{x}$, as time. It is just that time is not a dominate feature as it is in the next section on time-dependent models.

- **Multiple Linear Regression**. Regression has been used with simulation modeling since as early as the 1970's (Kleijnen 1975; Friedman 1984). A common and useful case occurs when it is reasonable to model f as a linear combination of parameters, $\mathbf{b}$. In this case, given an input vector,

x, the predicted value for y is simply the dot product of the parameter vector, **b**, with the input vector augmented with a 1, $[1\ \mathbf{x}]$, as in

$$y\ =\ \mathbf{b}\cdot[1\ \mathbf{x}] = b_0\ +\ b_1x_1\ +\ ...\ +\ b_nx_n.$$

The `predict` function predicts the response value, y, using ScalaTion's dot product operation: `def predict (z: VectorD): Double = b dot z`. Note, $\mathbf{z} = [1\ \mathbf{x}]$. The design of ScalaTion was to make the code look much like the mathematical notation. Using several data samples as a training set, the `Regression` class in ScalaTion can be used to estimate **b**. Each sample pairs an **x** input vector with a y response value. The **x** vectors are placed into a data/design matrix X row-by-row and a column of ones is introduced as the first column in X. The y response values form the response vector. The parameter vector can be estimated from the data using

$$\mathbf{b}\ =\ (X^\intercal X)^{-1}X^\intercal\mathbf{y}.$$

The `train` function below performs this calculation and determines the quality of fit, R^2.

```
def train () {
   b = (x.t * x).inverse * x.t * y       // parameter vector (b0, ... bk)
   val e    = y - x * b                  // residual/error vector
   val sse  = e dot e                    // residual/error sum of squares
   val sst  = (y dot y) - y.sum~^2. / n  // total sum of squares
   rSquared = (sst - sse) / sst          // coefficient of determination
} // train
```

For improved robustness, the `Regression` class also allows the pseudo-inverse to be computed using QR Decomposition. Note, `~^` is the exponentiation operator provided in ScalaTion.

- **Neural Networks**. Neural Networks have been applied in simulation starting in the late 1980's (Fishwick 1989). Regression may involve many possible functional forms that may be hard to assign meaning to, so one could use a more flexible/malleable fitting approach such as using ScalaTion's `NeuralNet` class as shown below. Assuming a three layer network (input, hidden and output layers), an intermediate vector, $\mathbf{h} = sigmoid(W^\intercal\mathbf{x} + \mathbf{w}_b)$, is calculated where W is a weight matrix, **x** is the input vector, $\mathbf{w_b}$ is a bias vector and *sigmoid* is an activation function. The response/output, $\mathbf{y} = sigmoid(V^\intercal\mathbf{h} + \mathbf{v}_b)$, is computed similarly using a second weight matrix V and bias vector $\mathbf{v_b}$. The $W = [w_{ij}]$ matrix indicates the strength of the weight between input x_i and hidden h_j. To this, a bias value, w_{bj}, is added. The V matrix and $\mathbf{v_b}$ vector play the same roles between the hidden and output layers.

```
def predictAll (x: VectorD): VectorD = {
   val h = sigmoid (w.t * x + wb)          // hidden layer
   sigmoid (v.t * h + vb)                  // output layer
} // predictAll
```

2.2 Time-Dependent Models

If time is a key feature involved in the modeling, there are a variety of modeling techniques that can be applied to time series data, starting with the classical ARMA models.

- **Times Series ARMA Models**. Positive results for modeling simulation outputs using time series analysis techniques have been obtained (Brandao and Nova 1999; Brandao and Nova 2003). The

ARMA class in ScalaTion provides support for the development of Auto-Regressive (AR) and Moving Average (MA) models of time series data. An AR(p) model predicts the next value y_t from the last p values each weighted by its own coefficient, ϕ_j The error/noise is represented by ε_t, as in

$$y_t \;=\; \mu \;+\; \phi_1 y_{t-1} \;+\ldots+\; \phi_p y_{t-p} \;+\; \varepsilon_t.$$

The code in ScalaTion works with zero mean data, x_t, where the mean, μ, has been subtracted from y_t. The coefficients, ϕ, are estimated using the Durbin-Levinson algorithm (Rao 2008). After these coefficients are estimated, the AR(p) model can be used for forecasting. An MA(q) model predicts the next value, y_t from the effects of prior noise/disturbances, as in

$$y_t \;=\; \mu \;+\; \varepsilon_t \;+\; \theta_1 \varepsilon_{t-1} \;+\ldots+\; \theta_q \varepsilon_{t-q}.$$

2.3 Models Based on Classifiers

When the output/response, y, is defined on small domains, e.g., $\mathbb{B}$ or $\mathbb{Z}_k = \{0,\ \ldots\ k-1\}$, then some classifiers used in data mining can be used for predictive analytics.

- **Bayesian Networks**. If one can estimate the conditional probabilities of $X_j = x_j$ given $Y = c$, then the BayesClassifier class can used. The best prediction for y is the value that maximizes the product of the conditional probabilities, as shown in

$$y \;=\; argmax_c\{\Pi_{j=0}^{n-1} p(x_j \mid c)\}.$$

  Although the formula assumes the conditional independence of x_js, the technique can be applied as long as correlations are not too high.

3 SIMULATION MODELING

The most recent version of the Discrete-event Modeling Ontology (DeMO) lists five modeling paradigms or world-views for simulation. So far in this paper, discussion has focused on functions with two vectors, the input, $\mathbf{x}$, and output, $\mathbf{y}$, and a scalar time, t. Simulation modeling adds to these the notion of state, represented by a vector-valued function, $\mathbf{s}(t)$. Knowledge about a system or process is used to define state as well as how state can change over time. Theoretically, this should make such models more accurate, more robust, and have more explanatory power. Ultimately, we may still be interested in how inputs affect outputs, but to increase the realism of the model with the hope of improving its accuracy, much attention must be directed in the modeling effort to state and state transitions. This is true to a degree with most simulation modeling paradigms or world views. These paradigms are briefly discussed below and explained in detail in Silver et al. (2011).

- **State-Oriented Models**. State-oriented models, including Generalized Semi-Markov Processes (GSMPs), can be defined using three functions, an activation function, $\{e\} = a(\mathbf{s}(t))$, a clock function, $t' = c(\mathbf{s}(t), e)$, and a state-transition function, $\mathbf{s}(t') = d(\mathbf{s}(t), e)$. In simulation, advancing to the current state, $\mathbf{s}(t)$, causes a set of events, $\{e\}$, to be activated according to the activation function. Events occur instantaneously and may affect both the clock and transition functions. The clock function determines how time advances from t to t' and the state-transition function determines the next state, $\mathbf{s}(t')$. In this paper we tie in the input and output vectors. The input vector, $\mathbf{x}$, is used to initialize a state at some start time, t_0, and the response vector, $\mathbf{y}$, can be a function of the state sampled at multiple times during the execution of the simulation model.

- **Event-Oriented Models**. State-oriented models may become unwieldy when the state-space becomes very large. One option is to focus on state changes that occur by processing events in time order. An event may indicate what other events it causes as well as how it may change state. Essentially, the activation and state transition functions are divided into several simpler functions, one for each event: $\{e\} = a_e(\mathbf{s}(t))$ and $\mathbf{s}(t') = d_e(\mathbf{s}(t))$. Time advance is simplified to just setting the time, t', to the time of the most imminent event on a future event list.
- **Process-Oriented Models**. One of the motivations for process-oriented models is that event-oriented models provide a fragmented view of the system or phenomena. As combinations of low-level events determine behavior, it may be difficult to see the big picture or have an intuitive feel for the behavior. Process-oriented or process-interaction models aggregate events by putting them together to form a process. An example of a process is a customer in a store. As the simulated customer (as an active entity) carries out behavior it will conditionally execute multiple events over time. A simulation then consists of many simultaneously active entities and may be implemented using co-routines (or threads/actors as a more heavyweight alternative). One co-routine for each active entity. The overall state of a simulation is then a combination of the states of each active entity and the global shared state, which may include a variety of resources types.
- **Activity-Oriented Models**. There are many types of activity-oriented models including Petri-Nets and Activity-Cycle Diagrams. The main characteristics of such models is a focus on the notion of activity. An activity (e.g, customer checkout) corresponds to a distinct action that occurs over time and includes a start event and an end event. Activities may be started because time advances to its start time or a triggering condition becomes true. Activities typically involve one or more entities. State information is stored in activities, entities and the global shared state.
- **System Dynamics Models**. System dynamics models were recently added to DeMO, since hybrid models that combine continuous and discrete aspects are becoming more popular. In this section, modeling the flight of a golf ball is reconsidered. Consider the response vector, $\mathbf{y} = [y_0\ y_1]$, where y_0 indicates the horizontal distance traveled and y_1 indicates the vertical height of the ball. Future positions of $\mathbf{y}$ depend on the current position and time. Using Newton's Second Law of Motion, $\mathbf{y}$ can be estimated by solving a system of Ordinary Differential Equations (ODEs) such as $\dot{\mathbf{y}} = f(\mathbf{y}, t)$, where $\mathbf{y}(0) = \mathbf{y_0}$. The `Newtons2nd` object uses the Dormand-Prince ODE solver to solve this problem. More accurate models for estimating how far a golf ball will carry when struck by a driver can be developed based on inputs/factors such as club head speed, spin rate, smash factor, launch angle, dimple patterns, ball compression characteristics, etc. There have been numerous studies of this problem, including Barber (2007).

3.1 Simulation Optimization

Simulation optimization, including both simulation via optimization and optimization via simulation, is becoming more popular and may be used for optimizing designs or for improving the models themselves (Pasupathy and Henderson 2011). One can think of a simulation model as having a parameter vector, $\mathbf{b}$, that needs to be estimated or fit based on pairings of input and output vectors, $\{\mathbf{y}$ and $\mathbf{x}\}$. In some cases, such as biochemical pathways, kinetics parameters are hard to measure directly, so an alternative is to adjust them by using simulation optimization to bring simulation results in line with experimental data. This is analogous to what happens in machine learning, where a training set of data is used calibrate or adjust parameters in a model (e.g., the weights, W and V, in Neural Nets). The optimization techniques themselves may be very similar. ScalaTion supports the development of simulation optimization solutions and includes several optimization algorithms, e.g., Linear Programming (Simplex), Integer Programming (Branch and Bound), Quadratic Programming (Quadratic Simplex), Nonlinear Programming (Steepest Descent, Conjugate Gradient and Quasi-Newton), and Heuristics (Tabu Search and Genetic Algorithm). Furthermore, the SoPT ontology (Han et al. 2011) can assist users developing such solutions.

4 MODELING CONTINUUM

Having examined techniques in both predictive analytics and simulation modeling utilizing a common notation, the paper now considers how these techniques can be viewed to belong to the same continuum.

In the era of *big data* (Jacobs 2009; Agrawal et al. 2011), data for the purposes of analytics or simulation modeling will become available at an explosive rate. Data will come from multiple sources including corporate data stores and on-line data sources such as Web accessible databases, Linked Open Data (LOD) following the Resource Description Framework (RDF) and Web Ontology Language (OWL) ontologies. These are also several initiatives to provide open data (e.g., the Open Data Group and the Open Government Initiative). At the same time, processing capabilities in computing clouds are also increasing substantially. The increasing availability of multi-core compute clusters also provides organizations with more storage and computational power. Programming techniques (e.g., Map-Reduce) and software libraries (e.g., Hadoop, Akka) are making it easier to exploit these computational resources. The convergence of these capabilities, the ability to collect and store vast amounts of data, the ability to carry out deep analytics and the ability to create sophisticated models that are well-calibrated and continually feed by data, suggest a coming revolution in the use, credibility and reliance on these analytics/modeling techniques.

There are numerous modeling techniques that can be used for predictive analytics. Let us consider this in more detail for the modeling of emergency healthcare facilities. The operation and efficiency of an emergency department or urgent care facility is dependent on several factors, such as staffing levels and inventories of supplies. Revenues and costs can also be added to the model. A simulation model can be used to predict profit over time, the expected waiting times of patients, etc. In section 5, we compare three fundamental modeling techniques: one from predictive analytics (Multiple Linear Regression), one analytic technique (Queueing Networks) and one discrete-event simulation techniques (Process-Interaction). All models are developed using ScalaTion.

One difference between the approaches is that the regression/time series analysis are more reliant of data, while the analytic/simulation modeling are more reliant on knowledge. One on hand, if one collects enough data about emergency healthcare facilities, predictions of waiting times and operating costs could be accurately made, so long as the current scenario does not depart too much from prior ones for which data have been collected. On the other hand, armed with the knowledge of how these facilities operate (e.g., queues and service centers), one can construct a simulation model. Still, without data to estimate parameters and fit distributions, which approach will be more accurate? Clearly, effectively combining data and knowledge can lead to more accurate and informative modeling.

In essence, the raw material for both predictive analytics and simulation modeling is data and knowledge. An important trend in the future will be that such data and knowledge will become more widely available. This will allow faster, larger scale modeling, improved reproducibility of results, and enhanced credibility.

The Semantic Web can play a vital role in this mission (Miller and Baramidze 2005) in the following ways: (i) Linked Open Data: An increasing amount of data and meta-data is being made available as LOD (Bizer et al. 2009) (typically in RDF) that is interlinked and available as SPARQL endpoints. (ii) Domain Ontologies: Knowledge of entities, systems or phenomena are being are being captured and organized in OWL ontologies. For example, the GlycO, EnzyO and ReactO ontology suite provides knowledge that can be used in creation of biochemical pathway models. (iii) Modeling Ontologies: The development of ontologies for modeling and simulation (e.g., DeMO and the Cognitive Systems Specification Framework (CS2F)) can serve as a bridge between domain ontologies and executable simulation models (Miller et al. 2004; Douglass and Mittal 2013).

5 APPLICATION TO HEALTHCARE

In the healthcare domain, one problem to be addressed for emergency departments/urgent care facilities is that of staffing. The solutions provided below are simplified to better illustrate the techniques. For more information on problems of this type, please see Tan et al. (2012). Given an estimated demand, how many

of various types of staff members should be hired, i.e., how many triage nurses, registered nurses, nurse practitioners, doctors and administrative clerks should be hired. The model includes $l = 2$ types of patients (regular and severe) and $m = 5$ types of employees.

Table 1: Model definitions.

Variable	Obtained	Description	Units
λ_k	input	arrival rate for patients of type k	hr^{-1}
μ_{jk}	input	service rate at resource j for patients of type k	hr^{-1}
f_k	input	fee charged to patients of type k	\$
d	input	patients dis-utility of waiting	\$ / hr
s_j	input	salary/wage for staff of type j	\$ / hr
x_j	optimize	staffing level for type j employees	none
n	output	treatment rate for patients	hr^{-1}
w	output	waiting time for patients	hr
c	$\mathbf{s} \cdot \mathbf{x}$	operating cost	\$ / hr
r	$\mathbf{f} \cdot \mathbf{n}$	revenue for patient service	\$ / hr
p	$r - c$	net profit	\$ / hr
u	$p - dnw$	overall utility	\$ / hr

The goal is to maximize a utility function based on profit as well as patient satisfaction that factors in a dis-utility proportional to patient waiting times. The optimization problem may be formulated as

$$\textbf{max } u(\mathbf{x}) \textbf{ subject to } \mathbf{x} \in \mathbb{Z}_+^m.$$

This is an Integer Nonlinear Programming Problem (INLP) where, unless assumptions/approximations are made, there is no closed-form expression for the objective function $u(\mathbf{x})$. For this modeling/optimization problem, the following three techniques are utilized: Process-Interaction Simulation Models, Queueing Networks and Multiple Linear Regression.

- **Discrete-event Simulation: Process-Interaction**. The above problem is a natural one to be solved using simulation optimization. This will be done by extending the process-oriented `Model` class provided by ScalaTion. The `ERModel` class (see supplement) defines a simple process-interaction model of our Emergency Department where the staff is represented by instances of the `Resource` class. The input to the model is a `VectorI` which contains staffing levels for the model. The external arrival and service time distributions for each staff member are modeled according to Exponential distributions with rates from λ and μ. Two different kinds of patients are generated in this model, `WalkInPatients` and `AmbulancePatients`. Their arrival distributions are based on the external arrival rates found in λ. Upon arrival, they are assigned a severity level of 0 (low) or 1 (high) based on a Bernoulli distribution where the probability of receiving a 1 is 25%. The model itself is constructed by defining the sources, sinks, resources, queues and transportss for the model as well as defining the script for each actor involved in the simulation (the model's screenshot (Figure 1) and source code are available in the supplement. Given the process-interaction model, we can easily define an objective function $u(\mathbf{x})$ that utilizes the statistics provided the simulation:

```
def u (x: VectorI, s: Double, d: Double): Double = {
    val m = new EmergencyModel(x)        // create model
    val results = m.simulate()           // simulate and gather stats
    val w = m.sumMeanWaitingTimes()      // sum of average waiting times
    val n = new VectorD (m.low, m.high) // patients served of each type
    val c = x dot s                      // operating cost
    val r = f dot n                      // revenue for patient service
```

```
    val p = r - c                          // net profit
    p - d * w * n.mean                     // return overall utility
} // u
```

Note, the `dot` operator is provided by ScalaTion as a concise way to take the dot product over two vectors. Now that the objective function, $u(\mathbf{x})$, has been defined in terms of the actual simulation model, it is ready to be optimized. As this is an INLP optimization problem, ScalaTion's `IntegerTabuSearch` optimizer for Tabu Search (Glover and Laguna 1998) is utilized, passing $u(\mathbf{x})$ as the function to be optimized.

- **Queueing Networks**. Due to the high computational demands of simulation optimization, one may attempt to use a simpler analytic modeling technique, such as Queueing Networks. In ScalaTion, the `JacksonNet` class can be used to determine the steady-state distribution for a network of M/M/c queues. The technique discussed is limited by two serious assumptions. The model only works if the inter-arrival times of patients into the network follow a Poisson process and service times are Exponentially distributed. More general types of Queueing Networks (e.g., BCMP) models are planned for the future. Still, they will have limiting assumptions compared to simulation models. The derived queueing model for the Emergency Department model can be found in the supplement. The objective function is defined using the steady-state results and is analogous to the one given in the last section.
- **Multiple Linear Regression**. Several factors can affect the staffing decisions, including the hourly pay rates for the various types of staff members, the costs of supplies, etc. We use Multiple Linear Regression (MLR) to predict patient waiting times, cost, revenue, profit and utility. The two techniques discussed above, have limitations: Discrete-event Simulation used in simulation optimization can be very time consuming, even when run on a cluster. Queueing Networks provide solutions more rapidly, but their accuracy can be questioned when their assumptions are violated. An important issue for MLR is the issue of getting the data; recall the earlier discussion of data-rich, knowledge-poor vs. data-poor, knowledge-rich. Clearly, it is important to collect real-world data, even simulation models need data. The data requirements for applying Multiple Linear Regression are much greater. Of course, there is no reason why simulation can not supply the regression models with some data, as is done in Response Surface Methodology (Carley et al. 2004). The regression models can therefore be used for interpolation, limited extrapolation and optimization, reducing the need for more time-consuming simulation runs.

6 APPLICATION TO SUPPLY CHAIN MANAGEMENT

In this paper we claim that predictive analytics is more reliant on data, while simulation modeling is more reliant on knowledge. Effectively combining data and knowledge can lead to more accurate and informative modeling. Supply chain management is one of the most mature fields of analytics and provides excellent support for our claims. A wide variety of time-dependent predictive analytics techniques are used in supply chain management to forecast product demand (Box and Jenkins 1976). As shown in Figure 2 (see supplement), forecasts of product demand feed the overall supply chain process, whose goal is to provide inventory to satisfy demand on a continuing basis. Simulation is often used to assess whether a supply chain will truly satisfy demand in the presence of a variety of uncertainties such as forecast error, supplier lead time, manufacturing lead time, and manufacturing yield. Here are a few of the many examples of supply chain simulation:

- **IBM Europe PC Study**: In the mid-1990s, IBM performed simulation modeling to develop a better understanding of its personal computer supply chain in Europe (Feigin et al. 1996). The supply chain was experiencing low service levels and excessive inventory. A manufacturing plant in Scotland was the primary source for PCs. The manufacturing execution strategy was Build To

Forecast and there were distribution centers and transshipment points in each European country. A discrete-event simulation model was built, along with a number of optimization models.

- **IBM Asset Management Tool (AMT)**: A couple of years later IBM started a global supply chain re-engineering effort whose goal was to achieve quick customer responsiveness with minimal inventory in the IBM global supply chain. To support this effort, an extended enterprise supply chain analysis tool, AMT, was developed (Lin et al. 2000). AMT integrated graphical process modeling, analytical performance optimization, and discrete event simulation to study a wide range of issues including inventory budgets, turnover objectives, customer service level targets, and effects of new product introduction.
- **IBM Pandemic Business Impact Modeler**: In 2006, IBM developed a simulation model to understand how a pandemic might impact a manufacturing companys employees and business performance (Chen-Ritzo et al. 2007). As shown in Figure 3 (see supplement), the model comprised six integrated sub-models to examine epidemiological, behavioral, economic, infrastructure, value chain, and financial aspects of the IBM ecosystem. The sub-models were constructed using a combination of system dynamics simulation, time step simulation, and linear programming.

In the above examples, predictive analytics and simulation modeling are used in combination to predict important supply chain results. There is a key difference in how uncertainty is treated in the two techniques. In the predictive analytics example, data are analyzed to generate a forecasted quantity, with uncertainty in the forecast expressed as a probability distribution (generally referred to as forecast error). The simulation models accept probability distributions as inputs, draw samples from the distributions, and assess the impact of the samples on the modeled processes. The simulation models generate an array of output values, which can be converted back into probability distributions using output analysis. In essence, predictive analytics turns data into distributions, while simulation modeling turns distributions into data.

The parameters of our predictive model must be trained on historical data in order to create a reasonably accurate forecasting engine. Similarly, the parameters of our simulation models must be calibrated against historical data in order to create "valid models" that reliably predict demand satisfaction. After validation, it is common to perturb a supply chain simulation model using what-if analysis by changing parameters, policies, processes, or network structures in an attempt to improve the ability of the supply chain to satisfy demand. At this point, supply chain simulation has gone beyond being a pure predictor and has become a tool to improve the supply chain. This was in fact the purpose of the simulation models described above, whose results were as follows:

- **IBM Europe PC Study**: $40M of savings were realized per year by removing many distribution centers and transshipment points, and changing the manufacturing execution strategy from Build To Forecast to a variant of Build To Order (see Figure 4 in supplement).
- **IBM Asset Management Tool (AMT)**: AMT was implemented in a number of IBM business units and their channel partners. Benefits generated by AMT included more than $750M of savings in material costs and price protection expenses in 1998, leading to IBM receiving the 1999 Franz Edelman award.
- **IBM Pandemic Business Impact Modeler**: The results of the model suggested that closing airports and establishing alternative suppliers and fulfillment sites may be the most effective combination for mitigating the impact of a pandemic on the revenue of a manufacturer, subject to a number of assumptions about the manufacturer.

Predictive analytics can also be extended to improve a supply chain. A forecasting engine operates by identifying the relationship between demand and selected demand drivers such as historical demand, weather, and prices. While historical demand and weather cannot be changed, prices can. Therefore, a demand forecasting engine can be used to improve a supply chain by leveraging its model of price elasticity to project price recommendations that increase demand (Dietrich et al. 2012).

A recent proof of concept built by IBM provides a second example of using predictive analytics to improve a supply chain. The goal was to improve the on-time delivery of a group of consumer products from a manufacturing site to a set of retail distribution centers. The analysis involved monitoring a large set of metrics across the supply chain over time and correlating exceptions in those metrics to late deliveries at the retail distribution centers, using a predictive analytic technique called Temporal Causal Modeling (Arnold et al. 2007). This regression based technique extends traditional mining of causality beyond statistical correlations to temporal relationships. The proof of concept identified five metrics, including forecast accuracy and distribution center capacity, which had the most impact on late deliveries. Improving these five metrics indicated a potential to significantly improve on-time delivery. However, the predictive model did not indicate how to improve the five metrics in supply chain operations. The next step in improving this supply chain may be to create a detailed simulation model focusing on the processes impacting the five most important metrics. In this way, the two techniques can work together to efficiently model and improve a supply chain.

A third example of using predictive analytics to improve a supply chain is an IBM tool called Quality Early Warning System (IBM 2012). This tool was developed for early identification of quality issues with a low rate of false alarms. It automatically predicts defect trends before such trends can be triggered by traditional, industry standard Statistical Process Control techniques. It is used by IBM in its manufacturing, development, procurement, and field warranty processes. Like the second example, this tool can be used to identify processes that need to be modeled in more detail to enable improved results. The tool saved IBM over $38M in quality related costs since its deployment in 2008.

In summary, the examples presented prove that predictive analytics and simulation modeling are important techniques for supply chain management. Both techniques can be used for prediction, one relying purely on the data and the other on a combination of knowledge and data. Both techniques go beyond pure prediction to enabling improvements in a modeled process through what-if analysis. Predictive models focus mainly on metrics and have little knowledge of underlying processes. Simulation models rely on deep knowledge about processes and provide a more detailed and flexible way to evaluate potential process changes. Together these technologies provide an efficient methodology to determine the focus areas of a process and analyze potential improvements to the process.

7 CONCLUSIONS AND FUTURE WORK

Analytics and modeling is a vast landscape with numerous competing and complementary techniques. Positioning these techniques along a modeling continuum as well as creating taxonomies and ontologies to describe and inter-relate them can help illuminate this vast landscape. From the available knowledge, data and purpose of a particular modeling study, the appropriate techniques can be chosen from the modeling continuum.

The ScalaTion package, which provides a wide selection of techniques for analytics and modeling, can be used as a testbed for exploring trade-offs along the modeling continuum. The examples given in this paper represents a preliminary study of some of the trade-offs. An integrated environment like ScalaTion supporting a modeling continuum would clearly be advantageous for supply chain management professionals. The development of models using ScalaTion can be assisted by utilizing the DeMO and SoPT ontologies. Models can be created manually, with GUI designers, or generated from instances in ontologies. Example code generators are available on the Web (see supplement).

With such technologies in place, an additional issue becomes how to take Web accessible knowledge and big data and assist modelers and/or domain specialists in creating predictive models. Preliminary work has begun on this with the DeMOforge project. To handle larger scale predictive analytics, simulation models and simulation optimization, work has begun to parallelize ScalaTion to run on multi-core clusters. Additional future work includes developing an ontology to support predictive analytics, handling of multiple ontologies, searching for relevant ontologies, using rules to suggest modeling techniques relevant to a particular problem, and searching open linked data and on-line databases for additional information.

REFERENCES

Agrawal, D., S. Das, and A. E. Abbadi. 2011. "Big Data and Cloud Computing: Current State and Future Opportunities". In *Proceedings of the 14th Int. Conference on Extending Database Technology*, 550–533.

Arnold, A., Y. Liu, and N. Abe. 2007. "Temporal Causal Modeling with Graphical Granger Methods". In *Proceedings of the 13th Int. Conference on Knowledge Discovery and Data Mining*, 66–75.

Barber, R. 2007. "Golf Ball Flight Dynamics". Technical report, Cornell University.

Barton, R. R. 1998. "Simulation Metamodels". In *Proceedings of the 1998 Winter Simulation Conference*, edited by D. Medeiros, E. Watson, J. Carson, and M. Manivannan, 167–174. Piscataway, New Jersey: IEEE.

Beers, W. C. V. 2005. "Kriging Metamodeling in Discrete-Event Simulation: An Overview". In *Proceedings of the 2005 Winter Simulation Conference*, edited by M. E. Kuhl, N. M. Steiger, F. B. Armstrong, , and J. A. Joines, 202–208. Piscataway, New Jersey: IEEE.

Better, M., F. Glover, and M. Laguna. 2007. "Advances in Analytics: Integrating Dynamic Data Mining with Simulation Optimization". *IBM Journal of Research and Development* 51 (3/4): 477–487.

Bizer, C., T. Heath, and T. Berners-Lee. 2009. "Linked Data–The Story So Far". *International Journal on Semantic Web and Information Systems (IJSWIS)* 5 (3): 1–22.

Box, G., and G. Jenkins. 1976. *Time Series Analysis: Forecasting and Control*. Upper Saddle River, NJ, USA: Holden-Day.

Brandao, R. M., and A. P. Nova. 1999. "Experimental Evaluation of Methods for Simulation Output Analysis". In *Proceedings of the 1999 European Simulation Symposium*, 601–607.

Brandao, R. M., and A. P. Nova. 2003. "Non-Stationary Queue Simulation Analysis Using Time Series Analysis". In *Proceedings of the 2003 Winter Simulation Conference*, edited by S. Chick, P. J. Sànchez, D. Ferrin, and D. J. Morrice, 408–413. Piscataway, New Jersey: IEEE.

Carley, K. M., N. Y. Kamneva, and J. Reminga. 2004. "Response Surface Methodology". Technical report, DTIC Document.

Chen-Ritzo, C., L. An, S. Buckley, P. Chowdhary, T. Ervolina, N. Lamba, Y. Lee, and D. Subramanian. 2007. "Pandemic Business Impact Modeler". In *Proceedings of the 2007 INFORMS Simulation Society Research Workshop*.

Dietrich, B., M. Ettl, R. Lederman, and M. Petrik. 2012, July. "Optimizing the End-to-End Value Chain through Demand Shaping and Advanced Customer Analytics". In *Proceedings of the 11th International Symposium on Process Systems Engineering*, 8–18.

Douglass, S. A., and S. Mittal. 2013. "A Framework for Modeling and Simulation of the Artificial". In *Ontology, Epistemology, and Teleology for Modeling and Simulation*, 271–317. New York: Springer.

Feigin, G., C. An, D. Connors, and I. Crawford. 1996. "Shape Up, Ship Out". *ORMS Today* 23 (2): 24–30.

Fishwick, P. A. 1989. "Neural Network Models in Simulation: A Comparison with Traditional Modeling Approaches". In *Proceedings of the 1989 Winter Simulation Conference*, edited by E. MacNair, K. Musselman, and P. Heidelberger, 702–709. Piscataway, New Jersey: IEEE.

Friedman, L. W. 1984. "Establishing Functional Relationships in Multiple Response Simulation". In *Proceedings of the 1984 Winter Simulation Conference*, edited by S. Sheppard, U. Pooch, and D. Pegden, 285–289. Piscataway, New Jersey: IEEE.

Glover, F., and M. Laguna. 1998. *Tabu Search*, Volume 1. Kluwer Academic Pub.

Han, J., J. A. Miller, and G. A. Silver. 2011. "SoPT: Ontology for Simulation Optimization for Scientific Experiments". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 2914–2925. Piscataway, New Jersey: IEEE.

IBM 2012. "IBM Quality Early Warning System". Technical report.

Jacobs, A. 2009. "The Pathologies of Big Data". *Communications of the ACM (CACM)* 52 (8): 36–44.

Kleijnen, J. 1975. "Metamodel for Sensitivity Analysis: The Regression Metamodel in Simulation". *Interfaces* 5 (3): 21–23.

LaValle, S., E. Lesser, R. Shockley, M. S. Hopkins, and N. Kruschwitz. 2011. "Big Data, Analytics and the Path From Insight to Value". Technical report, IBM.

Leemis, L. M. 2004. "Building Credible Input Models". In *Proceedings of the 2004 Winter Simulation Conference*, edited by R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 29–40. Piscataway, New Jersey: IEEE.

Lin, G., M. Ettl, S. Buckley, S. Bagchi, D. Yao, B. Naccarato, R. Allan, K. Kim, and L. Koenig. 2000. "Extended-Enterprise Supply-Chain Management at IBM Personal Systems Group and Other Divisions". *Interfaces* 30 (1): 7–25.

Miller, J. A., and G. Baramidze. 2005. "Simulation and the Semantic Web". In *Proceedings of the 2005 Winter Simulation Conference*, edited by M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 2371–2377. Piscataway, New Jersey: IEEE.

Miller, J. A., G. T. Baramidze, A. P. Sheth, and P. A. Fishwick. 2004. "Investigating Ontologies for Simulation Modeling". In *Proceedings of the 37th Annual Simulation Symposium*, ANSS '04, 55–63. Washington, DC, USA: IEEE Computer Society.

Miller, J. A., J. Han, and M. Hybinette. 2010. "Using Domain Specific Languages for Modeling and Simulation: ScalaTion as a Case Study". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 741–752. Piscataway, New Jersey: IEEE.

Pasupathy, R., and S. Henderson. 2011. "SimOpt: A Library of Simulation Optimization Problems". In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 4075–4085. IEEE.

Rao, S. S. 2008. "A Course in Time Series Analysis". Technical report, Texas A&M University.

Silver, G. A., J. A. Miller, M. Hybinette, G. Baramidze, and W. S. York. 2011. "DeMO: An Ontology for Discrete-event Modeling and Simulation". *Simulation* 87 (9): 747–773.

Tan, W.-C., P. J. Haas, R. L. Mak, C. A. Kieliszewski, P. G. Selinger, P. P. Maglio, S. Glissman, M. Cefkin, and Y. Li. 2012. "Splash: A Platform for Analysis and Simulation of Health". In *Proceedings of the 2nd ACM SIGHIT Symposium on International Health Informatics*, 543–552. ACM.

AUTHOR BIOGRAPHIES

JOHN A. MILLER is a Professor of Computer Science at the University of Georgia. His research interests include database systems, simulation, Web services and bioinformatics. Dr. Miller received the B.S. degree in Applied Mathematics from Northwestern University in 1980 and the M.S. and Ph.D. in Information and Computer Science from the Georgia Institute of Technology in 1982 and 1986, respectively. During his undergraduate education, he worked as a programmer at the Princeton Plasma Physics Laboratory. He is an Associate Editor for the ACM TOMACS, IEEE TSMC and SCS SIMULATION as well as an Editorial Board Member for the JOS and International Journal of SPM. His email address is jam@cs.uga.edu.

MICHAEL E. COTTERELL is a Ph.D. student in Computer Science at the University of Georgia, where he also received the B.S. degree in Computer Science in May 2011. As an undergraduate, he served as Vice Chairman of the UGA ACM chapter. Currently, he serves as the 2013–2014 Faculty Liaison for the UGA Computer Science Graduate Student Association. His research interests include Simulation, Optimization, Semantic Web and Domain-Specific Languages with inter-disciplinary applications in Analytics and Informatics. His email address is mepcotterell@gmail.com.

STEPHEN J. BUCKLEY has been a Research Staff Member at the IBM Thomas J. Watson Research Center in Yorktown Heights, NY since 1987. His research interests include business analytics, simulation, and supply chain management. At present his responsibilities include the management of a worldwide portfolio of analytic solutions and sales support for related services engagements. He received the Ph.D. degree in Computer Science from MIT in 1987 and the M.S. degree in Computer Science from Penn State University in 1978. His email address is sbuckley@us.ibm.com.