Modeling Uncertainty for Middleware-based Streaming Power Grid Applications

Ilge Akkaya
Electrical Engineering and Computer Science
University of California, Berkeley
ilgea@eecs.berkeley.edu

Yan Liu
Faculty of Engineering and Computer Science
Concordia University
Montreal, QC, Canada
yan.liu@concordia.ca

Edward A. Lee
Electrical Engineering and Computer Science
University of California, Berkeley
eal@eecs.berkeley.edu

Ian Gorton [*]
Pacific Northwest National Laboratory
Richland, WA, USA
ian.gorton@pnnl.gov

ABSTRACT

The power grid is incorporating high throughput sensor devices into power distribution networks. The future power grid needs to guarantee accuracy and responsiveness of applications that consume data from multiple sensor streams. The end-to-end performance and overall scalability of cyber-physical energy applications depend on the middleware's ability to handle multi-source sensor data, which exhibits uncertain behavior under highly variable numbers of sensors and middleware topologies. In this paper, we present a parametric approach to model middleware uncertainty and to analyze its effect on distributed power applications. The models encapsulate the entire data flow paths from sensor devices, through network and middleware components to the power application nodes that utilize sensor data streams. Using the Ptolemy II framework for modeling and simulation, we generate Monte Carlo samples of uncertain parameters that are used to generate parameterized middleware models that are used in end-to-end Discrete-Event(DE) system simulation simulation. The simulation results are further analyzed using regression methods to reveal the parameters that are influential in the limiting middleware behavior to achieve temporal requirements of the power applications.

Categories and Subject Descriptors

I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Discrete event, distributed, Monte Carlo*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*; D.2.11 [**Software Engineering**]: Software Architectures—*domain-specific architectures*

General Terms

Performance, Design

Keywords

uncertainty, simulation, middleware, power applications

[*]This work was done while at Pacific Northwest National Laboratory

1. INTRODUCTION

Power grid operators rely on data to make real-time control decisions. With high-fidelity and trustworthy phasor sensor data from hundreds of thousands of measurement data points, the future electric power grid will enable the use of numerous advanced and innovative distributed control methods. Phasor Measurement Units (PMUs) provide real-time and precisely time stamped grid measurements, which is a feature that enables the measurements to be time-aligned and merged at aggregation nodes for a better indication of grid stress. The challenge of combining phasor measurements for distributed power applications has been investigated by power engineers and researchers [5,7,8]. From a distributed cyber-physical system perspective, the challenge is to autonomously coordinate the data flow and access within distributed power systems [10].

In the distributed systems architecture, the entire power system is decomposed into non-overlapping subsystems that are connected via tie lines (buses). Each subsystem, usually corresponding to a balancing authority or a control center of a power utility, is able to run a local application and then exchange data with neighboring substations for coordination. Middleware that mediates data exchange between partitions of the grid is an integral to the communication infrastructure. Local results from each area are transferred into the middleware to be aggregated and time-aligned. The middleware also merges the time-aligned faulty phasor readings of the entire system, and broadcasts this information to all remote participants for global situational awareness [10]. An additional decision component, which is also part of the middleware, receives these intermediate results and notifies each area when global convergence of power system states has been achieved.

In this middleware centric architecture, a number of factors such as the number of iterations required for data exchange prior to global convergence, sensor data quality, the amount of sensors, and the grid partitioning affect the end-to-end response time of the power application. A systematic quantification of these factors gain importance in the overall and worst-case performance evaluation of the middleware.

In this paper, we present a model-based approach to specifying uncertain parameters in the middleware architecture. We develop models that represent the entire data flow from sensor devices, through network and middleware components to distributed application nodes. We use the Ptolemy II framework both to derive Monte Carlo (MC) samples of the uncertain parameters and to generate and execute DE models of the system with the sampled values of these parameters. The modeling approach for domain-specific components in the power application is presented in Section 2, followed by a detailed specification of the end-to-end discrete-event power system models, parameterization and sampling methods in Section 3. Finally, uncertainty analysis using regression techniques and evaluation of the significant parameters for robust middleware design are studied in Section 4.

2. MODELING APPROACH

The initial problem we study is the uncertainty modeling workflow for power grid application middleware. Figure 1 summarizes the followed methodology. This method aims at deriving the most significant uncertainty parameters from the systems architecture as random variables. Under the circumstances of a set of possible correlated variables, mathematical procedures such as principle component analysis (PCA) can be applied to convert correlated variables into a set of uncorrelated variables. Samples are generated by the Monte Carlo (MC) sampling method.

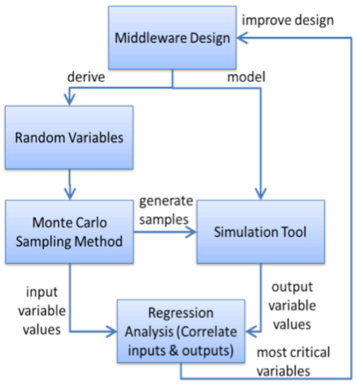

Figure 1: Middleware design guided by uncertainty analysis

For accurate representation of timing in the system, it is required to model the entire data flow through the systems entities including sensors, network and middleware components. Communication delays, software scheduling and timing properties are also considerations of the modeling process. We use the Ptolemy II framework for modeling and simulation purposes [4]. Ptolemy II is a heterogeneous actor-oriented design tool that provides an integrated modeling and simulation environment for the conceptual procedure in Figure 1. Ptolemy has a long history in Cyber-Physical Systems (CPS) modeling and is a robust modeling tool, since it enables heterogeneous compositions of models of computation, which is an essential feature for modeling energy systems that intrinsically include interactions of physical components and communication fabric [3, 6, 9].

3. MODELING SYSTEM ARCHITECTURE

We first model the top level systems architecture and then develop a hierarchical model to concentrate on middleware coordination.

3.1 Top Level Model

In Ptolemy, actors are components that communicate via ports. Using Ptolemy II, the high-throughput devices communicating over the network with packets can be abstracted to discrete-event (DE) components communicating via time stamped events, where each network packet is represented by a discrete event in the Ptolemy execution. DE execution is based on events composed of a token (value) and a tag (time stamp). The discrete event scheduler guarantees that events are processed in time stamp order. More precisely, actors that have the earliest time stamp input events are executed first. In a DE model, all actors share a global notion of time, namely the *model time* at the particular level of hierarchy. This imposes a global ordering of events within the model and therefore ensures determinacy. The modeling details of the domain-specific system entities are presented below.

Modeling PMUs. Phasor Measurement Units are the main sources of phasor data, abstracted as DE events as they are represented in the Ptolemy execution. We collectively model the PMUs residing within a local area in the grid as a cluster and denote this component as a `PMUCluster_i`, where i refers to the area number. This actor is parameterized by the `PMUCount` parameter and generates the corresponding number of events every iteration interval (parameterized by the `PMUPeriod` parameter). PMUCount is one of the inputs to the executable model, which in the outer hierarchy, sampled from a user-defined prior distribution.

Modeling Phasor Data Concentrators (PDCs). A PDC is responsible for receiving data from multiple PMUs and producing an aggregated data packet for each PMU at an application specific rate. We only model the relaying function of the PDC, where it produces data packets for each PMU and processes the packets in a FIFO pattern.

Modeling Distributed Power Applications. Each distributed power application is locally run on a computation cluster and is expected to consist of multiple iterations, interleaved with peer-to-peer communications with the coordinating areas. We use a generic computation node model that accepts SCADA and PDC packets and produces intermediate packets after each algorithm iteration. The passage of time is characterized by a stochastic delay that allows the user to tie a stochastic profile with the algorithm execution time at each iteration.

The top-level model is illustrated in Figure 2. In this architecture, the PMUs of each area produce data at 30 samples per second that are transmitted to the local PDC. Then data are relayed at the PDC and sent to the local computation node of each area to be utilized in the periodic power applications. We assume these deliveries use network connection links specified by North American SynchroPhasor Initiative (NASPI) [https://www.naspi.org]. In this case, each PMU is connected to the local PDC by a 56Kbps communication line. The packet size of each PMU message is assumed to be 128 bytes in compliance with the data format

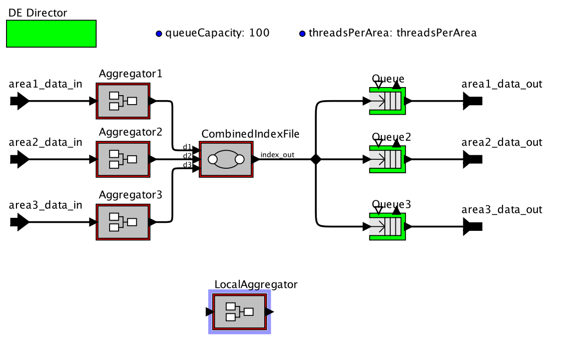

Figure 3: Middleware Aggregation Top Level Model

of IEEE C37.118 [1]. Each of these links is chosen to be 50 miles long, representing an average physical distance from a bus to the nearest PDC. We assume each PDC is placed 300 miles away from the local area and the middleware component is symmetrically located, at a maximum distance of 350 miles from the PDCs.

3.2 Middleware Model

Middleware models represent the data exchange behavior of the distributed power applications. We consider two main roles of the middleware function:

Aggregation. Middleware is responsible for receiving packets from the PMUs of all distributed areas and aggregating the packets into a universal index file. The functional model of the top-level middleware aggregator is given in Figure 3. PMU streams from each area are first taken into a local aggregator (named Aggregator in the Ptolemy model), where they are processed on a per-file basis. A synchronization unit, called CombinedIndexFile then aggregates the streams from all areas into one global output packet.

Convergence Control. Another important role of the middleware simulation is to determine whether a distributed power application has reached a global convergence state. For DSE and similar iterative power applications, the number of iterations until global convergence relies on several parameters such as PMU data redundancy and quality. To be comprehensive, we assume a variable number of iterations, ranging in the discrete set $\{1, 2, ..., 20\}$ until convergence has been reached.

3.3 Model Calibration

For calibrating the model to accurately represent the distribution of packet processing times, we use the data obtained from benchmark results carried out on the Apache ActiveMQ message broker that implements the aggregation and queuing behavior of the middleware. Following the distributions for the benchmarked number of sensor streams per area, we carry out a regression analysis on the time series obtained by the benchmarks to yield a fit estimate for the middleware processing overhead distribution. The maximum-likelihood distribution fit for the benchmarked data range is given by a Rice distribution parameterized as follows:

$$\begin{aligned} T_i &\sim Rice(\nu(N_{PMU_i}), \sigma(N_{PMU_i})) \\ \nu(N_{PMU_i}) &= 0.0302 \log(N_{PMU_i}) + 0.055 \\ \sigma(N_{PMU_i}) &= 0.0007 * N_{PMU_i} + 0.0414 \end{aligned} \tag{1}$$

where T_i is the random variable that denotes the processing time in seconds of a PMU stream associated with Area i, *Rice*() denotes a Rician distribution with parameters ν and σ, and N_{PMU_i} is the number of PMU streams delivered to Area i.

Figure 4 demonstrates a sample histogram for the distributions of per-file middleware processing times, for 250 concurrent PMU files. The Rician distribution fitting is then used in the local aggregators, denoted as `Aggregator_i` in Figure 3.

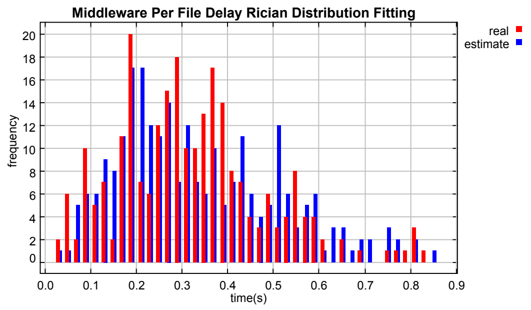

Figure 4: Histograms for simulation generated (estimate) and experimentally obtained (actual) middleware delay

To further simulate the concurrent queue architecture, we make use of the Ptolemy actor called `MultiInstanceComposite`, which is capable of generating a user-defined number of copies of a component and to run the instances concurrently. The local aggregator model allows each received PMU stream to be randomly assigned to one of the processing unit instances within a thread pool, each simulating a server with stochastic latency that follows (1). The outputs of all instances are then merged to yield a stream containing all the processed PMU streams for this particular area.

The top level model and the middleware models are governed by a *Director*, which implements a model of computation. Passage of time at each compositional level is governed by a local clock that enables the user to define different clock rates, clock drifts and each local clock's relation to real time, if desired, for further time-alignment simulation.

3.4 Monte Carlo Sampling

Smart Grid topologies are expected to be highly variable in the number of PMUs per area and the inter-area network characteristics. Monte-Carlo simulations integrated with the distribution system topology is a useful tool to evaluate the effect of certain model parameters on middleware performance. Figure 5 displays the top level Ptolemy model that is used to generate Monte Carlo samples of selected model parameters, explained in detail in Table 1. Here, the parameters are assumed to be uniformly distributed over a finite set of integers to avoid any bias towards a particular fashion for the PMU distribution. However, it should be noted that the number of iterations may follow a Gaussian-like distribution in practice.

In this paper, for a power system involving three distributed areas, we are interested in the 5-tuple `<PMU_Count_1, PMU_Count_2, PMU_Count_3, concurrencyLevel, numberOfIterations>` samples of the parameters. The simulation is run for 6000 seconds in model time for each sample tuple, roughly corresponding to 100 complete runs of an power application, in the specific case of the DSE, assuming each run of distributed state estimation needs to complete within 60

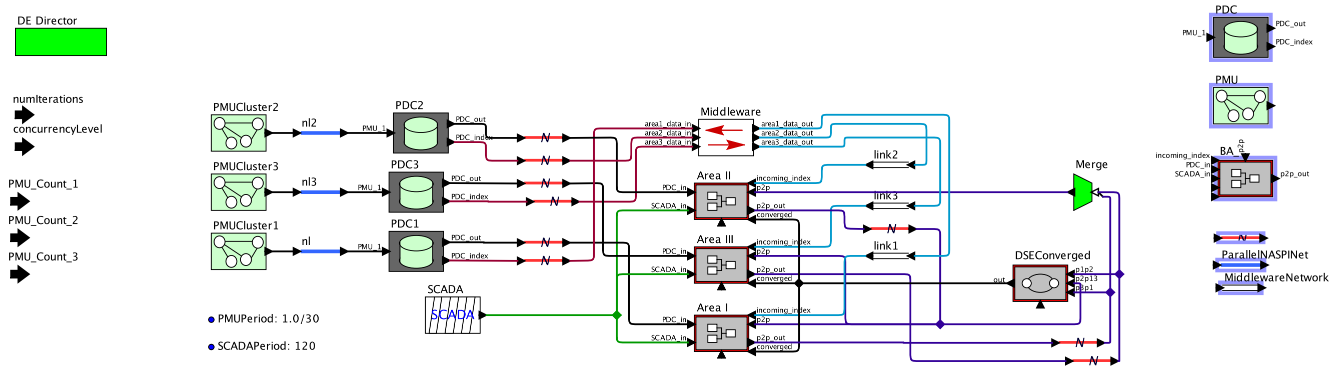

Figure 2: Overall System Architecture Simulation Model

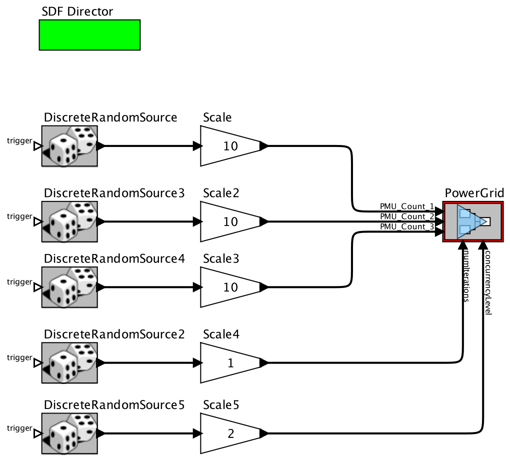

Figure 5: Monte Carlo sampling in Ptolemy

Table 1: Monte Carlo Variables and Respective Probability Mass Functions (range format:initial:increment:final)

Parameter Name	PMF	Range
PMU_Count_1	Uniform	10:10:500
PMU_Count_2	Uniform	10:10:500
PMU_Count_3	Uniform	10:10:500
concurrencyLevel	Uniform	2:2:20
numberOfIterations	Uniform	1:1:20

seconds. The average and maximum run time per run is recorded in addition to the Monte-Carlo parameter values.

4. UNCERTAINTY ANALYSIS

Following data collection using Monte-Carlo methods, we perform polynomial regression analyses to account for the effect and significance of the model parameters on the maximum end-to-end run time. We define the variables to be used in the regression analysis on Table 2.

The initial question addressed is the influence of the middleware concurrency level and the maximum number of PMUs per area on the end-to-end Distributed State Estimation (DSE) runtime. The polynomial curve fitting methods that use x_1 and x_2 as independent variables and y as the dependent variable that are run up to order 3 for each explanatory variable reveal that a bivariate polynomial regression equation that is cubic in x_1 and quadratic in x_2 is the best-fitting model in the studied set of polynomial fits. The best fit is evaluated in terms of highest Bayesian Information Criterion (BIC) and lowest sum-of-squares error (SSE) among the set of fits.

Table 2: Variables for Regression Analysis

Variable	Explanation
x_1	concurrency level
x_2	max`{PMU_Count_1,PMU_Count_2,PMU_Count_3}`
x_3	number of iterations
y	maximum end-to-end runtime

Table 3: Polynomial Regression Coefficient Estimates

Coefficient	Estimate	Confidence Interval
$Intercept$	18.32	[14.73, 21.93]
x_1	-6.9699	[-7.47, -6.46]
x_2	0.14869	[0.13, 0.17]
x_1^2	0.73883	[0.70, 0.78]
x_1x_2	-0.01544	[-0.02, -0.01]
x_1^3	-0.02236	[-0.02, -0.02]

Table 3 presents the best estimates of the polynomial coefficients for the regression fit, together with the confidence intervals for the coefficients. The p-value, which indicates the false alarm probability for the variable being estimated, is less than 1E-6 for all the variables taken into account. This provides high confidence that the regression model is accurate and avoids over-fitting.

The goodness of fit metrics presented in Table 3 further confirm that the variables chosen for Monte Carlo simulation sufficiently relate to the trend in the runtime distribution. The R^2 value indicates that the independent variables x_1 and x_2 account for explaining 96.8% of the observed data. Moreover, the fit yields the highest BIC among all bivariate fits of up to order 3. Since AIC and BIC are goodness of fit metrics that establish a tradeoff between model accuracy and complexity, maximizing these while minimizing the fit error of the model helps in avoiding parameter over fitting.

The regression analysis that best fits obtained simulated outputs with the input data set is given in Figure 6 with the 95% confidence bounds. The concurrency level of at

Table 4: Goodness of Fit for Polynomial Regression Analysis

SSE	R^2	RMSE	AIC	BIC
2519	0.968	1.784	3209	3251

least 10 is necessary for the middleware to scale in handling increasing number of PMUs. The response surface has linear trend as the number of PMU increases, but remains planar as the concurrency level increases. This indicates further increasing the concurrency level from 10 to 20 has marginal benefit.

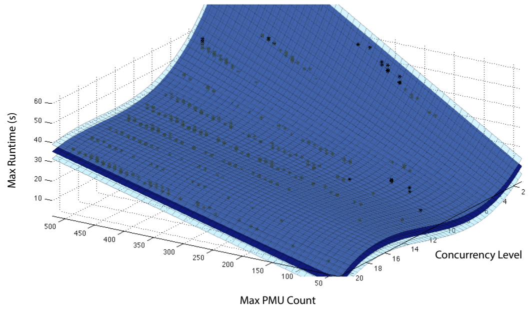

Figure 6: Polynomial Regression Analyses of Significant Monte Carlo Variables

Having shown a significant relationship between end-to-end maximum runtime and the model parameters middleware concurrency the maximum number of PMUs per area, we move on to investigate the effect of less significant model variables. The additional polynomial regression analysis includes the number of iterations (x_3) as the third explanatory variable candidate. The notation follows from the previous analysis.

The best fit is selected by evaluating the Akaike Information Criterion(AIC) and BIC on a set of polynomial fits up to order 3 in each explanatory variable. AIC favors the model order of $\{3, 2, 2\}$ and BIC favors the model of order $\{3, 2, 3\}$, respectively, in $\{x_1, x_2, x_3\}$. The AIC-optimal polynomial fit is chosen, and the results are presented in Table 5. The significant coefficient estimates are given, and the higher-order parameters with confidence intervals centered around zero are omitted from the result.

Comparison of Tables 4 and 6 reveal that, incorporating x_3 in the regression analysis has little improvement on the overall fit. Moreover, the coefficient estimates given in Table 5 that depend on x_3 are either numerically insignificant or have confidence bounds that intersect 0, suggesting that the coefficient values most likely do not express a significant explanatory relation to the observed variable y. This result provides more confidence that the number of iterations is a less significant variable that influences the maximum run time of the distributed state estimation algorithm, compared to the concurrency level and the maximum number of PMU streams per area.

To explain the insignificance of number of DSE iterations in the analysis, we refer to the system model presented in section 3. As only intermediate estimate data on tie-line buses are exchanged, the data communication load is only approximately 65K bytes per iteration. Even for a larger scale power system such as the Western Electricity Coordinating Council (WECC) with more than 1300 buses and 6700 loads, the data exchange between neighboring areas would remain to be a relatively small overhead and would have little contribution to the end-to-end algorithm runtime.

Table 5: Extended Polynomial Regression Coefficient Estimates

Coefficient	Estimate	Confidence Interval
x_1	12.2991	[7.82, 16.7761]
x_2	-6.9866	[-7.49, -6.4822]
x_3	0.1743	[0.15, 0.1997]
x_1^2	0.6242	[0.21, 1.0379]
$x_1 x_2$	0.7524	[0.72, 0.7884]
x_2^2	-0.0162	[-0.02, -0.0145]
$x_1 x_3$	0.0001	[0.00, 0.0001]
$x_2 x_3$	-0.0065	[-0.03, 0.0185]
x_3^2	-0.0019	[-0.00, -0.0003]
x_1^3	-0.0091	[-0.02, 0.0064]
$x_1^2 x_2$	-0.0225	[-0.02, -0.0214]

Table 6: Goodness of Fit for Extended Polynomial Regression Analysis

SSE	R^2	RMSE	AIC	BIC
2093	0.974	1.63	3079	3163

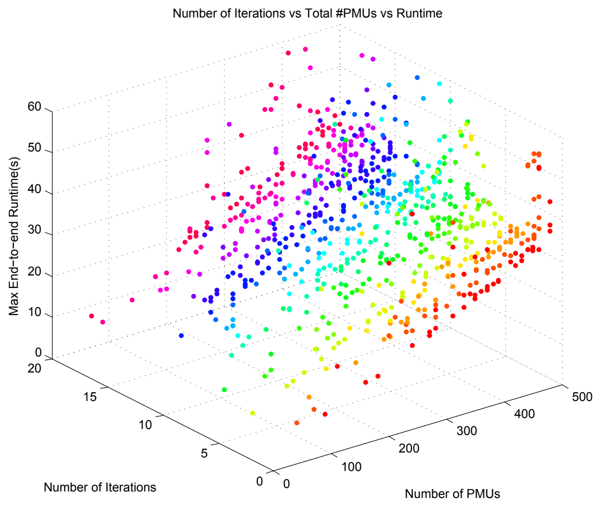

Figure 7: Effect of Iterations and Number of PMU Streams on end-to-end Runtime

5. RELATED WORK

In software architecture design, the accuracy of model-based quality evaluation relies on the parameters used in the evaluation methods. The value of some parameters are only available during system run-time. In our case, such parameters are the number of iterations of distributed state estimation, number of PMU devices, data quality of sensor streams and the concurrency level for adaptive middleware. To what degree these parameters have effects on the quality attributes provides insights on the optimal design of software architecture. This is especially useful for designing essential adaptive components that target the most significant parameters to the quality attributes in priority. Hausi et al. [12] proposed a framework for mapping the adaptation properties derived from control theory properties and quality attributes.

Current model-based quality evaluation methods commonly delegate different sets of probabilistic models to quality attributes. Markov chains, petri nets, queuing networks, finite state automata, stochastic processes, dependency graphs and fault trees are some widely-applied models for quality attributes of reliability, performance, safety, energy consumption. A comprehensive survey on the topic over 188 research papers is presented in [2]. These quality models are mathematically complex, and in general, it is an involved task to derive quality metrics as a function of input distributions.

An improvement to the paradigm was proposed by Meedeniya et al. [11]. The proposed approach still quantifies probabilistic properties using conventional reliability evaluation models, but Monte Carlo simulation is used to re-evaluate the model for the samples taken from input parameter distributions repeatedly. The demonstration of this approach was on an embedded anti-lock brake system that features adaptive cruise control. This approach only applies to a single function component within the system, it is not yet addressed how to integrate multiple quality models for different components, and correlate the quality effects of uncertain parameters to the entire system.

In our approach, we model the whole end-to-end data flow of a distributed power application including sensor streams, network, middleware components and distributed power applications. Our model allows refining the behavior and properties of the system in hierarchical heterogeneous models. We also generate samples of uncertain parameters using MC sampling method. Unlike in [11], MC sample generation is part of the power application model itself and is executable as the simulation runs. We demonstrate automated and scalable techniques of simulating complex distributed applications on large input sequences and show the interactions of the generated samples in the overall model workflow.

6. CONCLUSION

In this paper, a modeling based approach for quantifying model uncertainty in middleware that affect distributed power application timing behavior is presented. The models encompass the entire data flow from sensors to application nodes, through network and middleware components. Using the Ptolemy II framework, we create an integrated design environment that supports random sampling of model parameters and subsequent execution of the simulation model using the generated parameter set. We additionally carry out regression analysis to discover significant model parameters and evaluate their degree of effect on the end-to-end runtime. The results show that among the considered parameters, the maximum number of PMU streams for each area and the middleware concurrency level directly account for the maximum runtime of the end-to-end process. However, data exchange caused by iterations until convergence is a less effective parameter in determining the empirical worst-case runtime. This indicates that power applications can benefit from the distributed architecture with acceptable data communication overhead. It is key to communicate these results to power engineers to help calibrate the scale and configuration of distributed power applications for the future power grid and to determine additional scalability analyses to investigate using a model based design approach. Future directions include incorporating additional design parameters into the Monte Carlo analysis and investigating alternative distribution patterns for the probability distributions of the parameters.

7. ACKNOWLEDGEMENTS

This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

8. REFERENCES

[1] IEEE standard for synchrophasor measurements for power systems. *IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005)*, pages 1 –61, 28 2011.

[2] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya. Software architecture optimization methods: A systematic literature review. *Software Engineering, IEEE Transactions on*, PP(99):1, 2012.

[3] P. Derler, E. Lee, and A. Vincentelli. Modeling cyber-physical systems. *Proceedings of the IEEE*, 100(1):13 –28, Jan. 2012.

[4] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity - the ptolemy approach. In *Proceedings of the IEEE*, pages 127–144, 2003.

[5] A. Gomez-Exposito, A. Abur, A. de la Villa Jaen, and C. Gómez-Quiles. A multilevel state estimation paradigm for smart grids. *Proceedings of the IEEE*, 99(6):952–976, 2011.

[6] J. C. Jensen, D. Chang, and E. A. Lee. A model-based design methodology for cyber-physical systems. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 1666 – 1671, July 2011.

[7] W. Jiang, V. Vittal, and G. Heydt. A distributed state estimator utilizing synchronized phasor measurements. *Power Systems, IEEE Transactions on*, 22(2):563 –571, may 2007.

[8] G. Korres. A distributed multiarea state estimation. *Power Systems, IEEE Transactions on*, 26(1):73–84, 2011.

[9] E. A. Lee. Cps foundations. In *Proc. of the 47th Design Automation Conference (DAC)*, pages 737–742. ACM, June 2010.

[10] Y. Liu, J. M. Chase, and I. Gorton. A service oriented architecture for exploring high performance distributed power models. In C. Liu, H. Ludwig, F. Toumani, and Q. Yu, editors, *Service-Oriented Computing*, volume 7636 of *Lecture Notes in Computer Science*, pages 748–762. Springer Berlin Heidelberg, 2012.

[11] I. Meedeniya, A. Aleti, I. Avazpour, and A. Amin. Robust archeopterix: Architecture optimization of embedded systems under uncertainty. In *Software Engineering for Embedded Systems (SEES), 2012 2nd International Workshop on*, pages 23–29. IEEE, 2012.

[12] N. Villegas, H. Müller, G. Tamura, L. Duchien, and R. Casallas. A framework for evaluating quality-driven self-adaptive software systems. In *Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems*, pages 80–89. ACM, 2011.