JARTA — A Java library to model and fit Autoregressive-To-Anything processes

Conference Paper · December 2013

DOI: 10.1109/WSC.2013.6721508

3 authors:

Tobias Uhlig
Universität der Bundeswehr München
14 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Oliver Rose
Universität der Bundeswehr München
149 PUBLICATIONS **2,076** CITATIONS
SEE PROFILE

Sebastian Rank
Technische Universität Dresden
6 PUBLICATIONS **6** CITATIONS
SEE PROFILE

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

JARTA - A JAVA LIBRARY TO MODEL AND FIT AUTOREGRESSIVE-TO-ANYTHING PROCESSES

Tobias Uhlig
Oliver Rose

Universität der Bundeswehr München
Department of Computer Science
Neubiberg, 85577, GERMANY

Sebastian Rank

Technische Universität Dresden
Department of Engineering
Dresden, 01069, GERMANY

ABSTRACT

JARTA is a Java library to model and fit Autoregressive-To-Anything (ARTA) processes. These processes are able to capture the dependency structure of a system, in contrast to commonly used models, that assume independently distributed random values. This study uses a simulation model of a warehouse to demonstrate the importance of capturing dependencies when modeling stochastic processes. Consequently there is a need for a suitable modeling approach. With JARTA we provide a modern software package to model processes with an appropriate dependency structure. Its two main goals are providing a clean code base for integration in other projects and high transparency for educational purposes. To support these goals JARTA is published under an open source license at http://sourceforge.net/projects/jarta/.

1 INTRODUCTION AND MOTIVATION

Stochastic processes are a common approach to model system behavior without modeling a system in detail. Random number generators (RNG) used to implement these processes, are generally designed to generate independent and identically distributed (i.i.d.) values. In fact being i.i.d. is widely accepted as a desirable quality of RNGs (L'Ecuyer 2006). Considering real world systems the assumption of independence for subsequent events might be wrong. Indeed one can imagine various effects that inevitably will lead to dependencies in system behavior, e.g., internal states, psychological factors, or simple ordering of jobs. Consequently it might be necessary to model stochastic process with a specific dependency structure. Hence, we have to consider two basic questions:

1. Do significant dependencies occur in real world data?
2. If so, do they have an impact on system behavior or is their influence negligible?

With respect to the first question we performed an extensive study, analyzing data of various companies in production and logistics. The results of this survey can be found in Rank et al. (2012). To analyze the dependencies we determined the autocorrelation structures in the given data. Autocorrelation measures the dependency of values of a process (Schlittgen and Streitberg 2001) for a given lag l with:

$$r_l = \frac{\sum_{t=1}^{N-l}(x_t - \bar{x})(x_{t+l} - \bar{x})}{\sum_{t=1}^{N}(x_t - \bar{x})^2}.$$

Summarizing the results, we observed significant dependencies in 47 of 52 data sets. With dependencies occurring in nearly all data samples, we can focus on the second question. Various studies have shown the influence of autocorrelation on basic queuing systems. This will be discussed briefly in the next section. One could still hypothesize, however, that for complex models the influence of these dependencies is rather small. Perhaps the interactions in larger system could distort the dependencies to such an extent that no effect will be observed.

One goal of this publication is to refute this assumption. Therefore we performed a study using a model of a warehouse to directly show the impact of autocorrelation. The results demonstrate the risk of assuming independent behavior for stochastic processes when modeling complex systems. The experiment will be discussed in detail in Section 3.

The second goal of this work is to promote the modeling of stochastic processes with dependencies. As we will see in the next section various approaches exist that provide appropriate processes. However they are still not widely used in the modeling and simulation community (Rank et al. 2012). We introduce JARTA, an open source java library to model and fit ARTA processes, to make modeling with dependencies more accessible. We chose the ARTA approach to model stochastic processes for several reasons. Mainly it requires no user interaction to model the processes, in contrast to other approaches. The basic concept and advantages of the ARTA approach will be discussed in Section 1.2.

1.1 Related Work

The influence of dependencies in stochastic processes is known for quite a long time in the modeling community. As early as 1962 Runnenburg (1962) observed longer queues in a single-server-queue model when the input data showed higher autorcorrelation. Experiments by Livny et al. (1993) and Patuwo et al. (1993) confirmed this observation. More recent papers provided further proof for the influence of autorcorrelation (Altiok and Melamed 2001; Nielsen 2007; Civelek et al. 2009). Those studies relied on simple models to illustrate the effect of autocorrelation, using mostly single-server-queue models. For a more complex real world scenario Pereira et al. (2012) provided indications for comparable results.

Various approaches exist to model stochastic processes with a certain autocorrelation structure. They employ different techniques to generate autocorrelated data. Some like TES rely on distortion other use the concept of minification or maxification. Markovian chains, Brownian motion, or copulas are alternative approaches (Sklar 1973; Lucantoni et al. 1990; Glasserman 2003). For our work we rely on the ARTA concept which is an modification of the ARMA approach (Box and Jenkins 1970). ARTA uses Autoregressive-To-Anything processes by Cario and Nelson (1996) to model stochastic processes. Since it is the underlying mathematical approach implemented in JARTA we will discuss it briefly in the next section.

Despite the obvious necessity to consider dependencies for modeling and the available modeling approaches, even today the industry largely ignores the topic. While packages like MATLAB (The MathWorks, Inc. 2013) or R (R Core Team 2013) readily provide tools to fit processes appropriately, we are not aware of a software library that provides easily available and reusable code. This may be one reason why suitable methods are currently not integrated into existing simulation tools.

1.2 Autoregressive-To-Anything Processes

Autoregressive-To-Anything (ARTA) processes are a proven approach to generate random values with a given marginal distribution and autocorrelation structure. Generating an ARTA process with the desired properties can be easily automated. This is a big advantage considering the required user interaction during modeling of other techniques.

An ARTA process $\{Y_t\}$ models a stationary time series. It uses a standardized Gaussian autoregressive (AR) process with order p, transforming it into the desired marginal distribution F_Y. The underlying AR process $\{Z_t; t = 1, 2, ...\}$ is defined as follows:

$$Z_t = \alpha_1 Z_{t-1} + \alpha_2 Z_{t-2} + ... + \alpha_p Z_{t-p} + \varepsilon_t,$$

where p is the given maximal lag to consider, and ε_t is a series of independent random variables. The random variables are drawn from a normal distribution $N(0,\sigma^2)$ with mean 0 and variance σ^2. We can adjust the variance to generate a process Z_t with a marginal distribution $N(0,1)$, using:

$$\sigma^2 = 1 - \alpha_1 r_1 - \alpha_2 r_2 - ... - \alpha_p r_p,$$

where r_h is the desired autocorrelation for lag h ($r_h = Corr[Z_t, Z_{t+h}]$). Given an AR process with a marginal distribution $N(0,1)$ we can transform its output into uniformly distributed values $U(0,1)$ using the standard normal cumulative distribution function Φ. Applying the inverse distribution function F_Y^{-1} to the ensuing values results in process with the desired marginal distribution F_Y. Consequently the ARTA process is defined as:

$$Y_t = F_Y^{-1}[\Phi(Z_t)].$$

The remaining challenge is to adjust the autocorrelation structure of the base AR process, that directly determines the autocorrelation structure of the ARTA process. In general we can not find the appropriate autocorrelation coefficients r_h for the AR process directly. However, in Cario and Nelson (1996) a numerical search procedure is described to determine the right values. Given the autocorrelation structure of the AR process we can determine the autoregression coefficients (α_h) using the Yule-Walker equations (Schlittgen and Streitberg 2001). Putting it all together we receive a reliable tool chain to model stochastic processes. Cairo and Nelson also provide a reference implementation in Fortran (Cario and Nelson 1996). For practical purposes it is, however, difficult to integrate or extend the provided program.

2 JARTA - CONCEPT AND IMPLEMENTATION

JARTA is a Java library to model and fit ARTA processes. It is distributed as open source software under the Apache License, Version 2.0 (The Apache Software Foundation 2012). The project is hosted online (Uhlig and Rank 2013) at SourceForge. JARTA relies on the Apache Commons Mathematics Library a "library of lightweight, self-contained mathematics and statistics components addressing the most common problems not available in the Java programming language or Commons Lang" (Commons 2013).

JARTA is designed to use a factory design pattern to generate the desired modeling objects (see Figure 1). It provides one factory class to create ARTA processes and a second one to create the underlying AR processes. The *ArProcessFactory* creates an *ArProcess* with a given autocorrelation structure and uses a random number generator from the Commons Math project to model the normal distributed white noise (ε_t). The user can decide which random number generator is used, e.g. a Mersenne Twister or a Well random number generator. In general the RNGs provided by the Commons Math project are better suited for modeling than the default Java implementation, that is known to be flawed (Coddington, Mathew, and Hawick 1999).

Using a factory design pattern allows us to build very lightweight classes for the actual processes, since all the details needed for process generation are moved to the factory. This approach results in very clean code, that is easy to understand. It also simplifies the selection of the appropriate class to model a process. For example we use different implementations for processes with certain marginal distributions, i.e., normal distributions and uniform distributions. These implementations are simpler than the general case, since they consider special properties of the modeled distributions. The *ArtaProcessFactory* creates an *ArtaProcess*, automatically choosing the most effective available implementation. Each *ArtaProcess* is build by combining the appropriate distribution function with an *ArProccess* created by the *ArProccessFactory*. JARTA uses the distribution function provided by the Commons Math project. All distributions typically employed for modeling are available.

In general we put extra effort in code readability. With this approach it is possible to uses the JARTA project to learn the details of modeling and fitting ARTA processes. Software developers often prefer code examples to mathematical notations. With educational purposes in mind, we generally avoid cluttering the

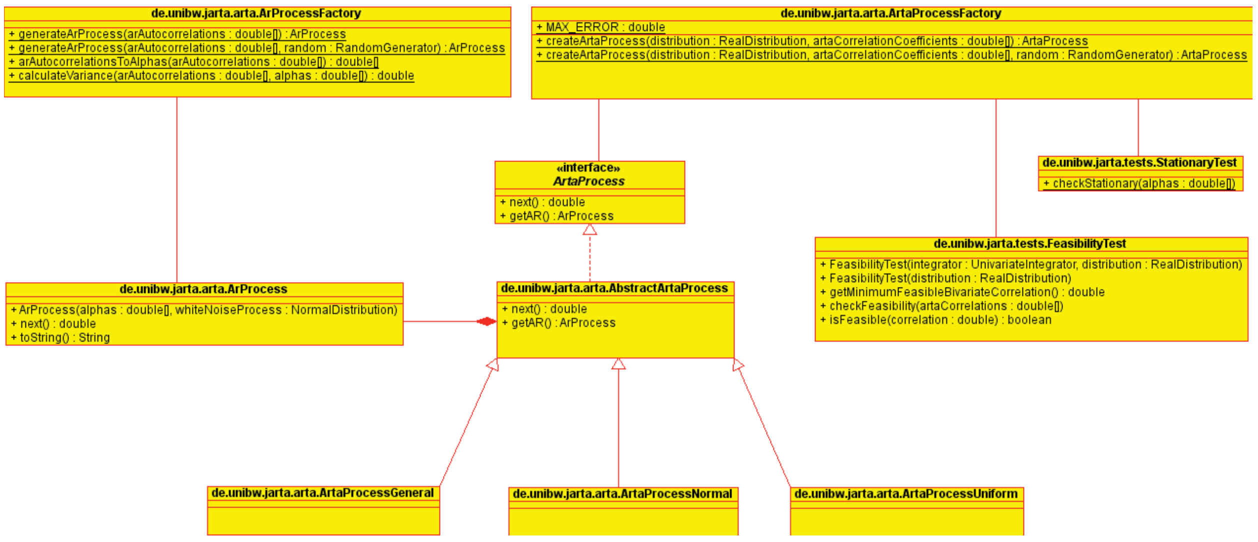
de.unibw.jarta.arta.ArProcessFactory
+ generateArProcess(arAutocorrelations : double[]) : ArProcess
+ generateArProcess(arAutocorrelations : double[], random : RandomGenerator) : ArProcess
+ arAutocorrelationsToAlphas(arAutocorrelations : double[]) : double[]
+ calculateVariance(arAutocorrelations : double[], alphas : double[]) : double
de.unibw.jarta.arta.ArtaProcessFactory
+ MAX_ERROR : double
+ createArtaProcess(distribution : RealDistribution, artaCorrelationCoefficients : double[]) : ArtaProcess
+ createArtaProcess(distribution : RealDistribution, artaCorrelationCoefficients : double[], random : RandomGenerator) : ArtaProcess
«interface»
ArtaProcess
+ next() : double
+ getAR() : ArProcess
de.unibw.jarta.tests.StationaryTest
+ checkStationary(alphas : double[])
de.unibw.jarta.tests.FeasibilityTest
+ FeasibilityTest(integrator : UnivariateIntegrator, distribution : RealDistribution)
+ FeasibilityTest(distribution : RealDistribution)
+ getMinimumFeasibleBivariateCorrelation() : double
+ checkFeasibility(artaCorrelations : double[])
+ isFeasible(correlation : double) : boolean
de.unibw.jarta.arta.ArProcess
+ ArProcess(alphas : double[], whiteNoiseProcess : NormalDistribution)
+ next() : double
+ toString() : String
de.unibw.jarta.arta.AbstractArtaProcess
+ next() : double
+ getAR() : ArProcess
de.unibw.jarta.arta.ArtaProcessGeneral
de.unibw.jarta.arta.ArtaProcessNormal
de.unibw.jarta.arta.ArtaProcessUniform

Figure 1: UML class diagram of core classes in JARTA.

code with run time optimizations. The only exception is the use of caching for certain functions. For those cases we use a cache to store results of run time expensive calculations.

Using JARTA is quite simple. There are two ways to generate an ARTA process. First, we explicitly use the desired distribution function and autocorrelation structure (see Figure 2). The second way is to take a given data sample and to fit an ARTA process to it. This approach uses an empirical distribution function and empirical autocorrelation coefficients taken from the sample to model the process.

```
// select the desired distribution
RealDistribution distribution = new ExponentialDistribution(1.0);

// define the desired autocorrelations
double[] acc =  {0.3, -0.1, -0.5};

// Factory generates the approriate ARTA process
ArtaProcess arta = ArtaProcessFactory.createArtaProcess(distribution, acc);

// get the next value from the ARTA process
double value = arta.next();
```

Figure 2: Example of using JARTA.

Currently we provide neither a graphical nor a command line interface for JARTA, however both could be implemented easily if according interest in the community would emerge. The focus at this time is to provide a low level library that can easily be reused in other projects.

3 STOCHASTIC PROCESSES WITH DEPENDENCIES IN A WAREHOUSE SIMULATION

In this section, we will discuss a study we performed, to demonstrate the influence of dependencies in stochastic processes on a warehouse simulation model. The considered scenario is a typical challenge in logistics. We use simulation to evaluate whether a warehouse can handle a given throughput of goods. Although we did not model an existing warehouse, the model we employed has the typical properties we usually observe in real world problems. The modeled warehouse (see Figure 3) consists of an automated storage and retrieval system (ASRS) with four aisles (A1-4). Furthermore there are two Pickers (P1, P2), an incoming conveyor (I), and outgoing conveyor (O). A transportation system connects all the elements of the model.

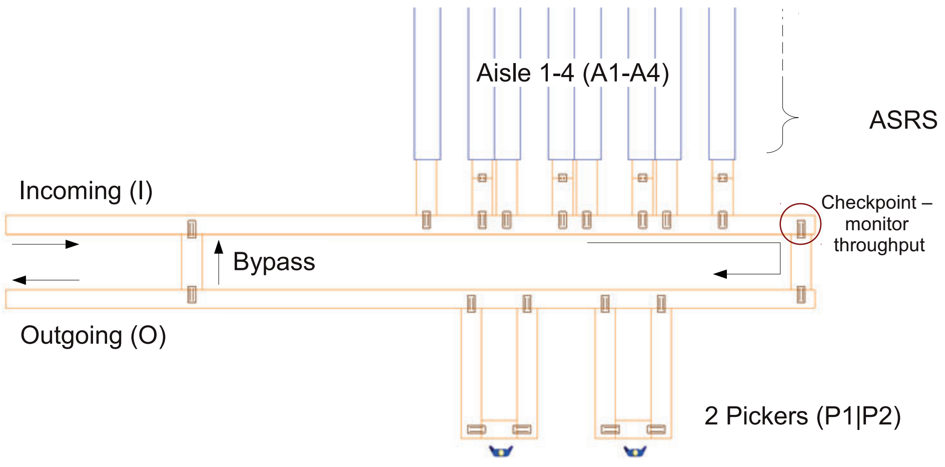

Figure 3: Model of the warehouse.

We define certain streams of goods which are modeled as stochastic processes (see Table 1). They represent the varying demands to transport goods between different locations. Each process is modeled using an exponential distribution (Law and Kelton 2000) to generate inter-arrival times between successive transportation requests.

Table 1: Different streams of goods in the warehouse. Each stream is modeled as stochastic processes. Generated values represent the time between two transportation requests.

$I \Longrightarrow L1$	$L1 \Longrightarrow P1 \Longrightarrow O$	$L4 \Longrightarrow P1 \Longrightarrow O$
$I \Longrightarrow L2$	$L1 \Longrightarrow P2 \Longrightarrow O$	$L4 \Longrightarrow P2 \Longrightarrow O$
$I \Longrightarrow L3$	$L2 \Longrightarrow P1 \Longrightarrow O$	$L1 \Longrightarrow P2 \Longrightarrow L3$
$I \Longrightarrow L4$	$L2 \Longrightarrow P2 \Longrightarrow O$	$L2 \Longrightarrow P2 \Longrightarrow L1$
$I \Longrightarrow P1 \Longrightarrow O$	$L3 \Longrightarrow P1 \Longrightarrow O$	$L3 \Longrightarrow P2 \Longrightarrow L4$
$I \Longrightarrow P2 \Longrightarrow O$	$L3 \Longrightarrow P2 \Longrightarrow O$	$L4 \Longrightarrow P2 \Longrightarrow L2$

Initially, we model the stochastic processes using random numbers that fit the i.i.d. criterion. The results generated during simulation, when no dependencies were considered, serve as reference (see Table 2). In the next step we use the JARTA library to generate processes that have some kind of dependencies. For this study we modeled only processes with significant autocorrelation coefficients (ACC) at lag one. During every simulation we measure the delay occurring for each request and monitor the number of goods passing by the checkpoint during a time frame of one hour.

Table 2: Experimental results showing the relative change for delays in comparison to reference (Ref) and the average hourly throughput of the warehouse.

Exp.	ACC at lag 1 for streams from					Delays - Percentiles			Throughput [goods/hour]
	I	A1	A2	A3	A4	25-p	50-p	75-p	[Mean ± Stdv]
Ref	0	0	0	0	0	36.9	37.3	37.8	118±10.5
0	0.50	0.50	0.50	0.50	0.50	+45%	+45%	+46%	126±17.0
1	-0.50	-0.50	-0.50	-0.50	-0.50	-7%	-8%	-8%	123±8.4
2	0.10	0.10	0.10	0.10	0.10	+6%	+6%	+6%	118±11.6
3	-0.10	-0.10	-0.10	-0.10	-0.10	-4%	-4%	-5%	118±9.5
4	0.25	0.25	0.25	0.25	0.25	+18%	+18%	+19%	121±13.8
5	-0.25	-0.25	-0.25	-0.25	-0.25	-4%	-4%	-5%	119±8.8
6	0.50	0	0	0	0	+23%	+24%	+25%	119±11.5
7	-0.50	0	0	0	0	-4%	-4%	-4%	118±10.2

The results of our study met our expectations. We did observe a change in the system, whenever dependencies were considered. In general positive autocorrelation leads to larger delays and more variance (see Table 2 experiments 0, 2, 4 and 6). The increased variance affected the delays as well as the observed throughput. On the other hand we received an improved system behavior when we considered negative autocorrelation (see Table 2 experiments 1, 3, 5 and 7). Even a single stream with a dependency structure affected the whole system (see Table 2 experiments 6 and 7).

The number of transported items in a given time frame varied much more with an increasing amount of dependencies. Figures 4 and 5 illustrate the changed system behavior. They compare the uncorrelated reference with the strong positive autocorrelation of experiment 0. Regarding the reference scenario the warehouse works perfectly well given the desired throughput. However when positive dependencies were introduced transportation often stalled because of deadlocks resulting from temporary high loads (see Figure 6). This is very interesting, since it shows that neglecting dependencies might have lead to false confidence in the planned system. According to simulation with independent stochastic processes the warehouse performed well and a planner would have deemed it suitable to build the actual warehouse. With the consideration of dependencies this verdict cannot longer be supported.

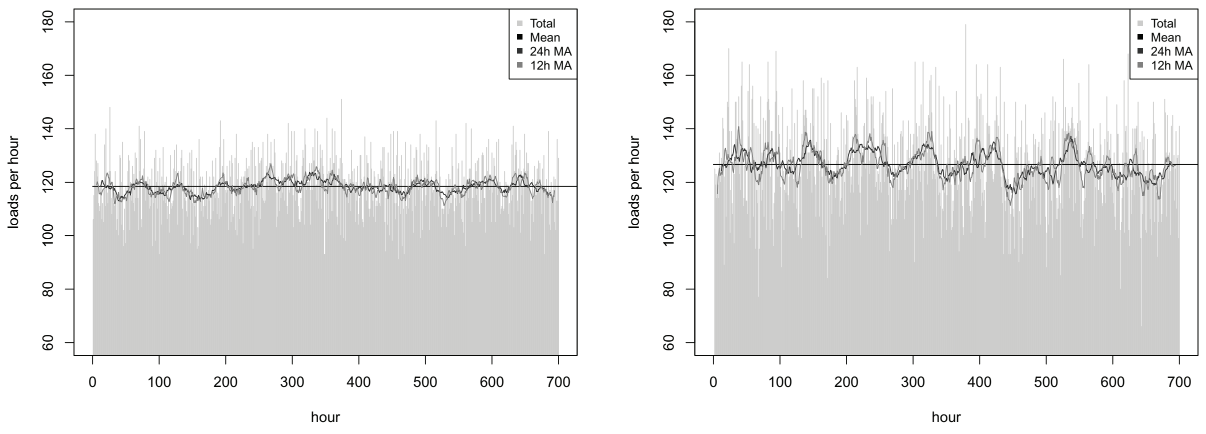

Figure 4: Hourly throughput of warehouse at checkpoint (without dependencies left and with positive autocorrelation right).

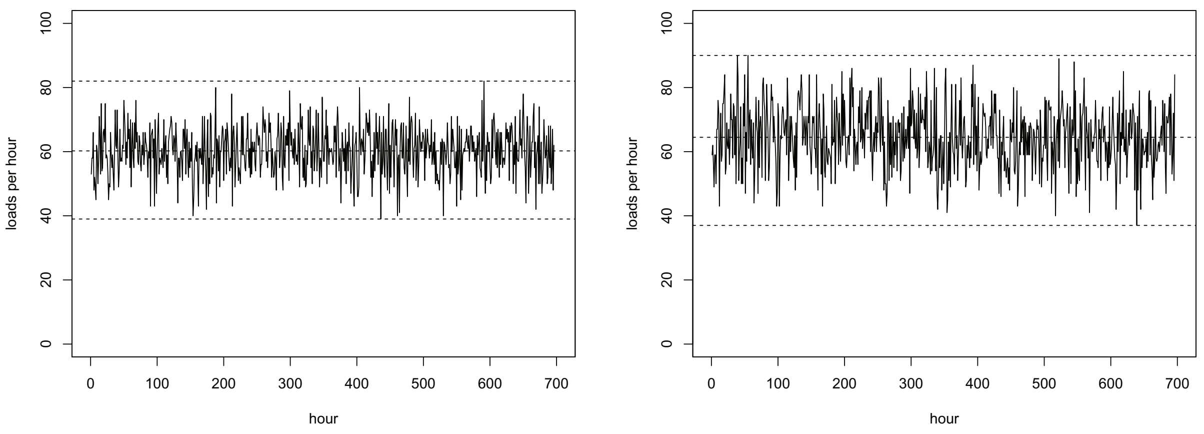

Figure 5: Hourly throughput of Picker 1 (without dependencies left and with positive autocorrelation right).

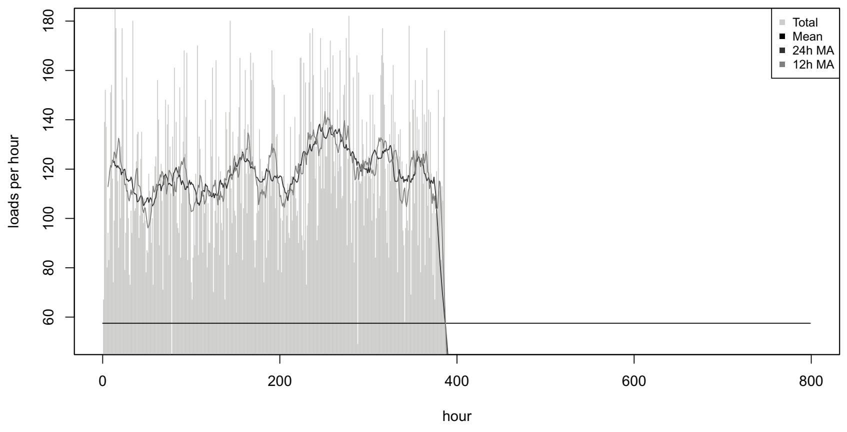

Figure 6: Deadlocks occurred when dependencies were considered.

4 DISCUSSION

As we have seen, including dependencies into the modeling process deserves strong consideration. However several limitations should be kept in mind. For one ARTA processes are strictly a model for stationary processes. Therefore seasonal effects or trends must be considered separately. A further challenge are dynamically changing dependencies, since ARTA processes models a constant dependency structure. In our experience, fitting dependencies from small samples – with less than 1000 values – leads to very unreliable results. Appropriate care must be taken to avoid overfitting. Especially with regard to automated fitting in JARTA the results cannot be applied without previous validation of the generated model.

That being said, even for cases were the dependencies structure is either unknown or changes dynamically one can still use ARTA processes to model extreme cases. The resulting behavior of the modeled system can be used for risk analysis. Considering the warehouse example, we have no certain knowledge of the dependency structure. However, considering various dependency structures during the modeling process raises the awareness with respect to correlation effects on system performance. Planning with dependencies can at least be used to anticipate these effects on systems. Policies can be considered to avoid certain dependencies and in a best case scenario they can be used to generate a certain dependency structure that promote a desired system behavior.

5 CONCLUSION AND OUTLOOK

Using the simulation model of a warehouse, we demonstrated the necessity to model stochastic processes that consider dependencies. ARTA processes are a proven way to do this. With JARTA we provide a modern solution to model and fit these processes. As an open source project JARTA is easy to use and to extend. It is open to participation and collaboration. We will continue the development to cover some of the weaknesses discussed in the previous section. The next logical step is to provide a JARTA-based software solution with a user interface for easy modeling. Since it is still a very young project further validation and testing is needed.

ACKNOWLEDGEMENTS

This research project was promoted by financial resources of the Federal Ministry of Economics and Labor through the German Federation of Industrial Research Associations ”‘Otto-von-Guericke”’ e.V. (AiF) under the authority of the German Logistics Association e.V. (BVL). Project number: 17344 BR

REFERENCES

Altiok, T., and B. Melamed. 2001. “The Case for Modeling Correlation in Manufacturing Systems”. *IIE Transactions* 33 (9): 779–791.

Box, G. E. P., and G. M. Jenkins. 1970. *Time Series Analysis: Forecasting and Control*. Holden-Day.

Cario, Marne C. and Nelson, Barry L. 1996. “ARTA Software”. Accessed April. 9, 2013. http://users.iems.northwestern.edu/~nelsonb/ARTA/.

Cario, M. C., and B. L. Nelson. 1996. “Autoregressive to Anything: Time-series Input Processes for Simulation”. *Operations Research Letters* 19 (2): 51–58.

Civelek, I., B. Biller, and A. Scheller-Wolf. 2009. *The Impact of Dependence on Queueing Systems*. Working paper, Carnegie Mellon University, Pittsburgh.

Coddington, P. D., J. A. Mathew, and K. A. Hawick. 1999. “Interfaces and Implementations of Random Number Generators for Java Grande Applications”. In *Proceedings of the 7th High-Performance Computing and Networking Conference*, edited by P. Sloot, M. Bubak, A. Hoekstra, and B. Hertzberger, Lecture Notes in Computer Science, 873–883. Berlin Heidelberg: Springer.

Commons, Apache 2013. “Math - Commons Math: The Apache Commons Mathematics Library”. Accessed April. 9, 2013. http://commons.apache.org/proper/commons-math/.

Glasserman, P. 2003. *Monte Carlo Methods in Financial Engineering*. Springer.

Law, A. M., and D. W. Kelton. 2000. *Simulation Modeling and Analysis*. third ed. Singapore: Mcgraw-Hill Professional.

L’Ecuyer, P. 2006. “Uniform Random Number Generation”. In *Elsevier Handbooks in Operations Research and Management Science: Simulation*, edited by S. G. Henderson and B. L. Nelson, 55–81. Amsterdam: Elsevier Science.

Livny, M., B. Melamed, and A. K. Tsiolis. 1993. “The Impact of Autocorrelation on Queuing Systems”. *Management Science* 39 (3): 322–339.

Lucantoni, D. M., K. S. Meier-Hellstern, and M. F. Neuts. 1990. “A Single Server Queue with Server Vacations and a Class of Non-renewal Arrival Processes”. *Advances in Applied Probability* 22 (3): 676–705.

Nielsen, H. E. 2007. “Autocorrelation in Queuing Network-type Production Systems”. *International Journal of Production Economics* 110 (1-2): 138–146.

Patuwo, E. B., R. L. Disney, and D. C. McNickle. 1993. “The Effect of Correlated Arrivals on Queues”. *IIE Transactions* 25 (3): 105–110.

Pereira, D. C., D. del Rio Vilas, N. R. Monteil, R. R. Prado, and A. G. del Valle. 2012. “Autocorrelation Effects in Manufacturing Systems Performance: a Simulation Analysis”. In *Proceedings of the 2012*

Winter Simulation Conference, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. Uhrmacher, 1–12. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

R Core Team 2013. "R: A Language and Environment for Statistical Computing". Accessed April. 9, 2013. http://www.R-project.org.

Rank, S., T. Uhlig, T. Schmidt, and O. Rose. 2012. "Beherrschung stark korrelierter Logistik- und Produktions-Prozesse". In *Tagungsband zum 8. Fachkolloquium der Wissenschaftlichen Gesellschaft fuer Technische Logistik e. V.*, edited by H. Zadek, 135–146. Magdeburg: Logistics Journal.

Runnenburg, J. T. 1962. "Some Numerical Results on Waiting-time Distributions for Dependent Arrival-intervals". *Statistica Neerlandica* 16 (4): 337–347.

Schlittgen, R., and B. Streitberg. 2001. *Zeitreihenanalyse*. 9. ed. Muenchen, Wien: Oldenbourg Wissenschaftsverlag.

Sklar, A. 1973. "Random variables, Distribution Functions, and Copulas". *Kybernetika* 9 (6): 449–460.

The Apache Software Foundation 2012. "Apache License, Version 2.0". Accessed April. 9, 2013. http://www.apache.org/licenses/LICENSE-2.0.html.

The MathWorks, Inc. 2013. "MATLAB". Accessed April. 9, 2013. http://www.mathworks.de/products/matlab/.

Uhlig, Tobias and Rank, Sebastian 2013. "A Java Library to Model and Fit ARTA Processes". Accessed April. 9, 2013. http://sourceforge.net/projects/jarta/.

AUTHOR BIOGRAPHIES

TOBIAS UHLIG is a PhD student and research assistant at the Universität der Bundeswehr München, Germany. He received his M.S. degree in Computer Science from Dresden University of Technology. His research interests include evolutionary computation and its application to scheduling problems and modeling. He is a member of the IEEE RAS Technical Committee on Semiconductor Manufacturing Automation. His email address is tobias.uhlig@unibw.de.

SEBASTIAN RANK is a PhD student at Technische Universität Dresden. He is member of the scientific staff at the chair of Logistics Engineering. Having received his Diploma (similar to M.S. degree) in Economics and Engineering from Technische Universität Dresden in 2011 his research interests include autocorrelation in logistics systems as well as the application of virtual reality in complex material flow systems. His email address is sebastian.rank@tu-dresden.de.

OLIVER ROSE holds the Chair for Modeling and Simulation at the Department of Computer Science of the Universität der Bundeswehr München, Germany. He received an M.S. degree in applied mathematics and a Ph.D. degree in computer science from Würzburg University, Germany. His research focuses on the operational modeling, analysis and material flow control of complex manufacturing facilities, in particular, semiconductor factories. He is a member of IEEE, INFORMS Simulation Society, ASIM, and GI, and General Chair of WSC 2012. His email address is oliver.rose@unibw.de.