Towards Accuracy-Adaptive Simulation for Efficient Design-Space Optimization

Armin Zimmermann and Christoph Bodenstein
System & Software Engineering
Technische Universität Ilmenau, Germany
{armin.zimmermann | christoph.bodenstein}@tu-ilmenau.de

***Abstract*—Designing complex embedded systems is a challenge, which benefits from using models for various tasks. Among them, an important one is finding a good (or the best) system design alternative for a given set of changeable parameters. Such decisions are usually based on non-functional properties which can be quantitatively expressed and evaluated using a model. Automatic optimization would be advantageous, but one of its problems is the enormous computational effort to evaluate the model for a large number of parameter sets. In this paper a new idea to speed up indirect optimization is proposed, which makes use of an integration of optimization control and underlying performance evaluation technique. Moreover, we propose to adaptively use refinements or abstractions of complex hierarchical models to control the trade-off between model complexity and result accuracy of a performance evaluation, and demonstrate this dependency with an example.**

***Index Terms*—Modelling, simulation, optimization.**

I. Introduction

The design of complex embedded systems is a highly complex task — while functional requirements have to be fulfilled, non-functional requirements such as timeliness or power consumption can be used to evaluate the set of possible design alternatives in search of a good solution. In abstract terms, this is an optimization with the goal of finding a parameter setting for which the best result for a certain cost or profit function is expected.

A model from which such performance measures can be derived is obviously necessary for this, which needs to be expressive enough to cover real embedded system problems. In this paper we use stochastic Petri nets [1], [2], a formally well-understood description technique for which many performance evaluation methods and tools exist. A formal basis is especially needed when structural techniques for approximation are used as described later.

In later design phases such models become more and more detailed and it is often not possible any more to use a direct optimization technique such as linear programming. Heuristics for indirect optimization[1] such as simulated annealing or genetic algorithms are much more efficient than the naive implementation of a complete search of the parameter space. However, these techniques require one evaluation of the model for each parameter set to be analyzed, which can still be a large number. As the models may be quite complex, their evaluation may be time-consuming for each of these steps, such that the overall algorithm becomes unacceptably slow.

In this paper we develop ideas for a more efficient optimization in such a setting. We point out the trade-off between accuracy and computation time of a model analysis and how it can be exploited during an indirect optimization. We propose a tighter coupling between optimization control algorithm and performance evaluation technique. Several evaluation methods can be applied and selected according to the necessary accuracy. A specific simulation-based method is proposed as an additional contribution. Hierarchical formal models such as stochastic (colored) Petri nets can be used to make a model analyzable on different levels of detail, which allows to control the accuracy/efficiency trade-off inside the model individually for different parts of it. This can be done depending on the significance of the model part for the overall evaluation function. We expect a significant speed up for the optimization of models for embedded systems

[1] called black-box randomized search in [3]

based on experience from earlier work (c.f. [4], [5] and Section II). In this paper, we adopt the application area of in-car communication as an example of model-based system design.

Model-based performance evaluation and optimization for embedded systems has gained a lot of interest in the literature. Evolutionary algorithms, for instance, are used to compute the optimum for multiple objectives in a single run. In this paper we assume problems in which the relative impact of differing multiple objective is specified, resulting in a single optimization function as a weighted sum of individual measures. A quantitative evaluation of methods in this field is given in [6].

An extensive overview of design-space exploration and optimization techniques for models of embedded systems is contained in [3]. Moreover, a hybrid approach using system-level simulation feeding traces into a Real-Time Calculus analyzer is developed. The results are implemented in the software tool framework EXPO.

The paper is structured as follows. The following section discusses the trade-off between evaluation accuracy and speed, recalls some applicable techniques form the literature, and introduces the idea of an integration of optimization control algorithm and performance evaluation. In Section III the idea to reduce simulation and optimization time by using an accuracy-adaptive simulation is presented. A stochastic Petri net example and its simulation results are presented in Section IV to give evidence for the idea's applicability. Finally, concluding remarks are given.

II. Efficient Design Optimization Techniques

Model-based software and system design processes for embedded systems [7], [8], [9] contain several optimization points. Examples include the allocation of software components to electronic control units (ECUs) [10], and the number and placement of ECUs together with the communication network structure. Different model-based implementation tools like Matlab, Ascet, MLDesigner, and Modelica are used to manage increasing system complexity, making the development more cost efficient and generate production code from models automatically. However, on the design level, these tools do not allow an optimization that is more efficient than standard indirect methods. Automatic optimization is thus not currently applied to non-trivial systems. Using the techniques described in this paper in automotive system design would thus be beneficial to find new configurations faster.

The main problem of indirect optimization methods is the high computational effort necessary. When the functional dependency between a parameter set and the corresponding value of the cost (or profit) function is not directly expressible, but requires a sophisticated algorithm (such as a simulation run), heuristics like simulated annealing can be used. This leads to repeated executions of the underlying simulation, which may require minutes or hours to complete for a statistically significant performance evaluation of a complex system. The overall indirect optimization will then take too long because of the huge number of possible parameter sets (i.e., the size of the design space).

Efficiency of such indirect optimization methods can be improved when the complete separation between optimization control algorithm (e.g., simulated annealing) and performance evaluation technique (i.e., a detailed simulation) is abandoned and some information is exchanged between the two elements in addition to the standard parameter set / evaluation result data. We point out that both parts of the overall algorithm usually include some kind of statistical uncertainty. The optimization control heuristic makes probabilistic decisions such as acceptance of a parameter set and generation of new candidates. Different performance evaluation techniques have individual tradeoffs between computation time and result accuracy. Possible candidates with increasing accuracy are approximation techniques, numerical analysis of simplified models, simulation of simplified models, and detailed simulation. Most applicable heuristic techniques do not require an exact (or highly accurate) result for the first parameter set evaluations, because the rough environment of promising regions in the design space are searched. It is thus not necessary to do a fully detailed simulation in these cases. Instead, it makes sense to control performance evaluation precision in a way that corresponds to the accuracy expected by the optimization step. There are many ways to set up heuristics

under this idea, of which one is proposed in this paper in more detail.

Without the idea to integrate optimization control and evaluation algorithm, there are two ways to speed up the overall algorithm: 1) reducing the number of parameter sets for which a performance evaluation should be started, and 2) reducing the computational effort per evaluation run. Both techniques will potentially decrease result quality. Parameter set reduction is usually done by using optimization heuristics [11] such as simulated annealing, taboo search, or genetic algorithms.

A reduction in the computational effort per evaluation is possible by either selecting a faster (and usually less exact) analysis technique or a change of the model itself. Among the first are numerical approximate evaluation techniques such as network calculus [12] (if applied to a detailed behavioral model), performance bounds [13], and state-space reduction methods [14]. Simulation can easily be used for an accuracy control by simply adjusting the required confidence interval size and error probability.

Model-reduction techniques in the literature aim at reducing the state space size indirectly by simplifying a model such that its main characteristics are kept. Examples include techniques based on aggregation [15], [16], [17], [18] and decomposition [19].

An improvement in speed of indirect optimization has been achieved by dividing the algorithm in two phases, and using a very fast approximate evaluation technique in the first phase [4]. The second, fine-grained optimization phase can then be started in a promising area of the parameter space and is tuned to converge much faster. The introduction of a first phase was inspired by ordinal optimization [20].

However, even in this method there is still no real accuracy control possible, although it can be seen as a special case of our more general idea presented here. We propose to compute the necessary accuracy during the optimization (which starts at a quite low level and increases over time for simulated annealing, for instance), and to use this accuracy requirement to control the performance evaluation steps. By doing so, evaluation accuracy will be significantly smaller in the beginning, which allows to use much faster evaluation algorithms from the set mentioned above. A new possible technique for this would be an accuracy-adaptive simulation method that uses refined or aggregated model parts individually, depending on the performance measure and result accuracy.

III. Towards Accuracy-Adaptive Simulation

While the principal tradeoff between evaluation accuracy and computational effort is well-known, methods for directly controlling it have attracted surprisingly few attention in the literature. Many system engineering approaches include a top-down modeling of the planned system [21], [22], which usually means enriching a model with more and more details. Software tools allow hierarchical refinements and performance evaluations of each refined model, but usually only the most refined model available is considered further.

Performance evaluation at different levels of abstraction is applied to speed up optimization in previous work [4] and other papers of the same authors. However, the approach is restricted to two distinct abstraction levels with individual analysis techniques. Moreover, both use the same model in a non-hierarchical way. The technique is thus not adaptively applicable.

A similar idea (without relation to optimization) is proposed in [23], but there the model refinements are only considered in different levels of the whole model. When viewing the refinement as a tree structure, only mode parts that are equally far away from the root node are considered for an analysis.

Different from the prior work we propose to adapt the refinement level of submodels *individually*, i.e., to simulate different parts of the system on different abstraction levels. An example overview is sketched in Fig. 1. Simulation time can be reduced while precision of simulation of special parts is very high and the precision of the whole system is still adequate to get significant results for optimization.

Individual refinement during simulation obviously allows more options to control the accuracy / speed tradeoff. An optimization function (cost or profit depending on the parameter set) will depend in different orders of magnitude on certain parts of a system

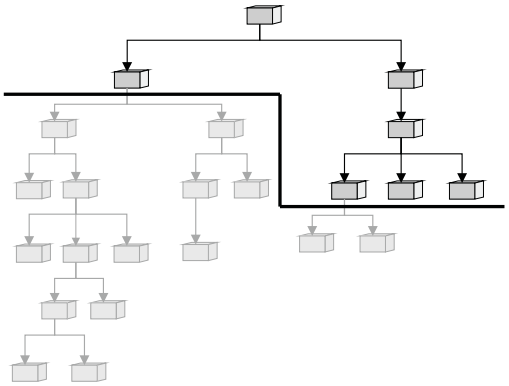

Fig. 1. Individual refinement levels of a model

model. The question which parts of the model should be more detailed than others should be solved depending on the type and structure of performance measure, or simply by an online sensitivity analysis during the optimization. Structural techniques such as the approximation used in [4] for Petri net models can guide such decisions. This is, however, future work.

When only certain refinement levels of the overall model are used, this requires a hierarchical model of a system, which is a standard requirement for complex system modeling which is available in stochastic Petri net models that the proposed technique uses. An additional prerequisite is that the hierarchical model must be able to be simulated / analyzed no matter which refinement setup is chosen. A solution to this requirement is not so obvious because beside technical tool issues the model itself must be correct and complete, independently of what individual refinement scheme is chosen. This is not the case in general hierarchical models, because usually only the fully detailed model is correct, and there is no way of checking whether a refined function block model behaves like one block (which is usually defined using program language constructs).

Essential for the overall approach is a consistency between modeled components and their refinements on every possible abstraction level. For formal models such as Petri nets, this is possible indeed, although the necessary techniques may not be readily available. We ensure this by keeping original and refined model parts structurally consistent. Informally this means that a refined model part will behave similar to the original one in terms of token input / output relations. Delays and synchronization are examples of issues that depend on the refined details. This approach is a principal advantage of Petri nets as a formal model in which, i.e., invariants are not destroyed by adding more details. Techniques for such well-formed refinements are known in the literature, cf. [21], [22].

Hierarchical Petri nets are usually defined such that either places (e.g., in [24]) or, more commonly, transitions (e.g., in [25], [5]) are refined by subnets. However, this means only that a transition-bordered subnet refines the transition, and that all places connected to the upper-level transition are visible in the lower-level subnet. To check if the subnet qualitatively behaves like the refined transition requires a structural analysis. Full coverage of this issue is future work. However, there is a rich body on aggregation techniques for stochastic Petri nets [26], [15], [27] which use structural analysis techniques to directly derive a less complex model with similar performance. The task we have to solve in our method, namely, checking if two models are similar, is much simpler and can use the algorithms described in the mentioned literature.

IV. An Example

In this section a simplified model of distributed computation in an automotive setting is used to show that the ideas of accuracy-adaptive simulation presented in the previous section can be applied. The trade-off between result accuracy and the level of detail of the model (together with the corresponding simulation effort) is shown. We restrict ourselves to discrete-event simulation of extended deterministic stochastic Petri nets (EDSPNs [2]), which allow transition firing times to be either exponentially distributed (depicted by a rectangle), deterministic (shown as a filled rectangle), or zero (immediate transitions are drawn as a vertical bar).

The software tool TimeNET [28], [29] is used for drawing and evaluating the models. It should be noted that the example itself is not realistic, but used for evaluating the methodology only.

An example model and its refinement

Figure 2 shows a high-level model of a distributed computation in an automotive environment. A sensor value is created from time to time (transition `SensorEvent` fires), and the event is transferred over a CAN bus (`CANTrans1`). The event is processed in a software module on a first ECU (`Process1`), resulting in an intermediate event, which is transferred to a second ECU (`CANTrans2`) and processed there (`Process2`).

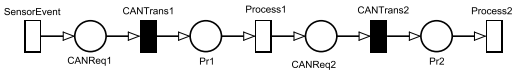

Fig. 2. Top-level Petri net model

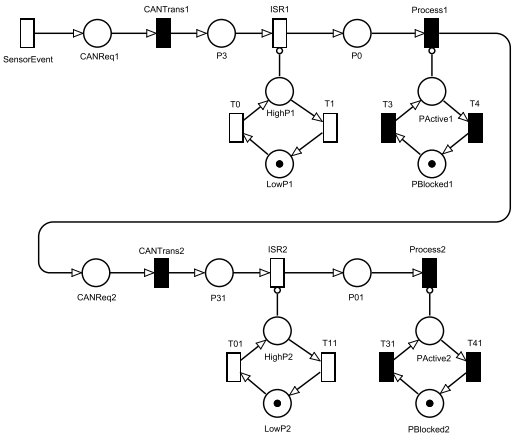

Fig. 3. Model with refined ECU software processes

Let us assume that we are interested in the end-to-end delay between a sensor creation event and the time when it has been processed by `Process2` (as a simplistic example). For this reason we want to compute both the mean end-to-end delay as well as the probability of missing a certain deadline on it (which can be interpreted as quantiles of the end-to-end delay distribution function). It should be noted that in the subsequent figures only the system models are shown, while extensions for the definition of performance measures (namely the quantiles) and their corresponding instrumentation of the model are not shown to increase understandability. This means, among others, that transition `SensorEvent` fires again only after the full transmission and processing has finished. Thus, only one message of our selected type will be present at any time in the system.

For this example, we assume that the first simple model has been refined in some points in the course of the design process, and the result is shown in Figure 3. Due to space restrictions we cannot detail the reasoning behind the refinements here.

The CAN bus transmissions are kept simple for now, but both processing activities have been enriched with more details. It was specified that both are executed with a two-phase interrupt handling, including the actual interrupt service routine (`ISR1` and `ISR2`) which is followed by a second-level process which does the actual processing (`Process1` and `Process2`). Moreover, the ISRs may be blocked because of a high-priority ISR running at the moment (there is a token in place `HighP1`. This is modeled by the small net fragments below the two ISR transitions. The second added detail is a model of process activation in the ECUs: there is a fixed schedule (deterministic transitions `T3` and `T4` for `Process1` and similarly for the second process) which allows our software components to be executed only from time to time, adding delay. The inhibitor arcs from the upper places keep the transitions from firing.

A further refinement step applies to the CAN bus model. We assume here that both transmissions use the same bus and that there is a (set of) higher-prioritized messages transported over it, thus blocking the bus for our application from time to time. This model extension is shown in Figure 4.

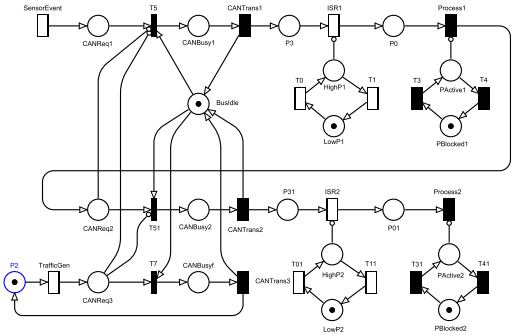

Fig. 4. Complete model including CAN bus refinement

Transition `TrafficGen` models the arrival of a high-priority message that requires CAN

bus access. The three immediate transitions `T5`, `T51` and `T7` describe bus access control and different priorities (using inhibitor arcs). This part of the model is based on the CAN bus MAC layer model presented in [30] with some simplifications that do not change the behavior. The numerical values from the reference are adopted (baud rate 10kbps), which lead to a delay of 10 (msec) for transitions `CANTrans<i>` with the payload assumed.

Although it has not been specifically shown in the model figures above, it should be quite clear that the refinements can be seen as hierarchical — for instance, `ISR1` and `Process1` in Figure 3 as well as the Petri net fragments below would represent the contents of a subpage refining transition `Process1` of the initial top-level model. There would be many possible ways of using the refinements or not in a later actual optimization controlling model details. For this paper, only the shown ones were selected, with an additional model which is not depicted. This fourth model is informally "in between" the accuracy of the ones shown in Figures 2 and 3. Only the software process transitions are refined in it, nothing else.

Simulation results

When we want to use the models for an accuracy-adaptive simulation, the fully detailed model is the starting point and the less refined ones are derived from it. This could be done by using the earlier versions of the model from the top-down design. However, this approach would not ensure validity of the less refined models. Transition delays are defined for these models, but they usually represent the less informed knowledge of earlier design stages. At the point when the model is completely refined, the more abstract models can be improved by analyzing the behavior of each refinement to set the transition delay accordingly. Response-time equivalent approximation techniques for Petri net models (compare, e.g., [31]) can be exploited for this task. This has been done for the transitions of the top-level model and their individual subnets in isolation. The mean delay of traversing the corresponding subnet is derived by a simulation with high accuracy and using Little's law [32] afterwards. The resulting delay is taken as the transition firing delay in the abstract model.

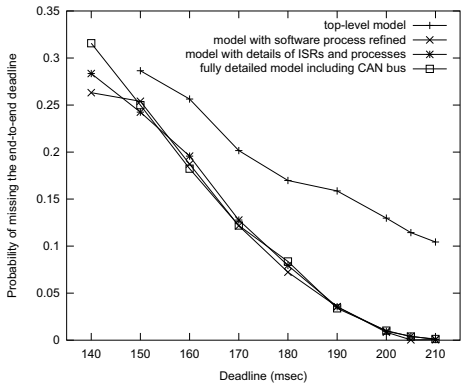

Fig. 5. Probability of missing the end-to-end deadline versus deadline

All computations have been carried out on a Intel centrino Core 2 Duo laptop computer running the eDSPN simulation module of TimeNET 4 [28] with 2.1 GHz in a Windows XP environment. Only simulation is used, numerical analysis is not possible because of the number of activities with non-exponentially distributed delay. The standard settings for TimeNET's simulation accuracy have been applied: confidence level 95% (probability of the actual result to be within the confidence interval) and relative half width of confidence interval of 10%.

The first evaluation covered the mean end-to-end delay for all models. It turned out that identical results of about $120\,\mathrm{msec}$ could be computed by simulating any one of the four differently refined models (within the limits of accuracy of simulation). This shows the effectiveness of response-time approximation and how well robust performance measures can be approximated with much less detailed models.

In a real-time embedded system not (only) the mean end-to-end delay is important, but the probability of meeting a specific deadline. This is equivalent to deriving a quantile of the end-to-end delay distribution function, a measure that depends on the function form (and thus higher moments than only the mean) much more than the first measure.

Figure 5 shows a plot of the probabilities for different deadlines and the four considered models. It is astonishing to see that already the "model with software processes refined", which represents a major simplification of the

full model, approximates the quantile values well. Only the top-level model alone is far away from the actual value, although even there the characteristic form of the curve is retained. In previous work it has been shown that such an approximation can already be used for an optimization step that detects promising regions of the parameter space [4], [5].

However, there are still significant differences in the results, which are not directly visible in the plot. For a deadline of 200 msec, for instance, the probability of missing it is 0.003859 in the fully detailed model. The model in which only the CAN bus is approximated (Figure 3) results in an error of 6.9%, the one in which also the ISRs are simplified has an error of 16.8%, while the most abstract top-level model results in an error of 1174%.

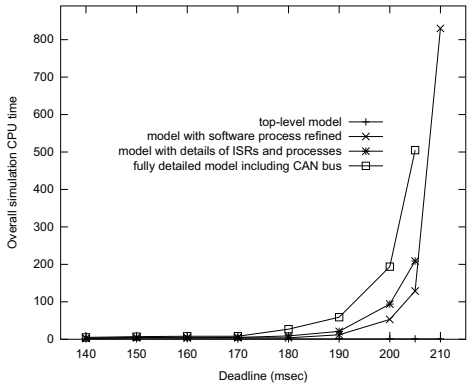

Fig. 6. CPU time for simulation runs

As we are interested in the trade-off between result accuracy and computation effort, the overall simulation times for the experiments that led to Figure 5 have been plotted in Figure 6. It shows the CPU time consumption for each simulation run in seconds. This is computed after a simulation run, by adding the single computation times of all simulation slave processes in the master/slave concept of TimeNET. The time of stopping is decided by the master process based on standard tests for the achieved statistical confidence, which is set by the user.

The results underline the claim that simulation run time depends significantly on the amount of model detail. For the top-level model only one second is enough even for the "hard" computations with deadlines above 200, where the other models already lead to significant computation times which might be unacceptably long for an optimization with thousands of evaluations. In addition, it is visible that each step of adding detail to the model leads to an increase in computation time. In the future we will analyze which refinement types have what impact, taking into account the type of performance measure. An optimization based on the accuracy-adaptive simulation idea would then be able to choose from the models (in much more detail for individual refinements and hierarchical levels than shown here), to get a model with the right trade-off between accuracy and computational effort for the current state of optimization control.

The findings must be reinforced by further example models in the future.

V. Conclusion

The paper presented ideas towards a more efficient automatic optimization for models of embedded systems. We propose to better integrate optimization control algorithm and performance evaluation method. Accuracy control information taken from the optimization heuristic such as simulated annealing can then be used to evaluate the model with the current parameter set only as accurately as necessary, with the potential of speeding it up significantly. This is possible because there are several performance evaluation techniques available with different accuracy / speed ratio. The paper listed some of them, and added another one that exploits top-down model construction and resulting refined hierarchical model. We propose to adaptively control the level of detail of refinements in the model for individual parts of it. A simplified example from the automotive domain is presented to show the trade-off between level of detail with its corresponding accuracy and computation speed. Moreover, it demonstrates that the type of performance measure computed will heavily influence the trade-off.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, ser. Series in parallel computing. John Wiley and Sons, 1995.

[2] R. German, *Performance Analysis of Communication Systems, Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.

[3] S. Künzli, "Efficient design space exploration for embedded systems," PhD thesis, ETH Zurich, Apr. 2006.

[4] A. Zimmermann, D. Rodriguez, and M. Silva, "A two phase optimisation method for Petri net models of manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 12, no. 5/6, pp. 409–420, Oct. 2001, special issue "Global Optimization Meta-Heuristics for Industrial Systems Design and Management".

[5] A. Zimmermann, *Stochastic Discrete Event Systems*. Springer, Berlin Heidelberg New York, 2007.

[6] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117 – 132, Apr. 2003.

[7] A. Sangiovanni-Vincentelli, "Electronic-system design in the automobile industry," *Micro, IEEE*, vol. 23, no. 3, pp. 8 – 18, May-June 2003.

[8] A. Sangiovanni-Vincentelli and G. Martin, "Platform-based design and software design methodology for embedded systems," *Design Test of Computers, IEEE*, vol. 18, no. 6, pp. 23 –33, Nov/Dec 2001.

[9] C. Bodenstein, F. Lohse, and A. Zimmermann, "Executable specifications for model-based development of automotive software." in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC 2010)*, Istanbul, Turkey, oct 2010.

[10] F. Lohse, V. Zerbe, and T. Luetzelberger, "Architecture analysis and optimization of reconfigurable, complex systems," in *Proc. 14th IEEE Int. Conf. on Intelligent Engineering Systems (INES 2010)*, Gran Canaria, Spain, May 2010.

[11] C. L. Reeves, *Modern Heuristic techniques for Combinatorial Problems*. Wiley, 1993.

[12] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, ser. Lecture Notes in Computer Science. Springer Verlag, 2001.

[13] J. Campos and M. Silva, "Structural techniques and performance bounds of stochastic Petri net models," in *Advances in Petri Nets 1992*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Verlag, 1992, vol. 609, pp. 352–391.

[14] B. R. Haverkort, "Approximate performability and dependability analysis using generalized stochastic Petri nets," *Performance Evaluation*, vol. 18, no. 1, pp. 61–78, 1993.

[15] J. Freiheit and J. Billington, *Formal Methods in Software Engineering*, ser. lncs. springer, 2003, vol. 2885, ch. New developments in closed-form computation for GSPN aggregation, pp. 471–490.

[16] F. Bause and P. Buchholz, "Aggregation and disaggregation in product form queueing Petri nets," in *Proc. 7th Int. Workshop on Petri Nets and Performance Models*, Saint-Malo, France, 1997.

[17] P. Buchholz, "Hierarchies in colored GSPNs," in *Application and Theory of Petri Nets 1993*, ser. Lecture Notes in Computer Science, M. A. Marsan, Ed. Springer Verlag, 1993, vol. 691, pp. 106–125.

[18] ——, "Aggregation and reduction techniques for hierarchical GCSPNs," in *Proc. 5th Int. Workshop on Petri Nets and Performance Models*, Toulouse, 1993, pp. 24–33.

[19] J. Freiheit and A. Zimmermann, "A divide and conquer approach for the performance evaluation of large stochastic Petri nets," in *Proc. 9th Int. Workshop on Petri Nets and Performance Models (PNPM)*, Aachen, 2001, pp. 91–100.

[20] Y. C. Ho, R. Sreenivas, and P. Valiki, "Ordinal optimisation of DEDS," *Journal of Discrete Event Dynamic Systems*, vol. 2, pp. 61–88, 1992.

[21] M. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer Academic Publishers, 1993.

[22] M. C. Zhou, F. DiCesare, and A. A. Desrochers, "A hybrid methodology for synthesis of Petri nets for manufacturing systems," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 8, pp. 350–361, 1992.

[23] M. Radetzki and R. S. Khaligh, "Accuracy-adaptive simulation of transaction level models," in *Proc. Design, Automation and Test in Europe (DATE)*. IEEE Computer Society, 2008, pp. 788–791.

[24] F. Bause, P. Buchholz, and P. Kemper, "QPN-tool for the specification and analysis of hierarchically combined queueing Petri nets," in *Quantitative Evaluation of Computing and Communication Systems*, H. Beilner and F. Bause, Eds. Springer Verlag, 1995, vol. 977, pp. 224–238.

[25] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, ser. EATCS Monographs on Theoretical Computer Science. Springer Verlag, 1992.

[26] N. Lilith, J. Billington, and J. Freiheit, "Approximate closed-form aggregation of a fork-join structure in generalised stochastic Petri nets," in *Proc. 1st Int. Conf. on Performance Evaluation Methodolgies and Tools (VALUETOOLS 2006)*, Pisa, Italy, oct 2006.

[27] J. Freiheit and A. Heindl, "Novel formulae for GSPN aggregation," in *Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2002)*, Fort Worth, Texas, 2002, pp. 209–216.

[28] A. Zimmermann, J. Freiheit, R. German, and G. Hommel, "Petri net modelling and performability evaluation with TimeNET 3.0," in *11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, ser. Lecture Notes in Computer Science, vol. 1786, Schaumburg, Illinois, USA, 2000, pp. 188–202.

[29] A. Zimmermann, M. Knoke, A. Huck, and G. Hommel, "Towards version 4.0 of TimeNET," in *13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB 2006)*, March 2006, pp. 477–480.

[30] L.-C. Cui, Z.-F. Zhao, X.-J. Xu, F.-M. Wu, and W.-Z. Shan, "Real time performance analysis of CAN bus based on TimeNET," in *ICICIC '08: Proceedings of the 2008 3rd International Conference on Innovative Computing Information and Control*. Washington, DC, USA: IEEE Computer Society, 2008, p. 191.

[31] J. Freiheit and A. Zimmermann, "Extending a response time approximation technique to colored stochastic Petri nets," in *Proc. 4th Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS)*,, College of William and Mary, Williamsburg, VA, USA, Sep. 1998, pp. 67–71.

[32] J. D. C. Little, "A proof of the queuing formula $l = \lambda w$," *Operations Research*, vol. 9, pp. 383–387, 1961.