Combinatorial Modeling Techniques in *Conjoint Simulation*

Axel Hein and Wolfgang Hohl
Institute for Computer Science III (IMMD III)
University of Erlangen-Nürnberg
Martensstr. 3
D - 91058 Erlangen, Germany
e-mail: alhein@immd3.informatik.uni-erlangen.de

1. Introduction

The *performability analysis* of computer systems is a complex and important issue. Massively parallel computer systems designed for the computation of critical and time-consuming applications have to provide *fault-tolerance mechanisms* in order to tolerate failures of components and to continue working even with deteriorated performance. Sophisticated and enhanced modeling tools and techniques are required to evaluate computer systems as soon as possible during the early design stages. We present a modeling framework called *Conjoint Simulation* [9], and put the focus on the use of combinatorial modeling techniques which are well known and frequently used in dependability analysis.

Conjoint Simulation strives to fulfill some of the requirements which have been raised in a survey of performability analysis [13], and which might play an important role in the future of the combined evaluation of performance and dependability. For instance, the improvement of the model construction interface is demanded by applying techniques which are familiar to system designers; the *design-oriented model* and the *evaluation-oriented model* are distinguished, where the first one is automatically converted into the latter representation form. *Design-oriented models* are close to the designer's knowledge and abstraction, whereas *evaluation-oriented models* are representation forms which can be directly evaluated.

2. *Conjoint Simulation*

We refer to the technique of combining object-oriented, process-based simulation models with Petri net and combinatorial dependability models as *Conjoint Simulation* to distinguish it from other *hybrid modeling* approaches. Hybrid approaches typically simplify a detailed model and combine simulation and analytical techniques to reduce the time needed to evaluate a model; these techniques provide an approximation of the results obtained from a detailed model. The primary objective of *Conjoint Simulation*, however, is to facilitate the representation of a system in detail so as not to abstract away system characteristics that are crucial to the performability of the system.

To design a *Conjoint Simulation* model, we partition the target computer system into four domains: architecture, workload, failure modes, and repair-maintenance mechanisms. These four characteristics cannot easily be integrated and varied in a monolithic model. Having investigated various modeling and representation techniques, we feel there is no single modeling technique that facilitates the representation of these four key characteristics in an optimal and easy-to-handle manner. Therefore, different modeling techniques, which are well-suited to represent the various parts of the system, should be used and combined to form the overall model.

With *Conjoint Simulation*, the designer is encouraged to develop parts of the system model independently and with different techniques. In fact, the system model consists of an *architecture-workload model* (*AWM*) and a *failure-repair model* (*FRM*). Breaking down the system model into an *AWM* and a *FRM* simplifies the representation and allows separate, independent evaluation of the two models. The *AWM* can be evaluated to determine the *performance* characteristics of the target system, and the *FRM* can be analyzed to investigate its *dependability*. Finally, the two models are combined to conduct *performability analysis*. Yet another advantage is that a given *AWM* can be evaluated with various *FRM*s, and vice versa. Usually, the *AWM* is the more complex part of the system. By separating the failure, repair, and maintenance characteristics into a separate *FRM*, the user can readily experiment with different *FRM*s without any changes to the *AWM*.

The *AWM* is based on object-oriented modeling and process-based simulation, while we use timed Petri nets and

combinatorial models such as series-parallel diagrams and fault trees to construct the *FRM*. We use a version of timed Petri nets where time is assigned to the transitions of the Petri net (*TTPN* - timed-transition Petri net); these *TTPN* correspond to the well-known *GSPN* (generalized stochastic Petri net [12]) apart from the fact that also non-exponential distribution functions are allowed.

A simple model of a computer system is shown in Figure 1 illustrating the basic modeling concepts of the *AWM*. The model consists of two processor objects prc_1 and prc_2 and a link object lnk_1. The link object simulates a generic link through which data are passed between a source and a destination. A simulation process is assigned to each processor object to perform the scheduling of workload processes (c_{10} and c_{20} in Figure 1). The simulation process c_{30} of the link object models the forwarding of messages sent between prc_1 and prc_2. The workload is represented by the two workload processes c_{11} and c_{21} which are assigned to the processor objects prc_1 and prc_2, respectively; c_{11} and c_{21} may model data processing such as numerical algorithms as well as the sending and receiving of messages.

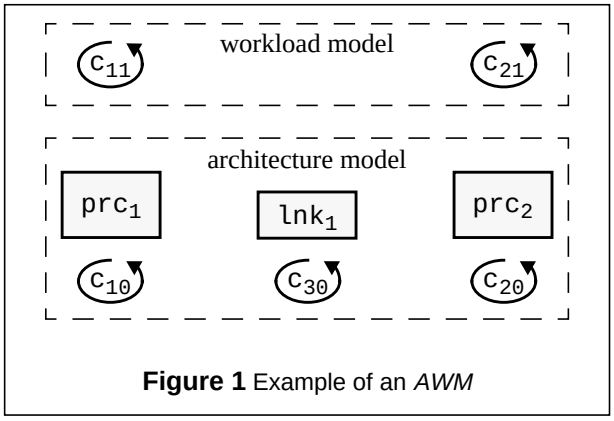

Figure 1 Example of an *AWM*

The interaction between *AWM* and *FRM* is based on the events of the *TTPN* model, that means on the enabling or firing events of transitions in the *TTPN* (an example is given in the paragraph below and in Figure 2). Therefore, two sets C^A_i and C^F_i are assigned to each transition t_i of the *TTPN* representation of the *FRM*. The sets C^A_i and C^F_i contain the operations which are performed in the *AWM* as soon as transition t_i is enabled or fires, respectively. Thus, we specify methods of the *AWM* which are invoked as soon as a transition is enabled or fires. It should be noted that these operations performed in the *AWM* can not only start activities to influence and alter the behavior of the *AWM*, but they can also query the status of the *AWM* in order to control and modify the *FRM*.

The *FRM* is typically represented as a *TTPN*. The example in Figure 2 shows a *TTPN* whose transitions control and trigger activities of the *AWM* (in the figures, thin bars represent immediate, i.e. timeless, transitions, and filled boxes symbolize timed transitions). As soon as the timed transition t_{inj} is enabled the applications of the workload model are started or, if the applications have been interrupted by an error or recovery mechanism, the applications are rolled back and are restarted from the last valid checkpoint. After a period of time defined by the distribution function f_{inj} of the timed transition t_{inj}, an error is injected into one of the processor objects of the *AWM*. After the firing of t_{inj}, the timed transition $t_{recover}$ is enabled and the applications of the workload part of the *AWM* are stopped. The distribution function $f_{recover}$ assigned to $t_{recover}$ defines the time between enabling and firing of $t_{recover}$; when $t_{recover}$ fires the faulty component of the *AWM* is replaced and the *AWM* is reconfigured. Finally, t_{inj} is enabled again.

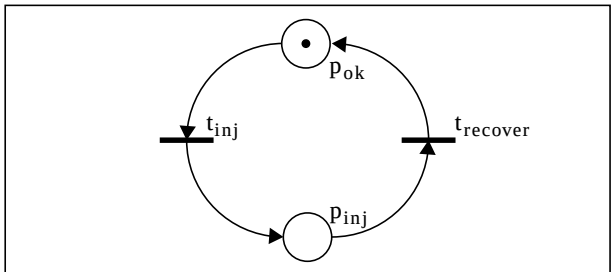

Figure 2 *TTPN* example

3. Combinatorial modeling techniques in *Conjoint Simulation*

Petri nets are the key modeling technique of the *FRM*; the development of large-scale and complicated Petri net models, however, is cumbersome and error-prone. Places, transitions, and arcs have to be correctly positioned to depict the logical relationships of the target system. For quantitative analysis, time behavior has to be assigned to the transitions, and the proper memory policies have to be chosen to model the dynamic and temporal characteristics. Therefore, we use an approach which is intended to retain the flexibility and the modeling power of Petri net modeling, and to facilitate the model design and model development for dependability analysis. For this objective, we use the graphical representations of *series-parallel diagrams* and *fault trees*. An interesting discussion of various methods used in dependability modeling has been presented in [10], where the modeling power of combinatorial and Petri net techniques is compared.

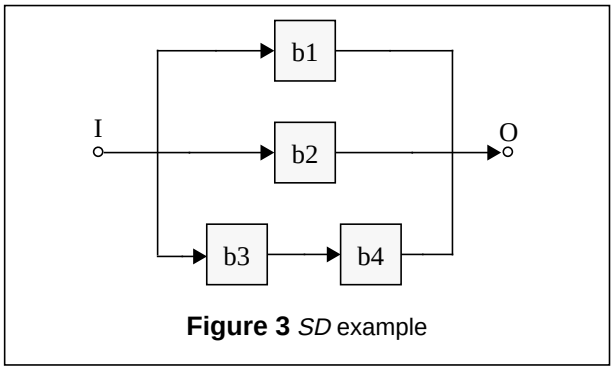

Figure 3 *SD* example

3.1 Conversion of combinatorial model types to Petri net representations

Series-parallel diagrams (*SD*) and *fault trees* (*FT*) are widely used techniques for the dependability analysis of non-repairable systems [15]. Typically, modeling tools providing methods for the analysis of these kinds of models do not consider repair. For instance in the modeling tool *SHARPE*, the analysis techniques for *SD* and *FT* models are specialized for reliability analysis and do not allow components to be repairable [14]. Only under very restricting conditions, *SD* and *FT* models can be used for the modeling of repairable systems [15]. Since we are using simulation to evaluate *Conjoint Simulation* models, we are able to take various distribution functions - such as deterministic time, Weibull or Normal distribution - to define time spans.

The equivalent *SD* and *FT* representations of a simple example comprising four components b1, b2, b3, and b4 are shown in Figure 3 and Figure 4. The system is in the *failed* state if b1, b2, and b3 or b4 are in the failed state.

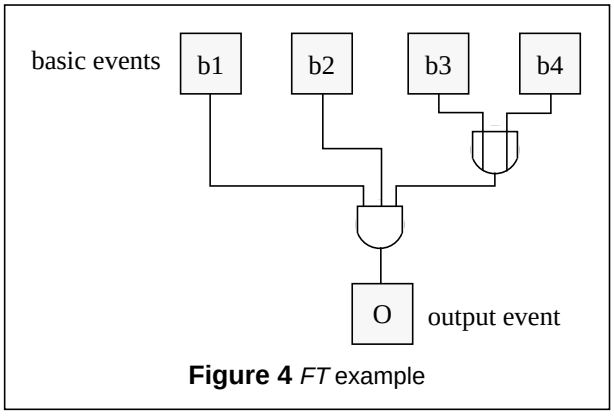

Figure 4 *FT* example

Because of the correspondence of *SD* and *FT* techniques, we refer to both, the blocks in *SD* models and the basic events in *FT* models, as *blocks*. Now, each block has the following attributes:

- A *failure distribution function* is assigned to each block defining the time to failure of this block.
- Each block is either repairable or non-repairable. In the first case, a *repair distribution function* for the time to repair is defined.

Since *SD* and *FT* techniques are well known and widely used in industrial applications, and the more powerful Petri net techniques are applied almost exclusively in academia, we take *SD* and *FT* techniques as the interface for model construction, and provide simulation-based evaluation techniques on the basis of Petri net models, which are automatically generated and are not visible to the model designer. For the ease of *FRM* modeling, we have implemented methods in the modeling environment *SimPar* providing a graphical interface for the development of *SD* and *FT* models, and automatically convert these models in corresponding *TTPN* representations [1], [4]. Furthermore, we extend the *SD* and *FT* representations by allowing shared repair facilities and repair policies such as fcfs (first come first served); these extensions are defined at the *SD* or *FT* level, but are evaluated exclusively in the Petri net representations [1]. Similar approaches have been discussed in [3] and [11]. Ereau et. al. use *synchronized Petri nets* in order to model in a *FT* local or global maintenance and further maintenance dependencies such as shared repair facilities [3]. An approach to incorporate various repair scenarios in *FT* models with repeated events is presented in [11].

The basic principle of the transformation from *SD* and *FT* models into *TTPN* models is the identification of the possible model states and the transitions between the states. Henceforth, we concentrate on the generation from a *SD* model without any loss of generality because of the correspondence of series diagrams and *OR* gates, as well as of parallel diagrams and *AND* gates. Obviously, each block, each series diagram, and each parallel diagram is in one of the two possible states, *failed* and *ok*. These states and the according transitions have to be depicted in the *TTPN*, including the temporal characteristics. *TTPN* modules are automatically generated for each block, each series diagram, and each parallel diagram of a given *SD*, and these modules are connected by drawing arcs and transitions between them. The *TTPN* modules of a block, a series diagram, and a parallel diagram are shown in Figure 5, Figure 6, and Figure 7.

Since each block is either *failed* or *ok*, only one of the two places p^b_{fail} and p^b_{ok} contains a token at any point in time. The timed transition t^b_{fail} models the change of the state *ok* to the state *failed*, and also represents the time to failure. The distribution function assigned to t^b_{fail} is the *failure distribution function* of the block. Accordingly, the timed transition t^b_{rep} models the time till the repair of a faulty block. t^b_{rep} is included only if the corresponding block is repairable; in this case, the *repair distribution function* of the block is assigned to t^b_{rep}. The marking of the

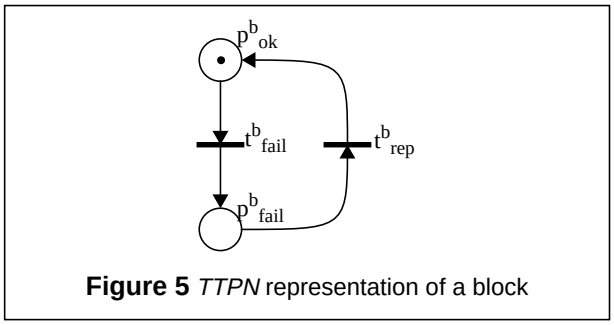

Figure 5 *TTPN* representation of a block

TTPN module corresponds to the actual state of the block of the *SD*, i.e., a token in p^b_{ok} indicates the *ok* state of the block, whereas a token in p^b_{fail} represents its *failed* state.

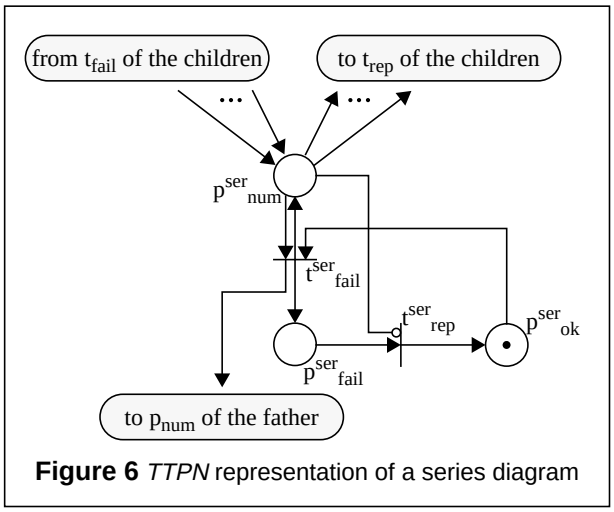

Figure 6 *TTPN* representation of a series diagram

A series diagram is considered *ok* if all of its children are *ok*. Therefore, a token is removed from the place p^{ser}_{num} when a child is repaired (Figure 6). If only one child has been *failed* and is repaired, the series diagram changes to the *ok* state. In this case, the immediate transition t^{ser}_{rep} fires, the token in p^{ser}_{fail} is taken away, and a token is deposited in p^{ser}_{ok}. The inhibitor arc between p^{ser}_{num} and t^{ser}_{rep} prevents t^{ser}_{rep} from being enabled as long as at least one child is *failed*. The arc from t^{ser}_{fail} to p^{ser}_{num} ensures that the number of tokens in p^{ser}_{num} remains equal to the number of *failed* children.

The *TTPN* module of a parallel diagram is similar to the one of a series diagram. Since a parallel diagram is in the failed state if all af the n children are failed, the arcs between p^{par}_{num} and t^{par}_{fail} as well as the inhibitor arc between p^{par}_{num} and t^{par}_{rep} have multiplicity n (Figure 7).

3.2 Conversion of the interactions between *AWM* and *FRM*

If the *FRM* of a *Conjoint Simulation* model is defined as a *SD*, operations are assigned to the blocks, series diagrams,

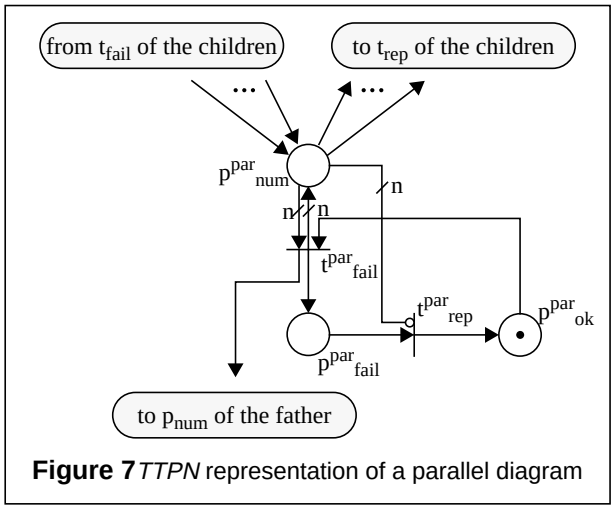

Figure 7 *TTPN* representation of a parallel diagram

and parallel diagrams; these operations launch activities in the *AWM* when the corresponding unit fails or is repaired. Operations in the *AWM* can be triggered when a block in the *SD* (or *FT*) fails or is repaired. Additionally, the failure and repair of a series diagram or of a parallel diagram (or of an AND gate or OR gate respectively) may launch operations in the *AWM*; for instance, the failure of a parallel diagram may cause the start of specific recovery processes in the *AWM*, or the repair of a series diagram allows a workload application to be restarted on the processor objects whose failure states are represented by the blocks of the series diagram.

Since the evaluation of the *Conjoint Simulation* model is performed at the *TTPN* level, the assignment of the operations has to be translated to the *TTPN* representation. During conversion of the *SD* or *FT* representation, the operations are automatically assigned to the transitions of the generated *TTPN*, which model the failure or repair of the respective unit.

For instance, the operations to perform in the *AWM* which are assigned to the series diagram of a *SD* model are transferred to the *TTPN* module of this series diagram (Figure 8). The operations which are triggered when the series diagram fails are assigned to the set $C^F_{fail(ser)}$ of the transition t^{ser}_{fail}. Equivalently, the operations which are launched when the series diagram is repaired are assigned to the set $C^F_{rep(ser)}$ of the transition t^{ser}_{rep}.

4. Conclusion

In this paper, we have presented a modeling approach which enables the model designer to specify the failure-repair behavior in the *FRM* as well as the interactions between *AWM* and *FRM* at the very descriptive level of *SD* and *FT* models. The conversion of the *SD* and *FT* representations of the *FRM* to the *TTPN* representation is automati-

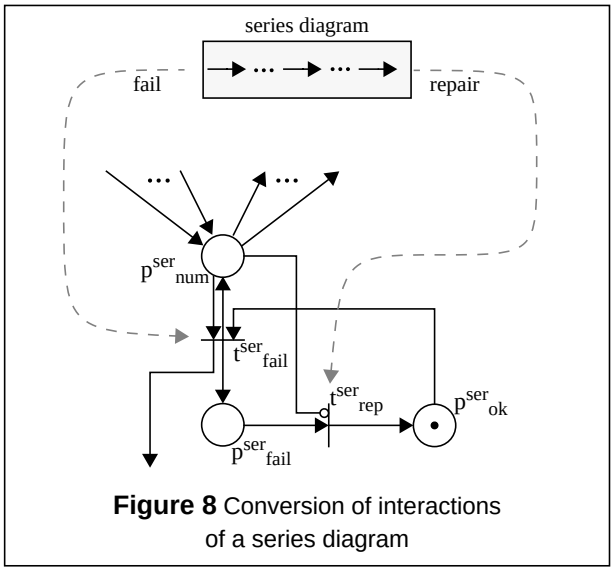

Figure 8 Conversion of interactions of a series diagram

cally performed; the Petri net representation is immediately linked to the *AWM* during model evaluation, or the model designer may modify and extend the Petri net performing a gradual refinement of the *FRM*.

The modeling techniques of *Conjoint Simulation* including the combinatorial model types are implemented in the modeling environment *SimPar* [6] which has been used for the performance and dependability evaluation of various multiprocessor systems ([2], [5], [7], [8]).

Abbreviations

AWM	Architecture-Workload Model
FRM	Failure-Repair Model
FT	Fault Tree
SD	Series-Parallel Diagram
TTPN	Timed-Transition Petri Net

Literature

[1] Bänsch, Klaus: *Hybrid-Simulation mit SimPar*, Diplomarbeit (Master Thesis), IMMD III, University of Erlangen-Nürnberg, 1996.

[2] Dalibor, Stefan, Axel Hein, and Wolfgang Hohl: *Simulation-Based Performability Evaluation of Fault-Tolerant Multiprocessors* in Proceedings of the ESM '95, European Simulation Multiconference, Prague (Czech Republic), June 1995, pp 699-703.

[3] Ereau, Jean-François, Hamid Demmou, and Malecka Saleman: *Dynamic Fault Trees Based on Synchronized Petri Nets* in Proceedings of ESS '95, European Simulation Symposium, Erlangen (Germany), October 1995, pp 253-257.

[4] Grams, Boris: *Entwurf und Implementierung praxisnaher Methoden zur Dependability-Analyse basierend auf stochastischen Petri-Netzen*, Diplomarbeit (Master Thesis), IMMD III, University of Erlangen-Nürnberg, 1995.

[5] Hein, Axel: *A Process-Oriented Simulation Model for the Performability Analysis of a Fault-Tolerant Multiprocessor System* in Proceedings of the ESS '94, European Simulation Symposium, Istanbul (Turkey), October 9-12, 1994, Volume II, pp 215-219.

[6] Hein, Axel and Klaus Bänsch: *SimPar - A Simulation-Based Environment for Performance and Dependability Amalysis of User-Defined Fault-Tolerant Parallel Systems*, Internal Report 1/95, IMMD III, University of Erlangen-Nürnberg, 1995.

[7] Hein, Axel, Klaus Bänsch, and Martin Ellermeier: *A Scalable Model of the Parystec GC Series and an Approach for the Modelling of Fault-Tolerant Parallel Systems*, FTMPS, Esprit Project FTMPS 6731, IMMD III, University of Erlangen-Nürnberg, February 1995.

[8] Hein, Axel and Martin Ellermeier: *A Scalable Model of the Power GC Series*, FTMPS, Esprit Project FTMPS 6731, IMMD III, University of Erlangen-Nürnberg, August 1995.

[9] Hein, Axel and Kumar K. Goswami: *Combined Performance and Dependability Evaluation with "Conjoint Simulation"* in Proceedings of the ESS '95, European Simulation Symposium, Erlangen (Germany), October 26-28, 1995, pp 365-369.

[10] Malhotra, Manish and Kishor S. Trivedi: *Power-Hierarchy of Dependability-Model Types* in IEEE Transactions on Reliability, Vol. 43, No. 3, September 1994, pp 493-502.

[11] Malhotra, Manish and Kishor S. Trivedi: *Dependability Modeling Using Petri-Nets* in IEEE Transactions on Reliability, Vol. 44, No. 3, September 1995, pp 428-440.

[12] Marsan, M. Ajmone, G. Balbo, G. Chiola and G. Conte: *Generalized Stochastic Petri Nets Revisited: Random Switches and Priorities* in Proceedings of the International Workshop on Petri Nets and Performance Models, Madison, WI, USA, August 1987.

[13] Meyer, John F.: *Performability: a Retrospective and Some Pointers to the Future* in Performance Evaluation, 14/1992, pp 139-156, Amsterdam North-Holland, 1992.

[14] Trivedi, Kishor S. and Robin A. Sahner: *SHARPE: Symbolic Hierarchical Automated Reliability and Performance Evaluator*, Introduction and Guide for Users. Gould CSD, Urbana, and Dept. of Computer Science, Duke University, September 1986.

[15] Villemeur, Alain: *Reliability, Availability, Maintainability and Safety Assessment*, Vol.1 Methods and Techniques, Vol.2 Assessment, Hardware, Software and Human Factors, Chichester, New York, [et. al.] : John Wiley & Sons Ltd, 1992.