Petri Net Based Software Dependability Engineering

Monika Heiner

Brandenburg University of Technology Cottbus
Computer Science Institute

Postbox 101344
D-03013 Cottbus
Germany

mh@informatik.tu-cottbus.de
Tel/Fax: (+ 49 - 355) 69 - 2794

Abstract:
Methods of software dependability engineering can be divided into two groups - methods to improve the software dependability and methods to predict the reached degree of software dependability. Among those methods, which aim at the improvement of software dependability, the Petri net based validation techniques to avoid faults during the development phase have attract a lot of attention in the last years. Within this framework, Petri net models play the role of a common intermediate software representation, from which different validation techniques are able to start - qualitative as well as quantitative ones. Based on this experience, the approach to integrate different methods on a common representation is extended by a formal method to derive Petri net models suitable for a structure-oriented reliability prediction.

1 Motivation

Starting from the taxonomy of dependability introduced in /Avizienis 86/, methods of software dependability engineering can be divided into two groups - methods to improve the software dependability by fault avoidance or fault tolerance (procurement) and methods to predict the reached degree of software dependability (assessment). Among those methods, which aim at the improvement of software dependability, different kinds of Petri net based validation techniques to avoid faults during the development phase have attract a lot of attention in the last years. A classification of software validation techniques should separate the validation methods (main principles) on the one side and the properties to be validated on the other side /Heiner 92/. These software validation techniques should then be embedded in a process model to develop software with high dependability demands. A suitable order of validation methods takes into account that

- validation should be applied as early as possible,

- the proper functionality is a prerequisite for an evaluation of quantitative properties, and
- the expected functionality can only be guaranteed in any case if all consistency conditions of context checking have been fulfilled.

Within this general framework a Petri net based methodology of software dependability engineering can be outlined. The approach combines qualitative analysis (1), monitoring and testing (2) as well as quantitative analysis in terms of performance evaluation/prediction (3) and reliability prediction (4) on the basis of a common Petri net-based intermediate representation of the parallel/distributed software system under consideration.

A sketch of the main principles which are common to the first three points is given in the next section, while the last point is discussed in the next but one section.

2 Net Based Methods to Improve Dependability

Different validation methods may require net models which vary partly in their level of abstraction. This variety comprises not only such typical quantitative parameters as delay and branching information (which are obviously necessary in case of quantitative analysis), but also the granularity of considered control and/or data flow, i.e. the degree of details concerning structural information. Therefore, in order to integrate qualitative as well as quantitative analysis on a common intermediate software representation, an important feature of a related methodology is the ability of a controlled structural reduction, combined with compression of any quantitative parameters.

In /Heiner 95/, a method is outlined how to develop these models step-by-step:

- qualitative models
  - control structure models as place/transition nets
    (Any branching information is neglected. Every structurally possible path of the model is considered to be realizable in the software. The set of execution paths of the model is greater or equal as the software's execution path set.)
  - control flow models as coloured nets
    (All control variables determining the actually control flow are added to the control structure model. So, the control flow model is generally much greater than the control structure one. But model and software have the same set of execution paths.)
- quantitative models
  - performance models as timed, interval or stochastic Petri nets
    (depending on the type of performance measures to be evaluated)
  - reliability models as stochastic Petri nets

All transformations (from the parallel software system description into a first qualitative Petri net model, and between the different kinds of net models) can be done formally, and therefore automated to a high degree.

Obviously, the individual validation techniques are essentially influenced by the analyzing possibilities available for the corresponding net classes.

(1) The validation of qualitative properties comprises two steps. At first, the context checking of general semantic properties (basically liveness properties) is done by a suitable combination of static and dynamic analysis techniques of (classical) Petri net theory. Afterwards, the verification of well-defined special semantic properties (among them safety properties) given by a separate specification of the required functionality is performed. During this second step, the power of classical Petri net theory is supplemented by the model checking approach, using temporal logic as a flexible query language for asking questions over the (complete/reduced) set of reachable states.

(2) The objectives of the monitoring and testing component are twofold. Besides the provision of the quantitative attributes according to the user-driven time abstraction level, the net-based testing method supports a systematic test of parallel systems. Techniques to derive automatically dedicated test suites and to measure the test coverage obtained are important features of this systematic testing.

(3) The validation of quantitative properties has to be based on quantitative net models. The frequency and maybe also delay attributes necessary to transform qualitative models into quantitative ones are provided by the monitoring and testing component or, in simple cases, calculated from the basic instruction sequences. The available Time Petri net classes differ essentially in the provided time concept (timed nets: constant delay, interval nets (usually called time nets): interval delay, stochastic nets: different kinds of probability distributions of the delay, or combinations of them). The choice of a suitable net class should be guided by the well-known engineer's basic principle to keep everything as simple as possible. So the answer depends on the properties to be validated.

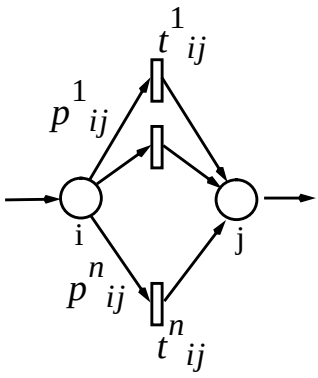
As long as there are hard deadlines to meet definitely, e.g. as it should be the case for systems with predictably timing behaviour, the exact evaluation by timed or interval nets is unavoidable. If average or probability distributions of performance measures like load, throughput, utilization etc. are wanted, then the application of stochastic Petri nets becomes useful.

In /Heiner 94/, an example is given how to obtain the quantitative model from the qualitative one by quantitative expansion and property-preserving structural compression using the so-called locally Markovian Object Nets (MONs) as stochastic net class. Conflict clustering within the underlying net and a concept for time abstraction of sequential software parts are crucial points of this transformation. General rules are given in those paper for a suitable decomposition of Petri nets into objects or to find suitable time abstractions, respectively.

3 Net Based Method to Predict Dependability

Based on this experience, we are now going to extend the approach to integrate different methods on a common representation by introducing an approach to structure-oriented reliability prediction inspired by /Roca 88/. This is intended as a first step in the direction

Figure 1: Input parameters of reliability evaluation.



in failure free case:

$$\sum_{k=1}^{n} p^k_{ij} = 1$$

in case of failure

$$\sum_{k=1}^{n} p^k_{ij} < 1$$

probability of unsuccessfully executing the path (i,j)

$$1 - \sum_{k=1}^{n} p^k_{ij}$$

notation:
pij - probability of entering the path (i,j) and executing the path successfully,
tij - (average) time to execute the path (i,j)

to incorporate further dependability measures too. Again, the model which we need as reference point for the evaluation to be done, should not be built from the scratch, but instead of this, the dependability model should be derived to a high degree automatically from that net models which we do have due to our validation efforts. Because we already know, how to map software onto (place/transition) Petri nets, we have then altogether a formal method to derive systematically Petri net models suitable for dependability prediction of the software under consideration.

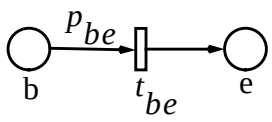
The quantitative parameters are now, in case of dependability prediction (see Figure 1),

- the probability of entering and executing successfully a given path and
- the (average) time to execute a given path.

The execution time can again be measured by the testing and monitoring component or, in special cases, calculated from instruction sequences. The probability of unsuccessfully executing a path (i,j) depends on the proportion of bugs in the total program. This proportion could be assessed from experience, e.g. by testing software of the same characteristics and size developed by the same programming team.

The method proposed consists basically of a set of rules describing the allowed structural reductions and the corresponding transformation rules of the quantitative parameters within any well-structured sequential substructures (see Figure 2).

Provided there are no information about the probability distributions of any inputs, then the assumption is justified that the probability of entering the path (i,j) is equal for all program branches. In that case, the total failure (hazard) rate *z* of the given software can be computed immediately for a well-structured sequential program which has been completely

reduced to 

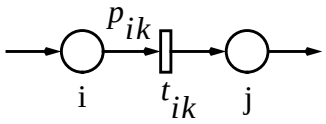
by $z_{be} = \frac{1 - p_{be}}{t_{be}}$.

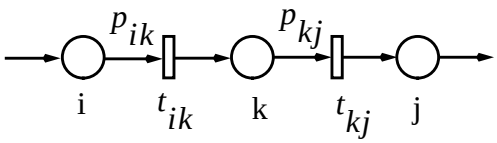
Generally, in case of parallel programs, the failure rate has to be evaluated by suitable stochastic Petri net evaluation tools. But a structural reduction combined with compression of quantitative parameters, done before as strong as possible, may reduce the computational costs essentially.

4 Final Remarks

The evaluation of the stochastic software model by conventional stochastic Petri nets tools requires a model transformation to maintain the right conflict solution strategy (see Figure 3). At first, the branching of control flow has to be decided, and afterwards the time consumption (of any sequential program parts) may take place. Obviously, this transformation comes along with introducing a lot of immediate transitions causing again many transient reachability states. Because these transient states are useless from the practical point of view it would be worth thinking over how to avoid them.

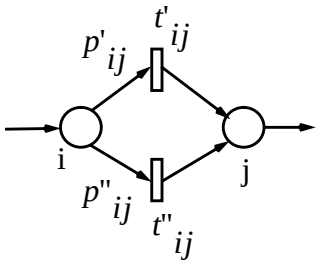
Figure 2: Structural compression within sequential parts.





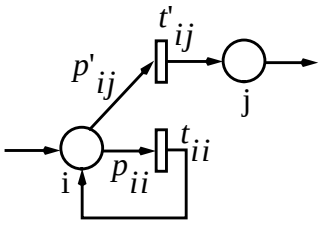
$$p_{ij} = p_{kj} \cdot p_{ik}$$

$$t_{ij} = t_{ik} + t_{kj}$$



$$p_{ij} = p'_{ij} + p''_{ij}$$

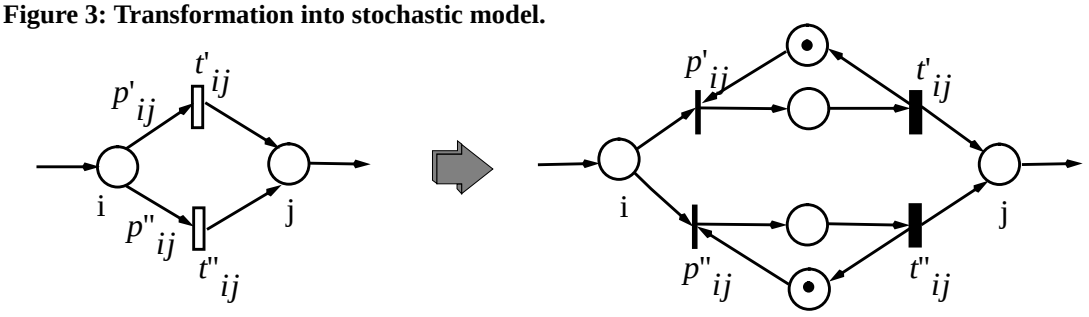
$$t_{ij} = \frac{p'_{ij} \cdot t'_{ij} + p''_{ij} \cdot t''_{ij}}{p'_{ij} + p''_{ij}}$$



$$p_{ii} = \frac{p'_{ij}}{1 - p_{ii}}$$

$$t_{ij} = t'_{ij} + \frac{p_{ii} \cdot t_{ii}}{1 - p_{ii}}$$

Figure 3: Transformation into stochastic model.



notation:

- immediate transitions
- deterministically delayed transitions with reservation of marks
- deterministically delayed transitions, no reservation of marks

The intended extension to other dependability measures requires more sophisticated fault models, which have to be added as environment assumption to the total evaluation model. To support the user, a library of appropriate Petri net components for different fault models would be useful.

5 References

/Avizienis 86/

Avizienis, A.; Laprie, J.-C.:
Dependable Computing: From Concepts to Design Diversity;
Proc. of the IEEE 74(86)5, pp. 629-638.

/Heiner 92/

Heiner, M.:
Petri Net Based Software Validation - Prospects and Limitations;
Techn. Report ICSI Berkeley/CA, TR-92-022.

/Heiner 94/

Heiner, M.; Wikarski, D.:
An Approach to Petri Net Based Integration of Qualitative and Quantitative Analysis of Parallel Systems;
Techn. Report BTU Cottbus, I-09/1994.

/Heiner 95/

Heiner, M.:
Petri Net Based Software Dependability Engineering - a Case Study;
to appear as Tutorial materials at ISSRE ’95 (The 6th Int. Symposium on Software Reliability Engineering), Toulouse, Oct. 1995.

/Roca 88/

Roca, J. L.:
A Method for Microprocessor Software Reliability Prediction;
IEEE Trans. on Reliability 37(88)1, pp. 88-91.