USING SIMULATION TO PREDICT QUALITY AND COST IN THE AUTOMOTIVE BUSINESS

Dr. Stefan Greiner
DaimlerChrysler Research
Dept. RIC/AS
70546 Stuttgart, Germany
HPC T 728
E-mail: stefan.greiner@daimlerchrysler.com

KEYWORDS

System modeling, parameter estimation, cost-and quality analysis.

ABSTRACT

Simulation in industry has reached a very high popularity. Especially in the automotive industry, simulation is used very heavily when developing new vehicle concepts (e.g. crash test simulation). Typically, continuous simulation is used for these types of applications.
On the other hand, discrete simulation is becoming more and more popular, because many problems can be mapped to discrete models. Consider, for example, the question of how many breakdowns for a fleet of vehicles can be expected during their warranty/courtesy period. In this case, we have a continuous time frame and discrete events, namely the breakdown of a vehicle. Before we can analyze a system, we have to set up a model for this system/scenario. Therefore, we need a modeling paradigm that is both easy to understand and well suited for numerical analysis. At DaimlerChrysler, we use the modeling paradigm of stochastic Petri Nets, which we have extended in order to fit our specific modeling needs. These extensions result in models that do no longer have the Markovian property and therefore a closed system solution was not longer easily computable. On the other hand, discrete system simulation was perfectly suited to solve these types of extended Petri Net models. Before we can analyze the model, using simulation, we have to parameterize the corresponding model. In industrial applications, we have very often field data available which need to be analyzed statistically to obtain the corresponding model parameters. The parameterized model can then finally be analyzed to obtain the results.
Based on different research activities at DaimlerChrysler, the Petri net modeling and analysis tool *Expect* was developed.
During the talk, the modeling paradigm Petri nets is explained. Then, it is shown how to obtain model parameters from field data, and finally, an exemplary analysis is performed, using the analysis tool Expect. Different modeling and analysis examples are discussed and demonstrated. Together with numerical examples, an introduction to the analysis tool *Expect* is provided. To obtain the model parameters, the tool ParEs is used.

PETRI NET MODEL (GSPN)

A Generalized Stochastic Petri Net (GSPN) is a state-transition system, where the transitions are assigned firing times. The set of all possible states in the Petri net is called the state space of the Petri net. Only in the case where the Petri net consists only of exponential- and timeless transitions, the state space can be solved in a closed form by setting up the balance equations and solving them. In the case of a transient analysis we obtain a system of linear differential equations to be solved and in the case of a steady state solution, the system of differential equations is reduced to a linear system of equations.
Especially in industrial applications, we have very often non-exponential distributions assigned to the transitions. Consider for example the mileage behavior of a vehicle. In this case it turns out that the underlying distribution has a log-normal nature. Or, in the case of modeling the failure behavior of mechanical components, the underlying distribution usually shows some type of Weibull nature. We have either an infant mortality failure behavior (high failure rate at the beginning and then a decreasing failure rate over time), a wear-out behavior (low failure rate at the beginning and then an increasing failure rate over time), or a mixture of both types. Electronic components usually show a random failure behavior. Taking this into account, a numerical solution of the Petri net by solving the flow equations is usually not possible. Therefore, we turn very often to system simulation to

Proceedings 18th European Simulation Multiconference
Graham Horton (c) SCS Europe, 2004
ISBN 3-936150-35-4 (book) / ISBN 3-936150-36-2 (CD)

solve a Petri net to obtain the corresponding performance, reliability, quality and cost measures.
In the following, we will introduce common types of distribution functions and how to obtain the parameters of a distribution function from field data in order to parameterize the corresponding Petri net models.

PARAMETER ESTIMATION

While stochastic Petri nets have become a widely used means for modeling complex systems, problems already arise in practical applications when the transitions in a net have to be parameterized. The goal of this paragraph is to introduce techniques, that allow the estimation of the parameters of several lifetime-distributions from field-data. These parameters serve then as input for the Petri nets. Besides the two- and tree-parameter Weibull distribution that are used in traditional reliability analysis, the exponential distribution, normal and logarithmic normal distribution and a new distribution-type, referred to as bathtub distribution, can be handled. The algorithms used for parameter estimation rely on traditional methods as regression and Maximum-Likelihood-Estimation, employing local and global optimization techniques.

THE WEIBULL FAMILY

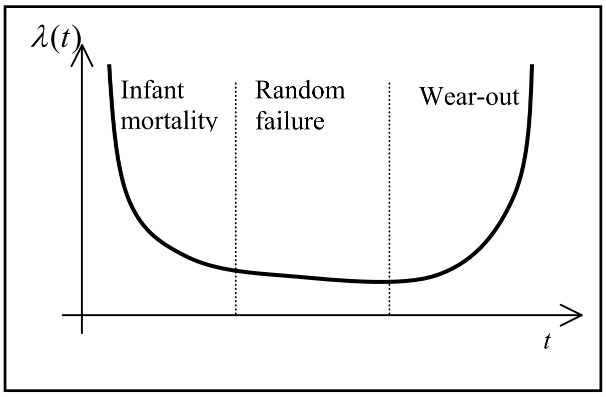
The Weibull distribution, introduced by Waloddi Weibull in 1937 [Abe94] is the most frequently used distribution in reliability engineering. The original Weibull distribution has the distribution function (CDF):

$$F(x) = 1 - e^{-(\frac{x}{\alpha})^{\beta}}$$

with the two parameters α and β (which we therefore refer to as the two-parameter Weibull distribution). α is called the characteristical lifetime, which is defined as the point, where F(α) $\approx$ 0.632 (i.e. 63.2% of all units have failed. The parameter β is referred to as the shape parameter, because it defines, whether the distribution models infant-mortality failures (β < 1) or wear-out failures (β > 1). In the case of β =1 we have the exponential case, i.e. random failure.

THE BATHTUB DISTRIBUTION

This distribution models the whole lifetime of a component with infant-mortality- random-and wear-out-failures. It is obvious from the previous paragraph, that this distribution can be built from the superposition (weighted sum) of two Weibull and one exponential distributions. The name bathtub distribution is derived from the curve of the hazard rate of this distribution which gives the well known bathtub shape as shown in the following picture:



But doing this, one can see that, with one exception, the exponential distribution, which should explicitly model the stable lifetime, has nearly no influence on the shape of the distribution. The reason for this is that the exponential part is implicitly contained in the superposition of the two Weibulls. Therefore, for practical reasons we decided to leave the exponential part out to reduce the number of parameters to be estimated from eight to six (for clarity: this is not because we consider the exponential part as unimportant but because its explicit representation is superfluous). The CDF of the bathtub distribution is therefore given by the following equation (please note that the parameter *p* determines to what extent the current distribution function is subject to infant mortality or subject to wear out):

$$F(x) = p * (1 - e^{-(\frac{x}{\alpha_1})^{\beta 1}}) + (1-p) * \begin{cases} 1 - e^{-(\frac{x - x0}{\alpha_3 - x0})^{\beta 3}} & x > x0 \\ 0 & x \leq x_0 \end{cases}$$

The two parameter Weibull is used to model infant mortality failures and the three-parameter Weibull models wear-out-failures and *p* (0 $\leq$ p $\leq$ 1) defines the weight of the terms (note: the index 2 has been omitted intentionally to remember the reader, that the explicit representation of the exponential part is missing). The exceptional case mentioned above in which the influence of the exponential part would be visible is the case if there are no wear-out-failures. But in this case, the second Weibull distribution can be used with β_3 = 1 to model the exponential part, such that this special case of a constant failure is also covered within the bathtub distribution.

THE NORMAL FAMILY

The distributions of this family, namely the normal and logarithmic normal (lognormal) distribution, are in general not used for the modeling of failure behaviour. The typical use of these distributions is the use as a mileage distribution, whereas the lognormal distribution has become the most commonly used distribution in this application area. The CDF of the normal distribution is given as:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

and that of the lognormal distribution as

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(\ln(t)-\mu)^2}{2\sigma^2}} dt$$

respectively. Hereby, μ is the mean value and σ the standard deviation.

DIFFERENT TYPES OF FIELD DATA

In this paragraph we describe the different modes of data that have to be treated by the estimation algorithms. Most presentations of the algorithms shown later implicitly assume a so-called *full sample*, i.e. all parts fail till the end of the test and the exact times of the failures are known. In reality, this is very rarely the case. One has to deal with clustered and suspended data. Clustered means, that the exact failure times of an error are unknown but one only knows that there were for example k_i errors for parts with a lifetime between x_i and x_{i+1} hours / miles. Suspended means, that some parts survived the end of the test, each of which with a certain lifetime / milage. Certainly, also the information about the suspended elements can be clustered, for example there were l_i parts with an operating time between x_i and x_{i+1} hours / miles, that did not fail. To put it in a nutshell, combining all possible modes of single / clustered data with failures and / or suspensions, there are six combinations that have to be considered in practice (the cases that contain only suspensions and no information about failures are not considered as all following estimation procedures need at least one failure for performing an estimation).
The main advantage of using field data for analysis is, that one can be sure that the data reflect the behaviour of the part under investigation under real conditions. Thus, by using field data, the most critical point of a test rig series -- does the simulated stress reflect the real usage stress -- is circumvented. But this convenience comes at a price: Usually, field data have very poor quality, such that the estimation algorithms must be able to work with this low quality, and commonly contains no information about the milage of the suspended elements, simply because one doesn't go to the garage if there is no error. Thus, in practice, the milage of the suspended elements has to be estimated, too.

LINEAR REGRESSION

In the following, the linear regression method for determining parameters of a distribution function is explained. Since the linear regression method is the simplest type of analysis techniques, we will focus on it. Other techniques as for example the maximum likelihood technique are not explained in detail. For more information on these techniques, the reader is referred to [BGdT98].
The basic idea of linear regression is to transform the CDF of the distribution into a linear form $y = a*x + b$, find a least-squares fit through the failures and finally calculate the distribution parameters from the straight line parameters. This is also the idea behind the well known probability papers. For example, in the case of the two-parameter Weibull distribution with CDF

$$F(x) = 1 - e^{-(\frac{x}{\alpha})^\beta}$$

the linear form can be achieved by taking the natural logarithm twice. This leads to the linear form

$$\ln(\ln(\frac{1}{1-F(x)})) = \beta * \ln(x) - \beta * \ln(\alpha)$$

Now, we define

$$\begin{array}{rcl} y & = & \ln(\ln(\frac{1}{1-F(x)}) \\ a & = & \beta \\ x & = & \ln(x) \\ b & = & -\beta * \ln(\alpha) \end{array}$$

and obtain the linear form *y=ax+b.* One of the problems in performing a least-squares fit to this form is, that one has only the x-values, i. e. the times of failure or suspension, but no y-values. Therefore one has to find an estimation for the correct plotting positions. According to [Abe94] median ranks, which have to be adjusted for the handling of suspended elements, are used for that. For the least-squares solution it is important that there is usually a large error in the time of failure, such that *x* should be taken as the dependent variable in the fit. Finally, the back-transformation of the straight parameters to the distribution parameters in the two-parameter Weibull case is done by the formulas:

$$\alpha = e^{-(\frac{b}{a})}$$

$$\beta = a$$

For the treatment of clustered failures and suspensions, there are different possibilities: For example, in [Lawl82] an algorithm for calculating the ranks in the clustered case is given. In practice, a simpler solution is possible: the failures or suspensions in a cluster are distributed either uniform or normal over the cluster (both versions have been implemented and tested but the differences are neglectable) and then the formulas for the single-data-case are used. Surely, this enlarges the number of data points that have to be considered in the least-squares estimation, but in practice the runtime of the estimation even with thousands of failures is within a few seconds.
The application of regression to the lognormal distribution leads to a numerical problem, as the linear transformation is

$$ierf(2*F(x)-1) = \frac{1}{\sqrt{2}\sigma}\ln(x) - \frac{\mu}{\sqrt{2}\sigma}$$

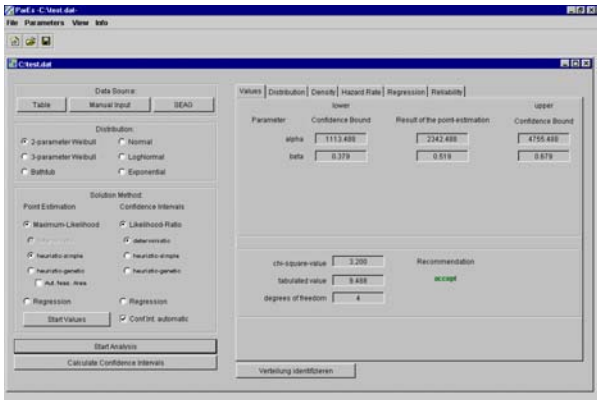
and therefore requires the efficient calculation of the inverse of the Gaussian error function.This can be done starting with the equality

$$erf(ierf(y)) = y \Rightarrow y - erf(ierf(y)) = 0$$

Defining $f(x_n) = y - erf(x_n) = 0$ we must find the zeros of $f(x_n)$ where x_0 can be approximated by one of the known total formulas for *ierf*. The solution follows the idea of Newton's Method [BuFa01] but uses a third-degree Taylor-polynomial instead of one of first order, which leads to faster convergence, and *erfc* instead of *erf* for numerical stability.
Another problem is the use of the three-parameter Weibull distribution with regression as one has to estimate three distribution parameters from two straight-parameters. As the parameters for α and β can be easily estimated by the method described above, it is obvious to separate the estimation of x_0. When using a probability paper it is a hint that there is an x_0 if the plotted points don't lie on a straight line. Hence it is adjacent to use the correlation coefficient, which indicates how well the points fit onto a straight line, for the estimation of x_0, i.e. the failure times have to be shifted such that the correlation coefficient is maximized. A golden-section-search is used for performing the optimization. The shift needed for the maximization is the estimation for x_0 and the estimation of the remaining parameters works on the transformed values.

Parameter Estimation Tool ParEs

This paragraph introduces ParEs, a tool for parameter estimation based on field data. ParEs (Parameter Estimation) is a GUI-based pure java-tool that incorporates all of the algorithms given above to give even the statistically non-experienced user the possibility to estimated parameters from field data and therewith to parameterize the transitions in his Petri net model. In the following, a screenshot of the tool is shown.



Parameter	lower Confidence Bound	Result of the point-estimation	upper Confidence Bound
alpha	1113.488	2342.488	4755.488
beta	0.379	0.519	0.679

chi-square-value	3.200
tabulated value	9.488
degrees of freedom	4

As datasources, the user has the choice between an ODBC-datasource, which is usually a database in the background, the manual input of failures and suspensions in a spreadsheet-like form or the selection of special car-components from a tree-structure. The user has the choice to pre-define a distribution function to be fitted or he can select the automatic detection option. In this case the system determines automatically which distribution function fits the given field data best. After the selection of the estimation-algorithm, the user may insert start values for the estimation, which are by default computed automatically, if someone wishes to fix some parameters to a special value. Finally, one can start either the computation of the point estimation and confidence-interval-calulation altogether or just the point estimation and the confidence-interval-calculation manually afterwards. The reason for this possibility is, that the calculation of the point-estimation is usually very fast while the computation of the confidence-intervals takes a lot of time, such that one should first be able to decide, if the distribution he guessed is correct before the lengthy part of the computation is started. To help the user in the decision if the supposed distribution is right, a Chi-Square goodness-of-fit test is performed, too. The final estimation results are presented to the user as the pure parameter values with the confidence bounds but also the typical plots of CDF, PDF, hazard-rate or the regression-straight line (if regression has been used) are provided. Finally, the resulting parameters are also provided as tagged data to facilitate the use of the component for example by the Petri net tool Expect.

The idea behind this data export facility is, to make the whole parameter estimation process completely invisible for the user in the case of input modeling. Because in this case, users are usually not interested in and are not familiar with the parameter estimation process and its special details but just need appropriate parameters for their model. For example, if one wishes to use a transition in a Petri net to model the failure behavior of a special part, one just has to select this part and need not to know anything about the parameters. All the rest (parameter estimation, data handling and model parameterization) is handled by the tool. For this reason, a whole web based system environment has been set up and ParEs is part of this environment. This environment has been developed in Java and is constantly extended.

In the following, we will demonstrate the accuracy of the methods given above. For a better judgement of the results, no real data were used but data that were generated by a random number generator for the given distribution. For all examples, the confidence level is set to 95%. In the Weibull example, 50 failures have been generated while in the bathtub case 500 failures were used. At a first glance, this seems to be only a few input data if one considers the fact, that in the case of real field data thousands of real vehicles are considered. But, on the other side, one has to consider the fact that the data used in this example are single data while in the case of real field data we have data clusters. Each of these clusters can contain thousands of data points. Thus, even with a small cluster size of only 1000 miles, 500 clusters would define a range from 0 to 500.000 miles which will surely cover the whole lifetime of a vehicle.

The time for the evaluation of the likelihood function depends linear on the number of failures and suspensions in the case of single data and on the number of clusters, otherwise. Thus, one would expect that enlarging the size of the sample will always result in an enlarged runtime in the same order of magnitude. Practically, this can be taken as a worst case estimation because a larger sample usually contains also more information. Thus, fewer iteration steps will be needed if MLE is used such that the increase in runtime will be sub-linear in the average case. As in the regression case, there is a closed form solution where the runtime is within a few seconds even with thousands of failures.

One question that is often asked is, how many failures are needed to perform a reasonable estimation. If using MLE, theoretically one typical failure is enough if there are suspensions. But, if x_0 of the tree parameter Weibull distribution has to be estimated, too, at least 15 failures are needed. Practical applications show that one can expect reasonable estimates if there are at least about 20 failures. But using these values, one has to keep in mind that this always means *representative* failures for the failure mode under investigation and not only random failures.

The first example to be demonstrated is a three parameter Weibull distribution. The following table shows the result of the estimation with the confidence intervals for local optimization with heuristic confidence interval calculation and regression. Hereby, α_l denotes the lower confidence bound for the parameter α while α_u denotes the corresponding upper confidence bound.

Parameter	True Value	MLE	Regression
α_l	---	137407	95435
α	150000	151029	150969
α_u	---	167381	279169
β_l	---	1.572	2.620
β	3.2	3.092	2.710
β_u	---	4.542	2.801
x_{0l}	---	0.736	44859
x_0	40000	40494	44859
x_{0u}	---	69144	44860

The computation of the above results took 18 seconds (point estimation: 3 seconds, confidence interval calculation: 15 seconds), while the regression needed only 4 seconds. There is only a slight difference in the upper confidence bound of β if an evolutionary strategy or the deterministic version is used for calculation. But, in this case the calculation time increases to about 5 minutes (compared to 15 seconds as before).

In the second example we apply our analysis algorithms to a bathtub distribution. The corresponding results are shown in the following table.

Parameter	True value	Lower Bound	Point Estimation	Upper Bound
α_1	3000	981	2026	4707
β_1	0.5	0.38	0.52	0.7
α_3	80000	79378	79944	80548
β_3	4.2	3.64	4.16	4.789
x_0	65000	62562	64645	66032
p	0.15	0.11	0.16	0.22

The computation time for the point-estimation was 29 seconds with the penalty method, and 4 seconds with the simplex method. The confidence intervall calculation took 241 seconds.

THE ANALYSIS TOOL EXPECT

The analysis tool Expect has been developed at DaimlerChrysler [HeGrHo02] and is designed for a broad range of applications, including safety, quality and cost analysis. Safety analysis is concerned with the design of safe automotive systems, including their interactions with humans. In the area of safety analysis, system measures as for example reliability, availability, load and throughput of system components and probabilities for critical states, are determined. Quality analysis is concerned with building models at the component, system and vehicle levels for obtaining information on product quality. One example of this is comparing the behavior of systems built using different components. Cost analysis allows, for example, the calculation of price models for maintenance and service packages, which depends on the level of service offered, as well as the age and mileage of the vehicle. In addition, statistical methods are used to predict the expected quality and cost into the future. In all cases, the goal is that these analyses can be performed as fast and as comfortably as possible, while covering a large number of variants, in order to study and compare different real-life scenarios.

The analysis tool Expect was designed with the goal of offering the modeler an easy to use and understand modeling environment and on the other side, allowing for an integrated system simulation, using discrete time event simulation. In the following, a short excerpt of the capabilities of Expect is described.

TOOL FUNCTIONALITY

The modeling and analysis tool Expect consists of a graphical editor, a simulator and a visualization component. It is implemented in Java in order to ensure platform independence, and also to make use of its special language features. One example is the dynamic loading of classes, which makes it possible to include functions in the Petri net model which are formulated in Java syntax, which can then be compiled and executed at run-time. This means that such functions execute with the same level of performance as the tool itself, since expensive parsing routines are no longer necessary. Furthermore, the Java compiler and interpreter are available everywhere free of charge.

The Expect graphical editor is designed with multi-document capability, allowing several nets to be edited simultaneously. This in turn allows sub-nets to be copied between models and for these to be compared quickly. Since Petri nets can quickly become very complex and unwieldy, Expect also allows hierarchical modeling. A net may be divided into sub-nets, which may be edited separately and linked together via transition or place interfaces. This supports both logical model development on the one hand, and clear graphical presentation of large models on the other. Expect contains a large number of configurable parameters; model parameterization is supported by dialog windows for each net component, which include plausibility checks for parameter values. All net components, with the exception of arcs, have unique names.

The visualization component of Expect can be started directly from the editor. This allows the user to study the behavior of the net by watching the token game. This is important for demonstration and debugging purposes. The visualization module contains continuous speed settings as well as a step-by-step mode. The enabling state and enabling times of timed transitions are visualized in order to further enhance understanding of the net's dynamic behavior.

The simulator may also be started directly from the editor. Both simulation up to a specified point in time or up to an absorbing state are permissible. Furthermore, a number of replications can be specified, and the simulator will provide appropriate statistical results. These include the values of the user-specified rewards in addition to the standard measures for places (probability of being non-empty, average number of tokens) and transitions (throughput and probability of being enabled.) In addition to this functionality, it is also possible to perform transient analyses by parameterizing the range and step size of solution time points. The simulator will then compute statistical values for each of these.

CLASS OF NETS SUPPORTED

Expect supports a very general class of stochastic Petri nets. These include many additional features which greatly enhance the usefulness of the tool to the modeler.

Places can be assigned an initial marking and a maximum capacity, which may either have a constant value or be defined as a function of the current state of the net. An otherwise enabled transition is disabled, if, by firing, the maximum place capacity would be exceeded.

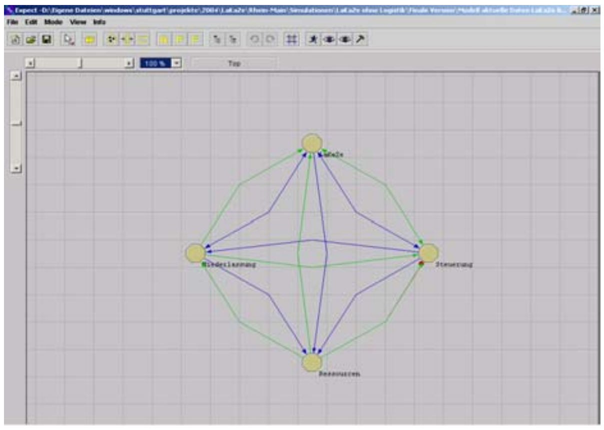
Both timed and immediate transitions are supported. Immediate transitions can be assigned a weight, whereas timed transitions can be assigned a firing time distribution chosen from a large set of alternatives. Firing time distributions range from exponential and phase-type distributions to Weibull and Normal distributions. In addition, the parameters for the firing probabilities and distributions may be defined as marking-dependent functions. Expect also allows the use of marking-dependent guard functions and priorities to control the enabling of transitions.

Transitions may be of single, multi- or infinite server type; multi-server and infinite-server transitions are treated by the simulator by assigning multiple firing times to the transition in accordance with the current enabling degree. Transitions may also be assigned a memory policy of type age or enabling. Memory policies define how the enabling time of a transition is treated when the transition becomes disabled for any reason other than itself firing. In the enabling case, a transition will "forget" that it has been enabled for a certain period; when it once again becomes enabled, a new firing time will be computed. In the case of the age policy, the transition "remembers" its enabling time, which shortens its remaining firing time when it once again becomes enabled. These memory policies are very important for modeling purposes.

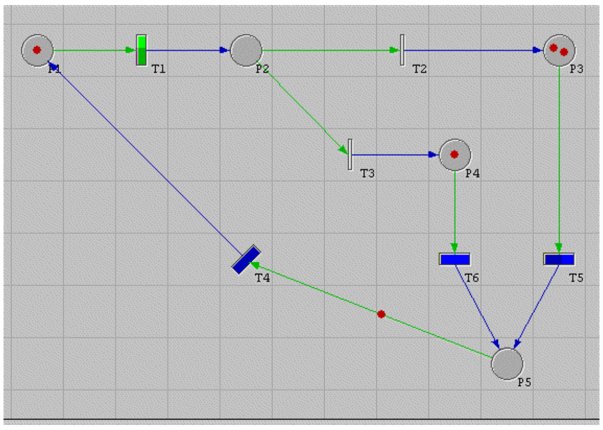
Expect also supports multiple arcs, whose multiplicity may be specified as a constant or with a function.

Various types of rewards are also available, including accumulated, non-accumulated and impulse rewards, which are associated with the firing of a transition. In addition to the net components themselves, independent parameters and functions may be defined, which can be referenced by the functions used to parameterize the net components. These facilitate the fast and safe modification of model parameters.
In the following, three screenshots of Expect demonstrate, how a system can be modeld in a hierarchical manner, how a detailed block looks like and and how analysis results are finally presented to the user.

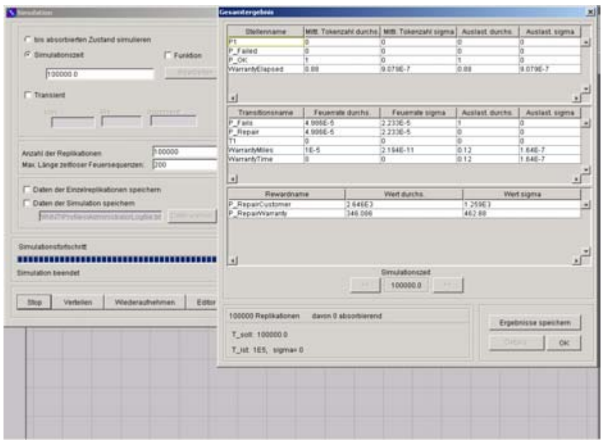
- Hierarchical top layer of a Petri net



- Detailed Petri net model within a hierarchical layer



- Representation of the analysis results



bis absorbierten Zustand simulieren
Simulationszeit: 100000.0
Funktion
Transient
Anzahl der Replikationen: 100000
Max. Länge zeitloser Feuersequenzen: 200
Daten der Einzelreplikationen speichern
Daten der Simulation speichern
Simulationsfortschritt
Simulation beendet
Stop | Verteilen | Wiederaufnehmen | Editor

Stellenname	Mittl. Tokenzahl durchs.	Mittl. Tokenzahl sigma	Auslast. durchs.	Auslast. sigma
P1	0	0	0	0
P_Failed	0	0	0	0
P_OK	1	0	1	0
WarrantyElapsed	0.88	9.079E-7	0.88	9.079E-7

Transitionsname	Feuerrate durchs.	Feuerrate sigma	Auslast. durchs.	Auslast. sigma
P_Fails	4.996E-5	2.233E-5	1	0
P_Repair	4.996E-5	2.233E-5	0	0
T1	0	0	0	0
WarrantyMiles	1E-5	2.194E-11	0.12	1.64E-7
WarrantyTime	0	0	0.12	1.64E-7

Rewardname	Wert durchs.	Wert sigma
P_RepairCustomer	2.646E3	1.259E3
P_RepairWarranty	346.066	462.88

Simulationszeit: 100000.0
100000 Replikationen davon 0 absorbierend
T_soll: 100000.0
T_ist: 1E5, sigma= 0
Ergebnisse speichern | OK

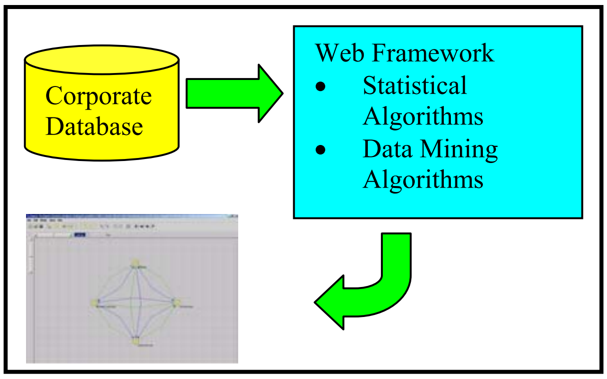
NEXT DEVELOPMENT STEPS

Expect is used very heavily at DaimlerChrysler Research in the form described here. In addition to the current functionality, various extensions are also planned, which will be briefly described.

One important application of Expect is the simulative prediction of the availability and reliability of current and future on-board vehicle systems. In addition, the tool is used to analyze quality measures and predict warranty costs. These applications include features for automatic report generation and integration with a corporate database, which contains up-to-the-minute reliability data. In order to access this database, an interface to Expect has been developed and is currently under test. Another very important topic while providing an interface between the analysis tool Expect and our corporate database is the pre-calculation of model parameters, based on our warranty data. Therefore, right now techniques are included into the tool Expect that allow the user to set up a model. Once this is done, the user can link every transition in the model with a component in our corporate database. As soon as this link is established, the system automatically extracts the field data from the corporate database that are associated with the link, performs a statistical analysis on these data and feeds the corresponding parameters into the model. Every time, the model is started and information in the corporate database has changed (e.g. new field data arrived at the corporate database), then the system automatically initiates a re-calculation of the parameters and starts a new simulation.

In this sense we are able to set up a quality-and cost tool suite, consisting of

- A Web-framework that contains statistical analysis algorithms for pre-processing the field data
- The analysis tool Expect that uses the pre-processed field data to parameterize and analyze the model

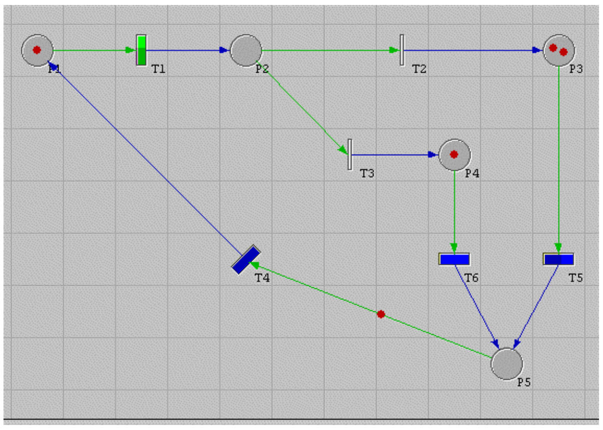


Many questions of interest can be answered by analyzing the state space of a net. These include qualitative properties such as deadlocks and livelocks in addition to the usual quantitative simulation results such as average reward values and marking probabilities. In particular, when all timed transitions in the net are exponentially distributed (or phase-type), then the state space can be converted into a continuous-time Markov chain, for which very efficient numerical transient and steady-state analysis methods are available. For these reasons, state space generation and analysis techniques will also be included in the tool.

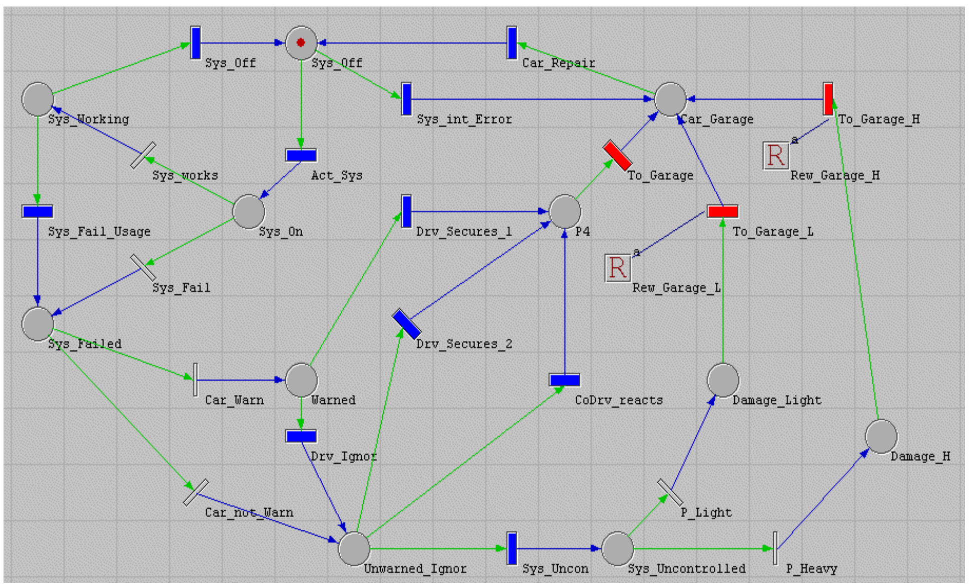
A further issue of interest is to increase the performance of the discrete-event simulation itself. This is motivated by the high accuracy requirements for safety analyses, requiring a large number of replications to achieve statistically significant results, and the large degree of stiffness of many models, which require a long time to reach steady-state. In both cases, very expensive computations can result. Our approach to accelerating simulations by aggregation of the state space [HeHoLu98] will be studied in the Petri net context. In addition, an automatic parallelization of the net has been incorporated into the tool. Hereby, not the Petri net is split into pieces and each beeing analyzed on a different computer but we just parallelize the replications across a network of workstations. This means that we distribute the overall Petri net to different computers. Each computer starts the simulation with a different starting value. In this case we avoid the communication overhead involved in a standard parallelization (distributing pieces of the Petri net to different computers). This type of parallelization proved to be very effective, stable and provided a linear speed-up. Therefore, this type of parallelization has been incorporated into the tool as a standard feature

MODELING EXAMPLES

Example 1: The following picture shows a screenshot taken during the visualization of the simulation of a small example net. The distribution functions of the transitions can be identified by the transition color. The expired enabling time of each transition is shown by the darker portion of the transition. Tokens are drawn in red. When a transition fires, tokens move symbolically along the input arcs from the input places to the transition and along the output arcs from the transition to the output places, according the various arc multiplicities. Viewing the simulation visualization in this manner significantly enhances the user's understanding of the net's dynamic behavior, and is very useful in presentations to non-specialists and of course for verifying (i.e debugging) the model.



Example 2: The next Expect screenshot shows a small Petri net, which is part of a model developed at DaimlerChrysler Research. The goal of this project was to study the reliability of a system over the entire life



cycle of a vehicle, including the interaction with the driver, in particular the manual deactivation of faulty systems. Rewards are used to compute the probabilities of light and serious damage occurring due to inappropriate behavior by the driver. The screenshot shows a Petri net with its initial state, in which the system is off. This is characterized by the token in the place *sys_off*. In this state it is possible to activate the system by the user or the system can fail unused. When the system is not used and it fails, it is detected by the system and it changes the state in defect. This behavior is modeled by the transition *Sys_intError*, which changes the state of the Petri net from *Sys_off* in *Car_Garage*. From this state, the system will be repaired and changes into the initial state of the net. If the system has not failed, the user can activate it and the token in place *Sys_off* walks to place *Sys_on*. By the activation of the system it is possible that the system is defect. In the case of no defect, the state of the system will change to working and the token walks to the place *Sys_working* by firing of the transition *Sys_works*. In this state the system can be deactivated by the user and the system changes in the state deactivated. This way from system deactivated to activated and back is the normal usage of the system, but while using the system, the system can fail and the token in place *Sys_Working* moves to the place *Sys_Failed*. This state of the system is the same state which will be reached if the system is down and will be activated. In the case of a system failure while the system is in use, either the driver can be warned by the vehicle or, in the case of a failure, the warning does not occur. When the user is warned, the token moves to place *Warned*. In this state, the driver can ignore the warning or he will secure the vehicle. Ignoring of the warning will lead to the same state as when the vehicle cannot warn the driver. This state is characterized by a token in place *unwarned_ignor*. At this point, the question arose, whether it is necessary to improve the availability of the warning or not. In the original model it was not obvious if it is or not. By analyzing the model, it was possible to prove that the warning must not be redesigned, because the most probable cause for damage was the ignorance of the driver. If the driver does not ignore the warning, he secures the vehicle (by braking, for example) and the system will be repaired at a garage. This case is modeled by the places *P4* and *Car_Garage*, and the transitions *Drv_Secures_1*, *To_Garage* and *Car_Repair*. In the case where the driver ignores the warning, or there is a warning failure, we have three different possibilities. The first way is the securing by the driver, if he is fast enough to react; the second is securing by the front-seat passenger if he can react fast enough. In both cases, the token will go to place *P4* and the system will be repaired in the garage. The last possibility is the worst one, in which the system has failed and the vehicle is in an uncontrolled state, in which there is no way to return to a secure state. In this situation, damage is unavoidable and the question is how heavy the damage will be. In the model, the damage is divided in two classes, light and heavy. These are modeled in the Petri net with the transitions *p_Light* and *p_Heavy*. In both cases, the vehicle is repaired after the crash and the vehicle with the system returns to the initial state. A very interesting question for DaimlerChrysler was the amount of light and heavy damage. For measuring this, the impulse rewards *Rew_Garage_L* and *Rew_Garage_H* are used. The model is a simplified model, because it abstracts from system functions and different failure modes of the system. Nevertheless, it shows the usage of Petri nets for analyzing systems and the behavior of systems with interactions with humans. With the model shown in this example, it was possible to prove that a failure in the warning system was not the main danger; the main problem was the driver ignoring the warning, which lead to new concepts for increasing the safety of the overall system.

REFERENCES

[ABE94] R.B. Abernethy, "*The New Weibull Handbook"*, 536 Oyster Road, North Palm Beach, Florida, 1994.

[BGdT98] G. Bolch, S. Greiner, H. deMeer, K. Trivedi, *"Queueing Networks and Markov Chains – Modeling and Performance Evaluation with Computer Science Applications"*, Jogn Wiley&Sons, New York, 1998.

[BuFa01] R.L. Burden, J.D. Faires*, "Numerical Analysis"*, 7th Edition, Pacific Grove: Brooks/Coole, 2001.

[HeGrHo92] S. Heller, S. Greiner. G. Horton, *"PeNeTo: A Petri Net Simulator for Fast Safety and Quality Analysis and Cost Prediction"*, Proceedings ESM 2002.

[HeHoLu01] S. Heller, G. Horton, M. Luber, *"Accelerating Discrete-Event Simulation via State Space Reduction",* ESM 2001, Prague, June 2001, Society for Computer Simulation and Modeling.

[Lawl92] J. Lawless, "*Statistical Models and Methods for Lifetime Data",* John Wiley, New York, 1982.

BIOGRAPHY

Dr. Stefan Greiner studied Computer Science at the University of Erlangen. After two years at Duke University (Prof. Trivedi) he received his PhD degree in 2000 from the University of Erlangen. Since 1997 he is working for DaimlerChrysler Research in Stuttgart. His research area is the analysis and prediction of quality and cost. In this context, a tool environment for quality, cost and safety analysis is developed and contantly extended.