Specification, Safety and Reliability Analysis Using Stochastic Petri Net Models

Frederick T. Sheldon
School of EECS, WSU
Pullman, Washington 99164-2752, USA
Sheldon@acm.org

Stefan Greiner and Matthias Benzinger
Perf. Modeling & Process Ctl. Rsrch Grp.
Dept. of CS IMMD IV, U. of Erlangen
Stefan.Greiner@informatik.uni-erlangen.de

Abstract

In this study we focus on the specification and assessment of Stochastic Petri net (SPN) models to evaluate the design of an embedded system for reliability and availability. The system provides dynamic driving regulation (DDR) to improve vehicle derivability (anti-skid, -slip and steering assist). A functional SPN abstraction was developed for each of three subsystems that incorporate mechanics, failure modes/effects and model parameters. The models are solved in terms of the subsystem and overall system reliability and availability.

Four sets of models were developed. The first three sets include subsystem representations for the TC (Traction Control), AB (Antilock Braking) and ESA (Electronic Steering Assistance) systems. The last set combines these systems into one large model. We summarize the general approach and provide sample Petri net graphs and reliability charts that were used to evaluate the design of the DDR in parts and as a whole.

1. Introduction

The sources of errors in complex systems include a wide range of possible failure causalities (e.g., untested manufactured flaws, software design and implementation defects including timing errors, etc.). The most prevalent types are highly dependent on the system, it's operating environment, workload and system design including the integration and testing process. Furthermore, in high-assurance systems, timing and performance issues must be considered. For example, embedded real-time systems (e.g., characterized by intense interaction with sensors and actuators) can control continuous reversible processes that typically possess the ability to tolerate brief periods of incorrect interaction either in values exchanged or the timing of exchanges. To consider such factors during the specification and design of such systems is a difficult undertaking. This work centers on understanding the likelihood of a critical failure (symptoms, causes and affects) in a vehicle system (e.g., automobile). Stochastic models were selected to comprehend the uncertainty (i.e., failure rate data was available). Additionally, there are numerous Stochastic Petri net (SPN) solvers available [8]. See Section 8 for more details regarding other advantages.

1.1 An initial state-based description

To begin, many aspects of safety can be evaluated using non-stochastic models to ensure that the models pass certain sanity checks. Various state machines, entity life history and MASCOT diagrams were constructed including a high-level system schematic (showing sensors, acuators, processing, communication, control pathways and redundancy). A low-level schematic of the Electronic Brake Control Module was also consulted. The final check involved determining which components contribute to specific failure symptoms (e.g., braking pressure loss on the front axle).

All system components were enumerated in a spreadsheet (as row labels) with their failure rates inserted at the row and column location corresponding to the component and the symptom (specified as column labels). There was considerable overlap among the components because of the duality of purpose that prevailed (e.g., a speed sensor is used to detect skidding and slipping). Yet, this technique was an effective way to delineate failure symptom dependencies. Each symptom was assigned a set of failure rates that were determined based on the various non-stochastic models described above. Our first cut at the analysis, as reported here, simply took the sum of all the rates in a set without accounting for functional overlaps (i.e., coincident failure dependencies).

1.2 A functional stochastic description

A bottom-up approach provided a means to determine, at the subsystem level, how a failed component could cause a critical failure that would be visible at the system level. Critical failures include such outcomes as loss-of-vehicle or severely degraded performance levels that cause unsafe operation. Empirical failure data was available for each component of the system. Components were grouped into sets of sensors, acuators and processing according to

Proceedings of the Tenth International Workshop on Software Specification and Design (IWSSD'00)
0-7695-0884-7/00 $10.00 © 2000 IEEE

their association with the various DDR (Dynamic Driving Regulation) sub-functions. Sets were selected on the basis that any component from the group would manifest the same failure effect (i.e., preventing the system from exhibiting the desired behavior). Essentially, the failure of a sensor pack may, for example, prevent the vehicle from detecting an oversteer condition that would normally cause the system to react by providing the appropriate compensation action (e.g., steering assistance). The set of sensors in the sensor pack were summed together and assigned to a single failure transition. Each failure transition was paired (competitively) with an operational transition (a simple conflict in the PN). The operational transition represents the desired system performance. In this way, the performance of the vehicle using an SRN can be assessed from the range of being fully functional to some degraded level. This explains how the models were functionally partitioned within their operational context.

The analysis is summarized here. The SPN model transitions that represent normal and abnormal operation. Each SPN was composed and solved to predict, over time, the ability of the system to complete the desired functions. Four sets of models were developed. The first three sets include subsystem representations of the TC (Traction Control), AB (Antilock Braking) and DSA (Electronic Steering Assistance) systems. The last set combines these systems into one large model. Four types of models were developed for each set including cyclic reliability and availability, as well as acyclic reliability and availability. Cyclic models continuously circulate a token to result in a larger reachability graph and constitute an altered Markov analysis. These model types are explained in Sect. 7 (Fig. 5). The stochastic analysis comprised a set of graphs showing the expected values of these measures (including MTTF) over time. In all, fifteen models were developed (using the C-based Stochastic Petri Net Language).

Section 2 provides a brief philosophy of modeling in which case the principle steps represent important points that impact the safety properties of interest [1,19,20,26]. Sections 3 and 4 explain how SRNs model reliability and availability (i.e., mathematical methods for stochastic modeling) and Section 5 gives the different operational scenarios. Section 6 explains the specification approach.

2. Modeling philosophy and approach

The basic modeling philosophy is shown in Figure 1. It begins by identifying the essential system components, the different ways they interact and introduces various assumptions. In making these assumptions, we consider three basic modeling tenants: (1) simplicity, (2) ease of evaluation, and (3) adequacy of measurements. The key to a successful modeling analysis is the skill needed to introduce assumptions [24,13].

Complexity prevents us from making a direct analysis [2]. A series of abstraction steps are necessary to combine system measures (from the real system) with the system model. Initially, the system model is created at a conceptual level (i.e., abstract model) that specifies the system in a way that offers adequate consideration for the factors that *may* be important (see Figs. 3 and 4). Data collected from *system measurements* are used to parameterize the model(s). However, the complete *system model* will usually contain details that prevent an efficient system analysis. In a second abstraction step the computational model is created to provide an essential structure that can be efficiently analyzed. The *computational* model is considered the highest level of model abstraction that can feasibly be solved. The process of refining the computational model is a matter of building confidence in the model. Thus, the process of *operational validation* (Fig. 1) is performed that results in a *modified model* with altered structure and input parameters/data. This step can be repeated until the computed *performance measures* are realistic (empirically valid) or fulfill the analysis requirements (are comparable to other model variants).

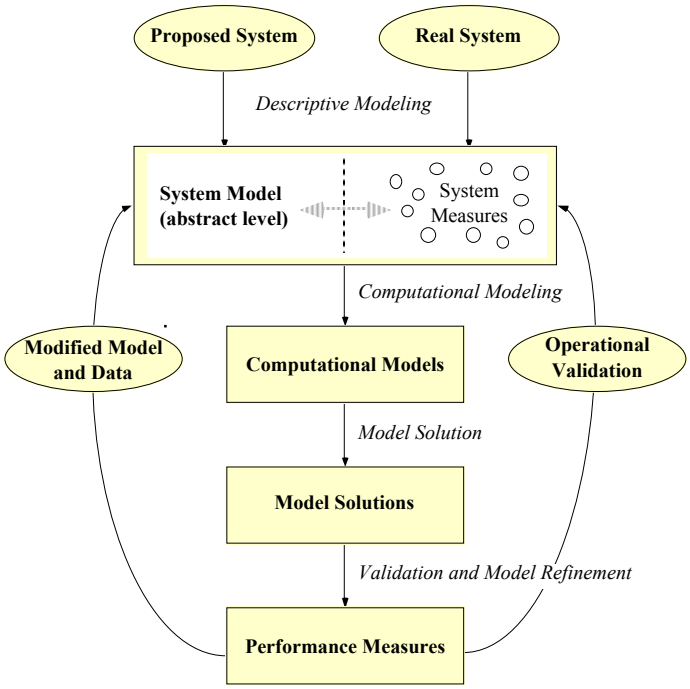


Figure 1. Modeling-cycle emphasizing the principle steps.

The stopping rule for operational validation is now based on relative comparisons between one refinement to any other. Different parameterizations are used to compare the different design possibilities. These comparisons are

formed with the goal of making architectural design decisions (i.e., sensitivity analysis [3-5]) to optimize the model's structure toward achieving critical functional and non-functional requirement harmonization.

3. Estimating reliability and availability

Stochastic Reward Nets (SRNs) are used to generate (large) Markov chains automatically starting from a concise description of the system [21-23,27]. SRNs can represent various properties including concurrency, synchronization, sequencing, as well as a multiplicity of activities and/or resources [15-18]. Some of these properties are relevant to the operational scenarios provided here. For example, the vehicle system exhibits concurrent (e.g., turning and braking) and synchronized behaviors (e.g., steering assist) that rely on the correctness, wear down (i.e., corrode, abrade, contaminate) and timeliness of such activities. For our purposes, the actual failure mechanism was abstracted away. By ignoring the physics of the failure we could focus on two important characteristics: (1) system reliability, and (2) system availability (i.e., reliability with repair of failed components.

The basic approach outlined above (Fig. 1) was used to derive the models. The models were developed from examining the sensors, actuators and processing configuration (i.e., schematics). In this way we could determine how a failed component (or group) would affect overall system operation. A number of failure scenarios result in the loss of vehicle or cause severely degraded performance. Ideally, we would like to identify every sequence of events that cause the unsafe system operation along with their likelihood (or rate) of occurrence. Note that failure rate is typically defined as the ratio of the number of failures in a given unit of measure (e.g., failures per unit of time, failures per number of transactions, failures per number of computer runs). In reliability modeling, the ratio of the number of failures of a given category or severity to a given period of time is, for example, failures per second of execution.

4. Mathematical methods

Combinatorial models such as reliability block diagrams, fault trees and reliability graphs are commonly used for system reliability and availability analysis. These models provide a concise description of the system and can be evaluated efficiently. However, they cannot represent dependencies that realistically occur in genuine life scenarios. Some examples of the dependencies that are not easily captured in such models include imperfect coverage, correlated failures, repair dependencies, performance-reliability dependence, among others [6,9]. These factors constitute the operational context sought in defining the approach used in this work.

The system model is the basis for a successful evaluation of the measures or properties of the system that are to be assessed. Depending on what properties are of interest, we must develop a representation of the real world characteristics that may affect those properties. We must ignore the characteristics that do not have any pertinent consequences or are not relevant based on the magnitude of their contribution to the results. Such factors may not have any appreciable or significant affect in comparison to the main factors that are to be considered.

Traditionally, performance analysis assumes a fault free system. Reliability and availability analysis is carried out separately to study the behavior of the system in the presence of component faults, disregarding the different performance levels in different configurations. Using SRNs, a reward rate is attached to each state of the Markov chain (known as a marking in the Petri net). Time dependent behavior can be studied in this way. We may consider instantaneous and interval availability, reliability (for a fault tolerant system) and computational availability (for a degraded system). Given the reward rate specification for a Markov reward model [14], the expected reward rate is computed in steady-sate, while the expected reward rate at some time t, is an instantaneous measure. The expected accumulated reward rate in the interval [0,t) and the expected accumulated reward until absorption gives the cumulative measures of interest [15]. The mean time to absorption (e.g., MTTF) is a special case of the expected accumulated reward until absorption. In this work, we have computed the reliability at some time t (i.e., R(t)), the instantaneous availability (i.e., A(t)) and the interval availability (A'(t)) as well as the MTTF for each model.

5. Operational scenarios

Three operational scenarios are considered in this analysis: skidding, slipping and steering. The following three sub-sections describe each assuming that they are independent of the other two. However, these systems are not independent. For example if the Antilock Brakes is malfunctioning, the other systems who share various components are normally switched off (i.e., TC and ESA would be disengaged). Therefore, in the final analysis, all three of the scenarios are combined.

5.1 Antilock braking (AB)

The *skidding model* simulates the Antilock Brakes (AB, an integrated part of the total braking system). It avoids locking of the tires while the brakes are applied, maintaining the vehicle's ability to steer. When applying the brakes on an AB-equipped vehicle, wheel sensors monitor the rotational speed of each wheel. The electronic brake control module (EBCM) computer automatically compares the speed of each wheel. If one wheel is slowing

at a faster rate than the others are, the computer sees that the wheel is beginning to lock up. The computer then responds by "pulsing" the brake line pressure to that wheel so it continues to rotate. A vehicle equipped with AB may take somewhat longer to stop during a panic stop situation because AB allows the wheels to keep rotating while slowing the vehicle. On the other hand, a vehicle with locked brakes can stop in a shorter distance (a locked-up wheel generates more friction), but the driver has little or no control. The pulsing of locked wheel(s) gives the driver the ability to steer the vehicle during the deceleration period. This ability to sense wheel speeds and to modulate braking pressure to prevent wheels from locking enhances vehicle stability and active safety accident avoidance. This model considers the EBCM, speed sensors and all braking system components.

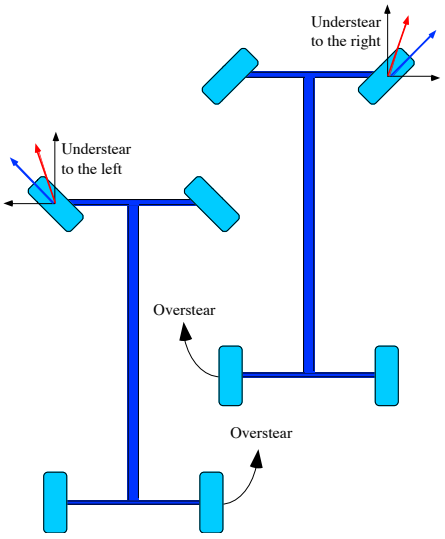


Figure 2. Over/under steer behaviors.

5.2 Traction control (TC)

The *slipping model* accounts for the factors involved during vehicle acceleration known as the traction between each tire and the road. Each tire has an upper limit on the amount of traction it can deliver. If the tire is driven beyond this amount, there is a decrease in adhesion to the road. In this case, the TC reduces power to the tire so that traction is restored to nominal values. The TC maximizes tires adhesion to the road and thus increases driving stability by controlling the vehicle's traction (e.g., in deep snow, sand or gravel). There are three ways to control traction. The first is called the limited-slip differential where engine torque transferred to the wheel with the best traction in any given situation. It is not an electronic system, and generally doesn't perform as well as newer types of traction control but due to its mechanical characteristics and limits, is considered completely fail safe[1]. The second type is known as brake system traction control. It works just like AB in reverse and uses the same sensors and actuators as AB to apply the brakes and keep a wheel from spinning until the wheel regains sufficient traction[2]. Each wheel is individually controlled, making this setup a perfect match for a variety of slippery surfaces. Generally inexpensive and highly effective, this system is designed for low speed slippage (i.e., braking components are used so that higher speed slip control generates excessive friction and heat). The third type is called drive-train traction control (DTC), which retards power delivery to the slipping wheel or wheels at any speed. Using the same sensors as the AB system, DTC employs a processor that will do one of four things: (a) close the throttle, (b) cut the fuel supply; (c) retard spark timing; or (d) shut down cylinders (more advanced DTC systems do all this plus provide feedback by pushing the accelerator against the driver's foot). This system cuts power in all slippery situations (a disabling switch is usually provided).

5.3 Electronic steering assist (ESA)

The *steering model* captures certain characteristics of the ESA system. This system can enhance overall vehicle control (within physical limits) in all three areas of vehicle performance: accelerating, cornering and braking. ESA effectively senses when a driver might lose control of the vehicle and activates individual wheel brakes and reduces engine torque to assist the driver to maintain stability. ESA combines wheel-speed, steering-wheel angle, yaw-rate and lateral acceleration data from sensors with the braking system and a computer processor provided by the EBCM. The processor compares the intention of the driver with directional reality and actively help the driver respond in critical situations (braking or not). The yaw-rate sensor monitors whether the vehicle is turning and together with data from the other sensors, recognizes both the driver's intention and actual vehicle motion. Corrective measures may be activated to help counteract oversteer and understeer. Braking can be applied separately to any of the four wheels (see Fig. 2):

```
If Under-Steer-Left then Brake(LeftRear)
If Under-Steer-Right then Brake(RightRear)
If Over-Steer-Left then Brake(RightFront)
If Over-Steer-Right then Brake(LeftFront)
```

For example, if the driver oversteers and the rear wheels begin to slide outward, ESA's computer counteracts by braking the outer front wheel, creating an opposing and stabilizing yaw force and by reducing engine power, if necessary. If the driver already has applied the brakes while cornering, the ESA will boost brake pressure at the front wheel that is on the outside of the turn and reduce brake pressure on the inner front wheel. If the driver

[1] Modern limited-slip differentials transfer power to the good wheel before slippage occurs, however, if both wheels are on a slippery (e.g., icy) surface, this advantage is not effective.

[2] The traction control engages at vehicle speeds up to approximately 24 mph (40kph) and switches off at 50 mph (80kph). Such systems can usually be completely disabled by a switch available to the driver.

encounters understeer, ESA brakes the rear wheel on the inside and helps bring the car back to the intended course[3].

6. Specification approach

Here we provide an overview and functional description of the system. We reverse engineered its design. Several diagrams are provided including a high-level state transition diagram, a Structured Diagram (based on Jackson's Method similar to an entity life history diagram used in the Structured Analysis and Design Technique), and a MASCOT diagram (Modular Approach to Software Construction, Operation and Test) are provided [28]. These diagrams provide basic comprehension of the nominal system behavior. The PNs, on the other side, provide an expressive compact representation including absorbing (or failure) states

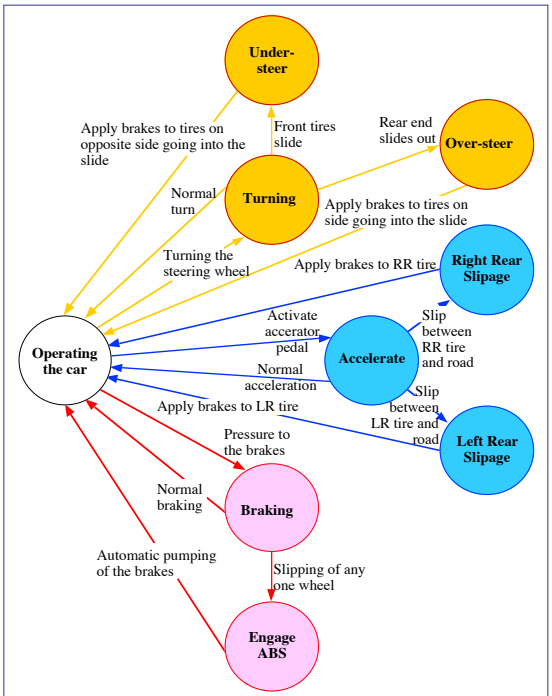


Figure 3. FSM depicting nominal operation.

6.1 Basic overview

Consider the simplified finite state machine (FSM, Fig. 3). The system may enter any one of the states shown. This assumption allows us to analyze each of the subsystems independently. If each is completely independent of the other, then the unreliability of the system as a whole is determined by its weakest link. However, to study system behavior without such an assumption provides the basis for observing and assessing degraded system operation.

During operation, the state of the system changes based on certain stimuli (e.g., pressing on the throttle). When slippage is detected during acceleration, a control activity is activated causing eventual return to the normal operating state. The basic definition of an FSM obligates that the system occupy only one state at any one time. However, in our interpretation, this FSM may occupy simultaneously any of the three different sets of states (turning, accelerate, braking). This is realistic because in most such vehicle systems braking while engaging the accelerator simultaneously is not prevented (this activity actually accelerates wear out).

Accordingly, lets consider the case where 3 tokens are deposited into the state "operating the car." We'll permit all three tokens to circulate with the condition that only one token per branch is allowed. For example, the vehicle could accelerate and turn together. Braking does not prevent acceleration when the torque delivered by the engine out weighs the resistance provided by the brakes. This interpretation provided the basis from which the three subsystem models were derived. In the end we can combine all three subsystems in a manner consistent with the simple Figure 3 synopsis.

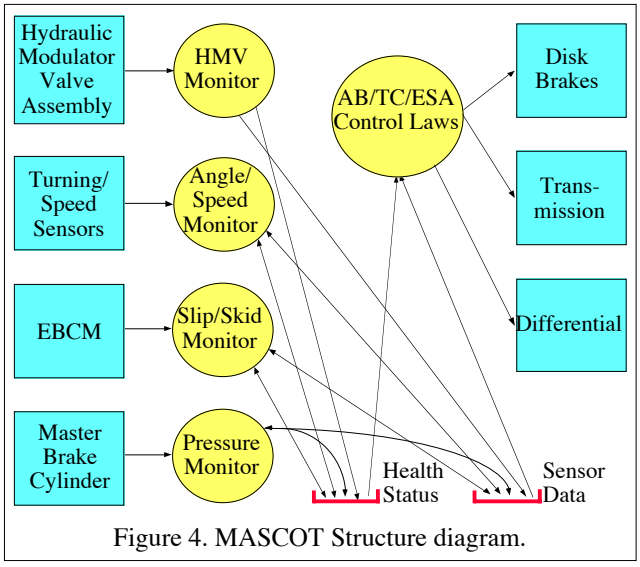


Figure 4. MASCOT Structure diagram.

6.2 Functional description

A high level representation of the physical system is presented in Figure 4 that describes the monitoring, communication and control entities. The bubbles are known as activities that represent a schedulable process. The boxes represent hardware devices, a tray represents a pool of shared data (e.g., sensor data, system status, etc.) maintained during all modes of operation, and the channel ("I" shaped) represents a pipeline of data flow. The various components are shared by each of the different sub-systems (TC, AB and ESA).

6.2.1 Averting slippage with the TC system

The power transmission that occurs during vehicle starting (moving from zero to some nonzero speed) or acceleration, depends on the slip between the tire(s) and the road. Accompanied by increasing slip, adhesion raises

[3] In some vehicle systems the transmission is provided with a winter and summer mode switch that in the winter mode starts the vehicle out in second gear, and with hard acceleration upshifts occur at lower speeds.

up to its limit (see S.574, [29]). A further increase of the slip results in a decrease of the adhesion, at that time the wheel speed increases at one or both drive wheels. In such cases, the TC reduces the traction slip to valid values. The TC first performs an increase of the adhesion and secondly maintains the driving stability.

6.2.2 Averting a skid with the AB system

The Antilock Breaking (AB) is a control system integrated into the brake system (see S.627, [29]). It avoids skidding (i.e., locking the tires) during braking activity. AB maintains steerability and driving stability.

6.2.3 Steering assistance with the ESA system

There are constraints between the yaw-rate, steering-angle and the velocity that define normal driving behavior (see S.670, [29]). If these constraints are exceeded, especially driving in a curve (the vehicle will over/under steer), lateral acceleration is necessary to maintain the constraints within their proper limits. The AB may brake all 4 wheels to stabilize the car, as long as the constraints are out of their valid limits. In certain cases, to maintain the proper limits, the AB will brake certain wheels independently (e.g., oversteer to the right will cause braking of the left front and left rear wheels).

7. Four generic models

This section presents the characteristic Petri net for each of the four model types (Fig. 5). These types include (1) Connected Cyclic Reliability (CCR), (2) Connected Cyclic Availability (CCA) (3) Disconnected Reliability (DR) and (4) Disconnected Availability (DA) models. In each model the components cooperate to perform the designated subsystem function (i.e., TC, AB, and ESA). In general each subsystem function includes sensing and detecting an off-nominal behavior based on a combination of sensor inputs. Consequently, a control function is summoned. In general, failures occur anyplace in this scheme and component groups have combined failure rates (sum of the rates of each component).

The reliability models predict the Mean Time to Absorption (MTTA) of a system, where the MTTA is the average time prior to the system failure. The availability models predict the percentage of time the system will be operational. The recycle transitions are assigned a high rate of firing to simulate an immediate transition. Actually, they may be replaced with an immediate transition, but for simplicity we used a timed transition. This step avoids the problem of creating vanishing markings, since these can lead to vanishing loops. Vanishing loops are problematic in SPNP (Stochastic Petri Net Package), the tool used to solve the models [10-12].

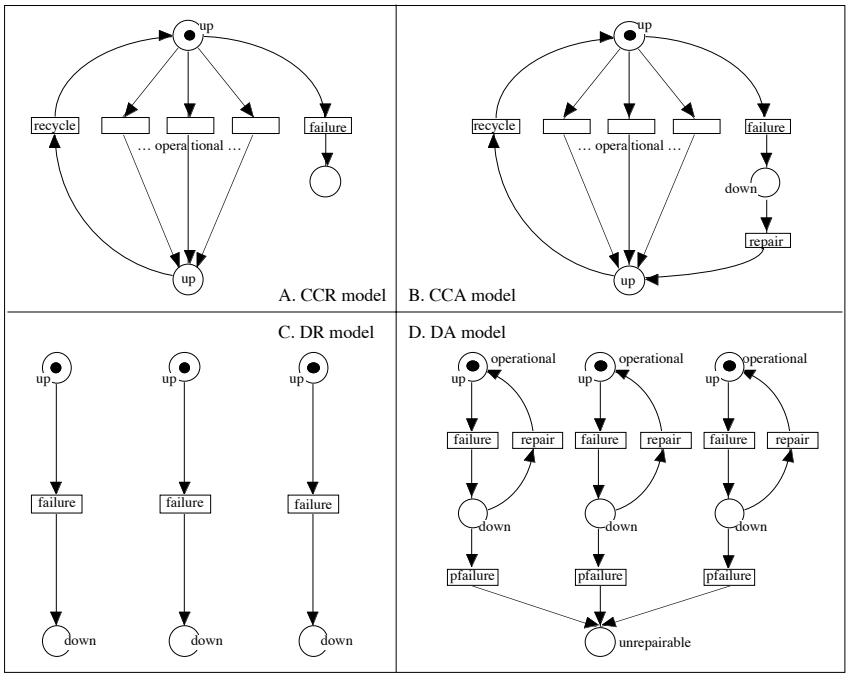


Figure 5. Generic models showing the various approaches used.

7.1 Connected cyclic reliability (CCR) models

In the CCR model (Fig. 5a) the initial marking consists of one token in the *up* place. The *operational* transitions fire alternately indicating that a particular sub-component has functioned. This allows for the model to include different operational dependencies that are present in the actual system. The initial marking creates a simple conflict among the operational transitions and the failure transitions. The firing rates assigned to the various operational and failure transitions represent a *competition* among the operational components and the possibility of a failure occurring. Competing operational transitions were assigned the firing rate of 1/n (where n is the number of operational transitions in conflict). The system remains operational if the initial token recycles among the operational places. If a failure transition fires, it will consume the token and the system will have failed. Eventually the system will enter a failed state depending on the rates assigned to the failure transitions. The time it takes for this to occur is the mean time to failure.

Instances of the CCR actually derived the operational transitions into sensing (or detecting) and actuating (controlling) the sensed behavior.

7.2 Connected cyclic availability (CCA) models

The CCA models (Fig. 5b) are similar to

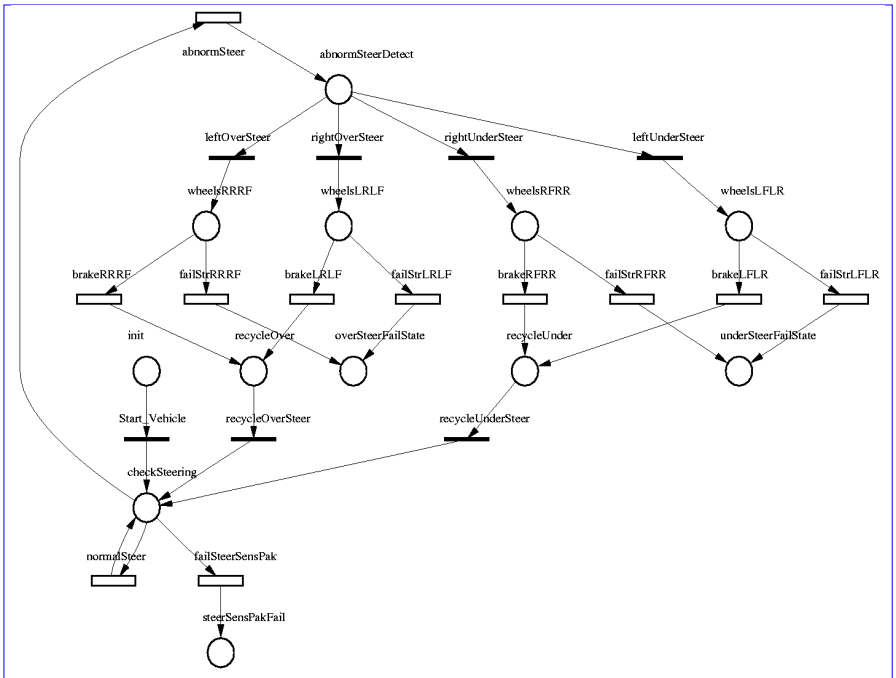


Figure 6. ESA sub-model PN (cyclic version used for reliability analysis).

the CCR models except they introduce a transition to represent the process of repairing the failed system. In both scenarios a failure at either the sensor or the actuator/control processing side may occur and be repaired (i.e., after the failure transition fires the token enables the repair transition). The system is completely unavailable for some period of time until the repair is complete (i.e., represented by the firing of the repair transition). In both the CCR and CCA models a *recycle* transition is used to restart the competition among the operational and failure transitions.

7.3 Disconnected reliability (DR) models

The generic DR model is shown in Figure 5c. The initial marking includes a token in each *up* place, denoting that each component operates independently of the others. Accordingly, each component is assumed to fail independently from any of the others. The firing rate of each of the operational transitions represents the possibility of a component failing.

In contrast to the generic CCR and CCA models there is a conflict between each of the operational transitions and its individual failure transition. Instead of grouping the failure rates for a collection of components into one failure transition they have been separated out to represent the case of degraded operation. The system can therefore continue to operate while one or more, but not all, of the components have failed. The system fails when every token has been absorbed by a *down* place (i.e., every token has been absorbed).

7.4 Disconnected availability (DA) model

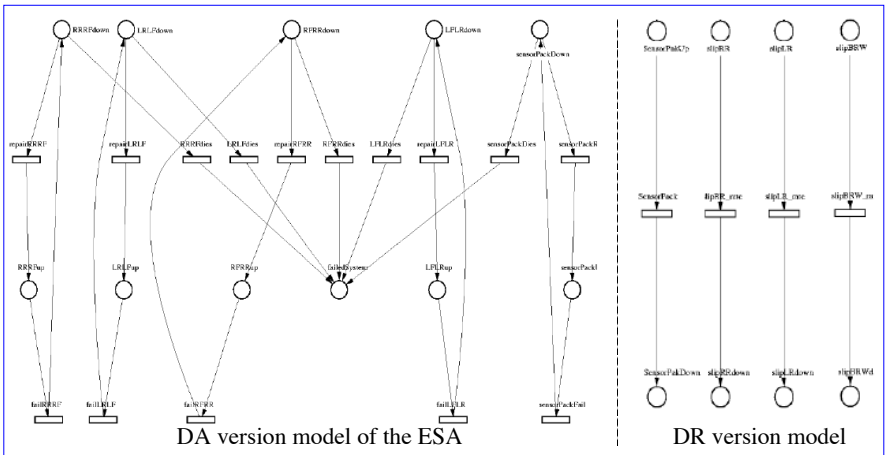
The generic DA model (Fig. 5d) was constructed for the purpose of determining system availability. The initial marking includes a token in each of the *up* places (similar to the DR model). However, there is in the DA model a competition in the DA model between the repair transition and an absorbing place that represents an unrepairable or obsolete state. System performance degrades during the periods while components are failed. During that time there is higher stress on the other operable components (this 2nd point is not actually accounted for in the DA models). All of the tokens are eventually absorbed into the *unrepairable* place at which time the function will have completely stopped.

7.5 Generic models summarized

The functionality of the CCR (Fig. 6) and CCA models may be summarized as follows. The initial marking consists of exactly one token in the *up* place. This token enables any one of the operational transitions that compete for the token that also enables one or more failure transitions. Thus, there is a competition between the operational transitions and the possibility of a system failure. Whenever the token is in the *fail* place, the system is down. In contrast, for the DR and DA models (Fig. 7), there is a token in each of the *up* places initially. The system continues to operate until all of its components have failed. The system (as a whole) fails only when every token has been absorbed.

8. Summary and plans

This paper describes the specification and reliability analysis using Stochastic Petri nets. The SPNs were composed to model the individual components that cooperate and interoperate to perform the designated functions of the vehicular DDR. We were interested in understanding the likelihood of critical failures. Thus, we determined at the subsystem level how a failed component could cause a critical failure that would be visible at the system level. Critical failures included such outcomes as loss of vehicle or severely degraded performance that may cause unsafe operation. Given the operational data that was available for each component, we determined how likely such components, within their operational context, are to cause something unsafe to occur. Some example results from the stochastic analysis are provided in the appendix.



DA version model of the ESA DR version model

Figure 7. ESA sub-models used for reliability/availability analysis respectively.

Safety is one of the main issues that need to be addressed by modern systems specification and analysis. There are often missing or inaccurate data and unidentified hazards. Incorporating safety into a model makes it more valuable, provides additional insights into its properties and may suggest new types of solutions that do not appear in models that do not account for safety. The goal of future work is three-fold. First, develop practical modeling techniques that will facilitate extension of existing stochastic models into models explicitly incorporating safety. Second, develop efficient techniques for such models that would not only allow for solving the stochastic aspect but also provide an additional insight into the safety aspects. This second point may be more naturally accomplished using logical analysis by formulating reasonable safety requirements. Finally, the main theoretical idea from which these developments will be built is the decomposition of the stochastic problem into a finite number of manageable scenario subproblems and the coordination of their solutions by specially designed algorithms [10,25].

Such algorithms, currently available in many tools, have come quite a long ways in the last years. Still, there is a great deal of disconnectedness among the steps needed to (1) understand the problem, (2) break it into manageable subproblems, (3) develop models that are realistic in terms of the sub-problems they represent and combining them into the larger more complex context.

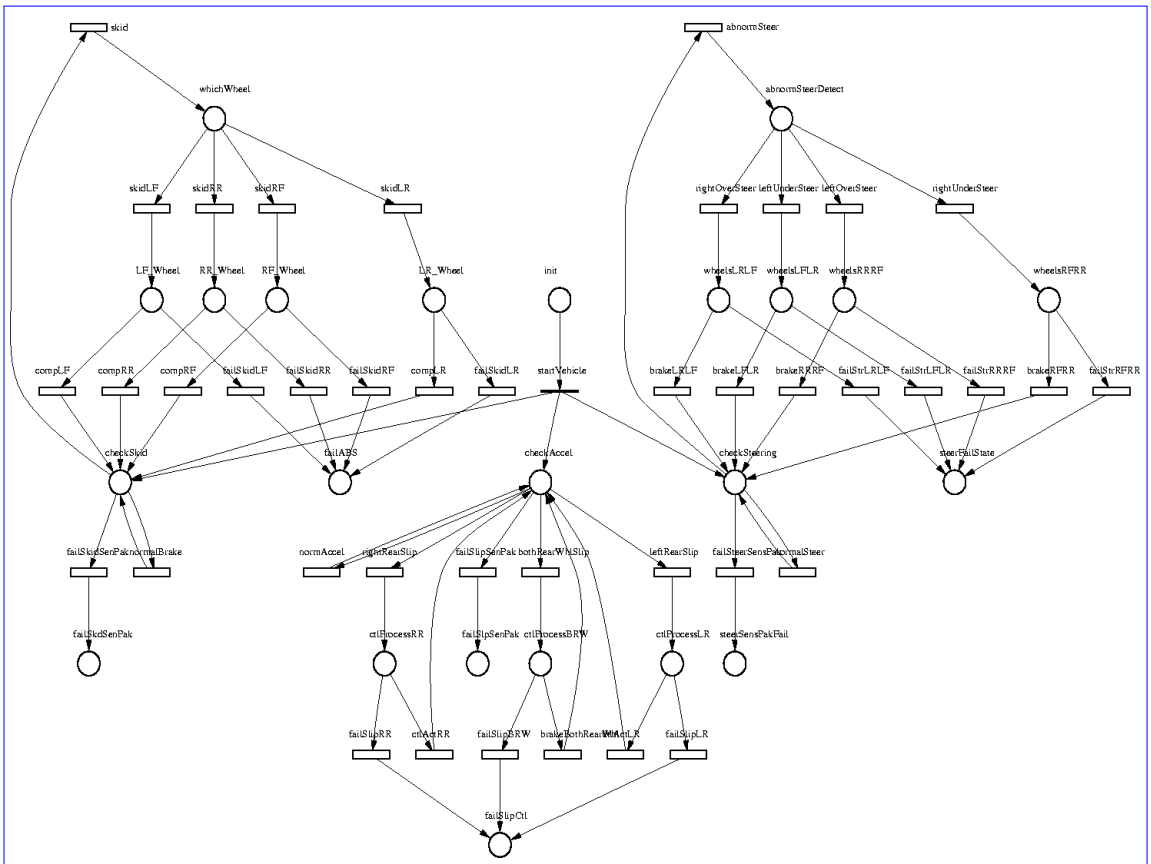


Figure 8. Combined model Petri net (cyclic version used for reliability analysis).

Acknowledgements: The authors recognize the contribution of Mr. David Dugan who dutifully wrote the numerous programs for the experiments conducted using the CSPL (i.e., SPNP) and Mr. Zhihe Zhou for his critical comments (both WSU students). We would also like to thank the IWSSD 2000 reviewers for their comments and questions.

9. References

1. I. L. Yen, Paul, Raymond, and Mori, Kinji, "Toward Integrated Methods for High-Assurance Systems," *IEEE Computer*, vol. 31, pp. 32-34 (incl. 35-46 by others), 1998.
2. R. Marie, Jean-Marie, Alain, "Quantitative Evaluation of Discrete-Event Systems: Models, Performance and Techniques," presented at Fifth Int'l Wkshp on PNPM, Toulouse, France, 1993.
3. H. Choi, Mainkar, Varsha and Trivedi, Kishor S., "Sensitivity Analysis of Deterministic and Stochastic Petri Nets," presented at MASCOTS, 1993.
4. V. Mainkar, Choi, Hoon and Trivedi, Kishor, "Sensitivity Analysis of Markov Regenerative Stochastic Petri Nets," presented at 5th Int'l Wkshp on PNPM, Toulouse, France, 1993.
5. J. T. Blake, Reibman, Andrew L. and Trivedi, Kishor S., "Sensitivity Analysis of Reliability and Performability Measures for Multiprocessor Systems," SIGMETRICS, Santa Fe, NM, 1988.
6. M. Malhotra, and Trivedi, K.S., "A Methodology for Formal Expression of Hierarchy in Model Solution," Fifth Int'l Wkshp on PNPM, Toulouse, 1993.
7. W. Sanders, Obal, W., Qureshi, A. and Widjanarko, F., "The UltraSAN Modeling Environment," *Performance Evaluation*, vol. 24, pp. 89-115, 1995.
8. R. T. Sahner, Trivedi, K. S., and Puliafito, A., *Performance and Reliability Analysis of Computer Systems - An Example-Based Approach Using the SHARPE Software Package*. Boston, Kluwer, 1996.
9. K. S. Trivedi, Malhotra, M., "Reliability and Performability Techniques and Tools: A Survey," presented at Messung, Modellierung und Bewertung von Rechen-und Kommunikationssystemen, Aachen, Springer-Verlag, Berlin pp. 27-48, 1993.
10. G. Ciardo, Muppala J. and Trivedi, K.S., "SPNP: Stochastic Petri Net Package," presented at 3rd Int'l Wkshp on PNPM, Kyoto, Japan, 1989.
11. M. K. Molloy, "Performance Analysis Using Stochastic Petri Nets," *IEEE Trans. on Computers*, pp. 913-917, 1982.
12. A. Reibman, Veeraraghavan, M., "Reliability Modeling: A Modeling Overview for System designers," *Computer*, vol. 24, pp. 49-57, 1991.
13. S. Greiner, "Stochastic Analysis of Computer Science Applications: Theory, Models and Solution Methods," PhD Dissertation in *Computer Science, IMMD IV*. Germany: U. of Erlangen, 2000, pp. 250.
14. J. K. Muppala, Wang, Wei and Trivedi, Kishor, S., "Dependability Evaluation Through Measurements and Models," Duke Univ., EE Dept., Durham, NC Fnl. Rpt. NSF Grant CCR-9108114, 1994.
15. G. Ciardo, Marie, R., Bruell, S., and Trivedi, K., "Performability Analysis Using Semi-Markov Reward Processes," *IEEE Trans on Computers*, vol. 39, pp. 121-1264, 1992.
16. K. M. Kavi, and Sheldon, F.T., "Specification of Stochastic Properties with CSP," IEEE Int'l Conf. on Parallel and Distributed Systems, Taiwan, 1994.
17. G. Balbo, "On the Success of Stochastic Petri Nets," presented at PNPM, Durham, NC, 1995.
18. J. C. Laprie, Kaaniche, M. and Kanoun, K., "Modeling Computer Systems Evolutions: Non-Stationary Processes and Stochastic Petri Nets - Application to Dependability Growth," PNPM, Durham, NC, 1995.
19. N. Levenson, et. al., "Safety Analysis Using Petri Nets," *IEEE Trans. S/E*, vol.13, pp. 386-397, 1987.
20. A. D. Lewis, "Petri Net Modeling and Software Safety Analysis: Methodology for an Embedded Military Application," in *Computer Science*. Monterey, CA: Naval PG Sch., 1988, pp. 98.
21. S. Donatelli, Ribaudo, M. and Hillston, J., "A Comparison of Performance Evaluation Process Algebra and Generalized Stochastic Petri Nets," presented at 5th Int'l Wkshp PNPM, Durham, 1995.
22. G. Balbo, Donatelli, Susanna and Franceschinis, Giuliana, "Understanding Parallel Program Behavior through Petri Net Models," *Journal of Parallel and Distributed Computing*, vol. 15, pp. 171-187, 1992.
23. G. Balbo, Donatelli, S., Granceschinis, G., Mazzeo, A., Mazzocca, N. and Ribaudo, M., "On the Computation of Performance Characteristics of Concurrent Programs Using GSPNs," *Performance Evaluation*, 19, pp. 195-222, 1994.
24. F. T. Sheldon, and Greiner, S., "Composing, Analyzing and Validating Software Models to Assess the Performability of Competing Design Candidates," Annals of Software Engineering 8, pp. 239-287, 1999.
25. F.T. Sheldon, "Analysis of Real-Time Concurrent System Models Based on CSP Using Stochastic Petri Nets," presented at 12th ESM, Manchester, UK, 1998.
26. F. T. Sheldon, and Kavi, K.M., "Position Stmt: Linking Software Failure Behavior to Specification Characteristics II," Int'l Wkshp on Eval. Techniques for Dependable Systems, San Antonio, 1995.
27. G. Ciardo, Trivedi K.S., "A decomposition approach for stochastic reward net models," *Performance Evaluation*, vol. 18, pp. 37-59, 1993.
28. David Budgen, Software Design, Addison-Wes, 1994.
29. R. Bosch, Automotive Handbook, Bentley Pubs 1997.

10. Appendix A: Example models and results

The DDR combines the TC, AB and ESA subsystems into one large system to improve vehicle driveability. The DDR was reverse engineered (RE) to the extent of producing the diagrams shown in Figures 3 and 4. These two figures are provided as representative of the numerous artifacts that were produced as part of the RE process. (Lack of space has forced us to be selective.) A high-level system schematic showing sensors, acuators, processing, communication, control pathways and redundancy was produced. In addition, all system components were listed in a spreadsheet (as row labels) with their failure rates inserted at the row and column location corresponding to a particular fault/symptom (listed as column labels). If a faulty component could lead to a given symptom then its contribution to that type of failure was identified and recorded. This technique delineated failure/symptom dependencies. The relationship of a particular component to a given symptom was more easily identified from constructing the FSM/MASCOT diagrams (Figs. 3 and 4).

We investigated four different generic model types for each sub-function (TC, AB and ESA) and several combined models. Figures 6 and 7 gives an example of the ESA sub-system SPN while Figure 8 provides the composite system. These two diagrams exemplify the general relation between the sub-models and their larger

"composite" context. The ESA structure (Fig. 6) can be seen in the upper right corner of the composite model (Fig. 8, connected by the "checkSteering" place). These diagrams were automatically generated from the SPN models written in CSPL (C-based Stochastic Petri Net Language). The graphical Petri net representation (e.g., Figs. 6–8) significantly aid in checking, through inspection, the correctness of each model. The graphical form makes possible both structural and hazard analysis at the sub-system level.

The composite model state space was too large to solve given the limitation of our computer. The stochastic sub-models were solved (Figs. 9–10). Reliability decreases precipidously. R(80,000hrs) is zero for the acyclic model. The cyclic CCR model is composed of contextual transtions representing various operational activities. These activities compete with the failure processes. Thus, the CCR degrades at a much slower rate (i.e., R(160,000 hrs) is 0.07). The repair process (Fig. 10) has a significant effect. Availability reaches a steady state in the range of 0.5-0.7. The increase in availability after 3 million hours is likely due to numerical instability (error accumulation).

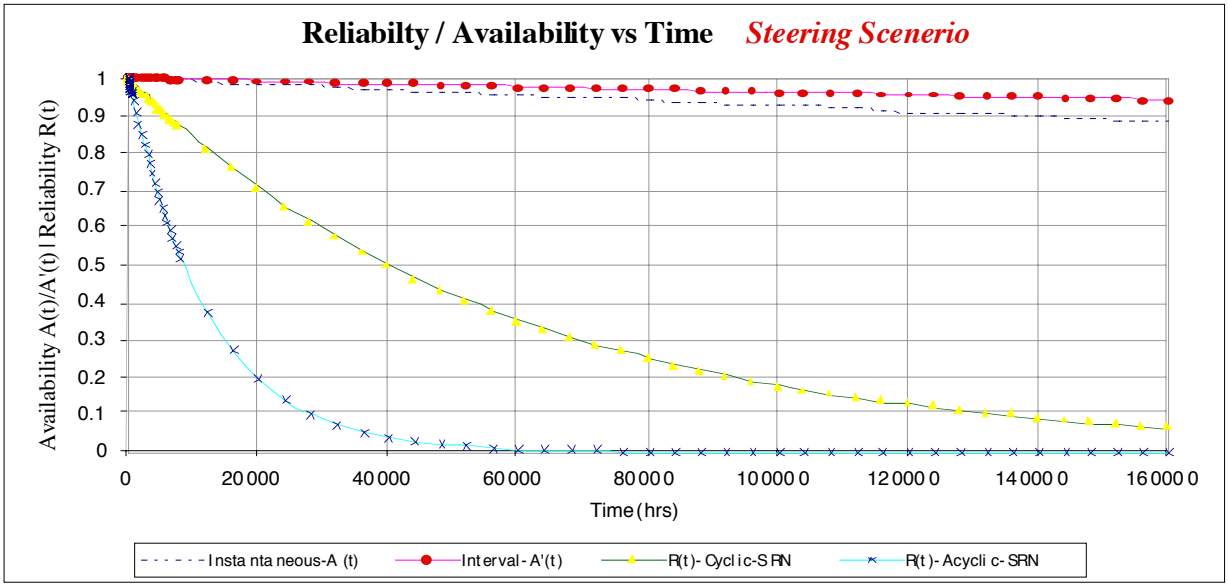


Figure 9. ESA results for reliability (R –CCR and DR versions) and availability (A –DA version) vs time.

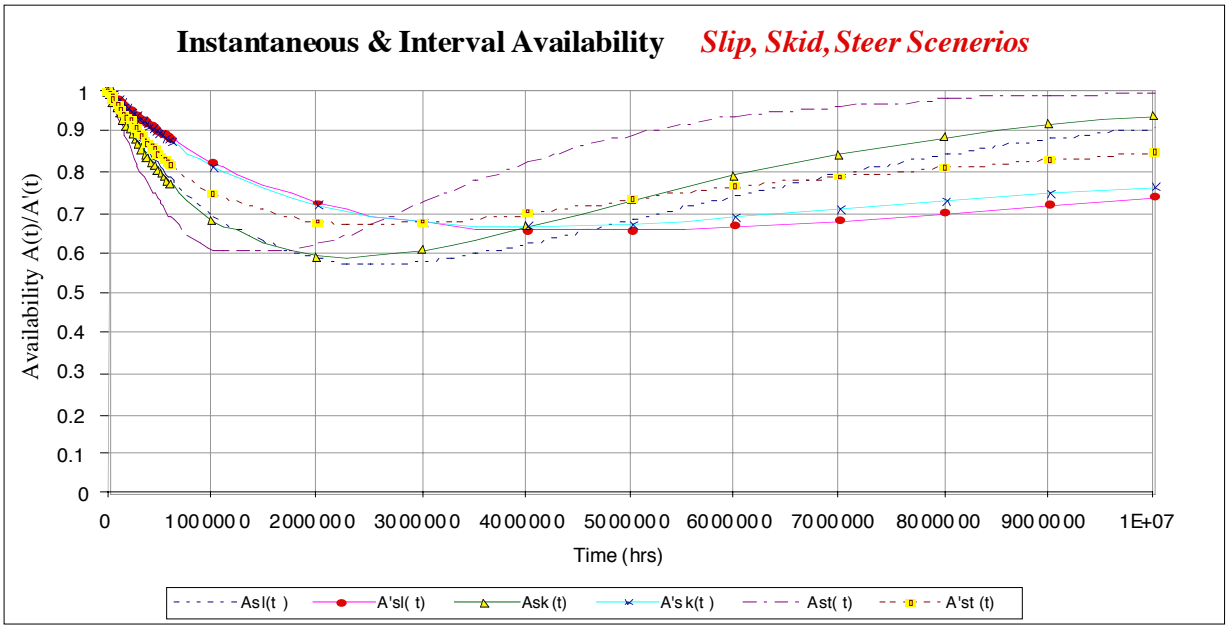


Figure 10. Availability based on extra long runs (0 – 10 million hours) for all three sub-models (DA version).