Reprint: 12th European Simulation Multiconference, June 16-19, 1998 (Manchester, United Kingdon).

ANALYSIS OF REAL-TIME CONCURRENT SYSTEMS MODELS BASED ON CSP USING STOCHASTIC PETRI NETS

Frederick T. Sheldon
Department of Computer Science
The University of Colorado at Colorado Springs
1440 Austin Bluffs Parkway
Colorado Springs, Colorado 80918, USA
E-mail: Sheldon@cs.uccs.edu

KEYWORDS

Software engineering, Computer systems, Model analysis, Real-time, Performance and reliability analysis.

ABSTRACT

This paper addresses the real-time and reliability analysis of models for concurrent systems. Such models define independent entities that cooperate by explicit communication. Communications represent visible actions which, if they do not occur or are delayed beyond their deadline, will cause a failure to occur. This approach converts a formal functional system description into the information needed to predict its behavior as a function of observable parameters (i.e., topology, fault-tolerance, deadlines, communications and failure categories).

The CSP-based models are translated into Stochastic Petri nets (SPNs) using our tool CSPN (CSP-to-Stochastic Petri Nets).[1] CSPN uses algorithms which codify the canonical translations between *essential* CSP constructs and SPNs. The term "CSP-based" is used to distinguish between the exact notation of Hoare's original CSP and our textual representations which are similar to occam2. The CSP-based grammar is sufficient to preserve the structural properties of the original model. Consideration of other CSP properties (e.g., traces, refusal sets, livelock, etc.) are not precluded, however they are not considered here. A basic example is provided to illustrate the link between failure behavior and model characteristics (i.e., derivation of timing failure probability and reliability predictions as they would affect the cost for implementing a candidate model specification).

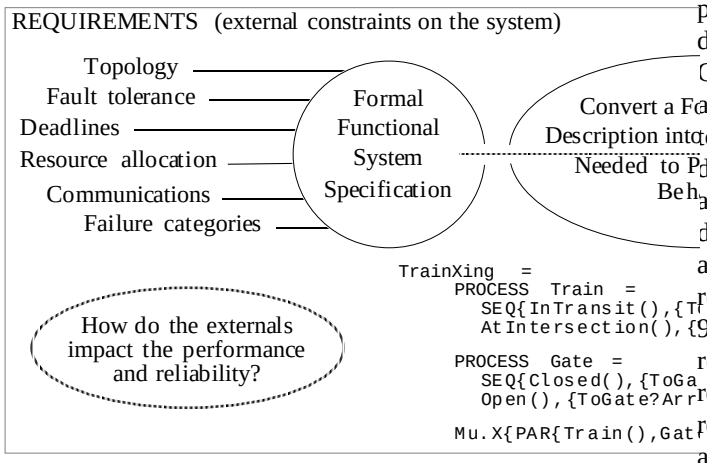


Figure 1. Predicting reliability of model specifications.

1. INTRODUCTION

The source 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, its operating environment, workload and system design including the integration and testing process. Furthermore, in critical 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 (Shin and Kim 94). To consider such factors during the specification, analysis and design such systems is a difficult undertaking. Moreover, the systems designed and built today have greater functionality and higher performance (e.g., confidence gained from many operational hours, legacy systems that are evolved and have been refined). Whether these systems are *more robust* and *more reliable* is a less obvious question. In assuming they are more reliable, the question then becomes "...at what price?" The challenge is to develop effective methods and realistic models for reasoning about and evaluating such systems prior to building costly prototypes.

As may be visualized in Figure 1, formal mathematically precise methods should be used to design such systems [Ostroff 92]. Given a formal model of a system and its external constraints (e.g., topology, communication, deadlines), what mechanisms are available for avoiding errors and how do they impact the behavioral aspects (i.e., performance and reliability) of the system [Kavi et al. 95]? As models are refined, the reliability and performance requirements can also be refined to reveal the trade-offs in design alternatives such as deciding what are the critical system elements, what features of the system should be changed to improve the system's reliability, or validating performance and reliability goals using stochastic models.

A system is modeled using our CSP-based language. This grammar does not restrict the consideration of correctness properties, however only that the structural and functional properties be preserved. Once the model has been translated, it may be solved using Markov reward analysis [Ciardo et al. 91, 89, Chiola 89, Marsan et al. 84, 89, Tomek et al. 94, Bolch et al. 98].

This paper introduces the formalisms and gives a simple example to illustrate the approach: (1) conversion of the CSP model into SPNs, (2) identifying system failure modes by

[1] CSP stands for Communicating Sequential Processes [Hoare 85] and the language is called PCSP for *Parsable*-CSP.

inspecting the PN structure, (3) identifying how failure modes can be handled, (4) specifying the appropriate fault handling mechanism back at the CSP level and (5) cost prediction.

2. CSPN FORMALISMS

The CSPN tool codifies the CSP to PN translations which permit the evaluation of stochastic properties of a CSP system model. The translations are based on the operational PN semantics for CSP defined in by E-R Olderog [87] and therefore permit us to explore *how to verify that both functional and non-functional requirements have been satisfied* using proven translations between CSP and PNs.[2] CSPN satisfies Olderog's theory to the extent that it can be put into a practical experimental setting and provides the versatility of others similar to occam [Jones and Goldsmith 88].

2.1 Communicating Sequential Processes (CSP)

CSP is simple, concise and the process composition rules are very generic (e.g., $\|$, $\rightarrow$, $\sqcap$, $\square$, $\|_b$, P; Q, etc.). CSP as a formal specification method provides rigor and an excellent theoretical foundation for discussing mathematical abstraction of process interactions (i.e., communication, concurrency, recursion, etc.); rules to help in the implementation of processes (i.e., laws used to prove a specification is satisfied); how processes can be composed together into systems where components interact internally and with their environment; definition of a mathematical theory for deterministic and nondeterministic processes; algebraic laws which describe the essential properties of operations that are useful in expressing new problems, solutions and proofs.[3]

The CSP model was developed by Hoare and later extended by Olderog [Hoare 85, Olderog 86, 87]. The basic idea is that systems can readily be decomposed into subsystems which operate concurrently and interact (among and with their common environment). Parallel composition of such systems is as simple as sequential composition using traditional languages (e.g., Pascal). Major benefits from using CSP include avoiding many of the problems associated with parallel programming (e.g., shared resources and multi-threading), a secure mathematical foundation for the avoidance of errors, and for achievement of provable correctness. In addition, CSP provides a complete mathematical definition of the concept of nondeterministic processes which enables modeling of stochastic processes as Petri nets.

Using this approach, systems are built from processes. The simplest process is an action (an assignment, input or output). Larger processes are built by combining smaller processes in a construction. PAR (or $\|$), SEQ (or ;), NDC (or $\sqcap$), DC (or $\square$), and Mu.X{} (or $\mu X\bullet$ P) are constructor primitives. The CSP grammar used by the canonical translation rules is provided in (see Figures 2a and 2b and Sheldon 96). An example construction is: PROCESS my_example = SEQ{P,Q,R}; where each process is performed in succession. A process need not be declared. Declared processes are used like a function call except their internal (pre-defined) structure is hidden. Larger processes are formed from pre-defined processes. A statement list is a sequential list of n 1 statement(s). A statement can be an event (or trigger) which causes a process to engage in an action (e.g., a $\rightarrow$ P an implication). Input and output require a channel which provides unbuffered, unidirectional point-to-point communication among concurrent processes. A guarded process combines one or more processes, each of which is conditional on an input, a Boolean expression or both. An expression can be an integer, Boolean or relational (Boolean expressions must consist of Boolean variables prefixed with "@"). Operands can be integers, variables, integer expressions or relational expressions (distinct from Boolean).

2.2 Stochastic Petri Nets (SPNs)

The PN in its simplest form is a directed bipartite graph, where the two types of nodes are known as *places* (circles) and *transitions* (bars). Places normally represent *events* while transitions represent *actions*. In modeling (see Murata 89, page 542), using the concept of conditions and events, places represent conditions, and transitions represent events. A transition has a certain number of input places and output places representing the preconditions and post-conditions of an event. Applying the notion in CSP of event-action pairings, we employ a slightly different abstraction where the conditions are the events that cause actions (transitions) to take place. For example, a coin inserted in a vending machine causes a candy to be dispensed, the event is the *coin insertion* (token on an input-place) while *dispensing* (an action) is a one-input-place transition firing as a result of the coin insertion[token on an output-place]).

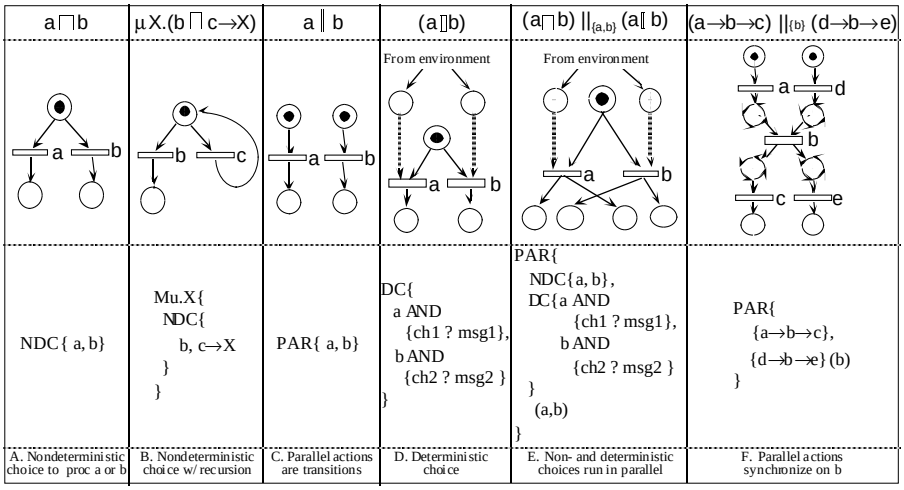


Figure 2a. Example CSP $\rightarrow$ Petri net translation rules.

---

[2] One *could* specify using CSP that certain system components would fail (stochastically) and what actions should be taken. Such specifications are more natural with PNs. Likewise, one *could* possibly use PNs to specify logic and correctness properties.

[3] Process calculi like CSP, ACP, CCS etc. are inherently asynchronous in the purest sense because they have no notion of time. However, timing analysis has been considered by others [Lee et al. 94].

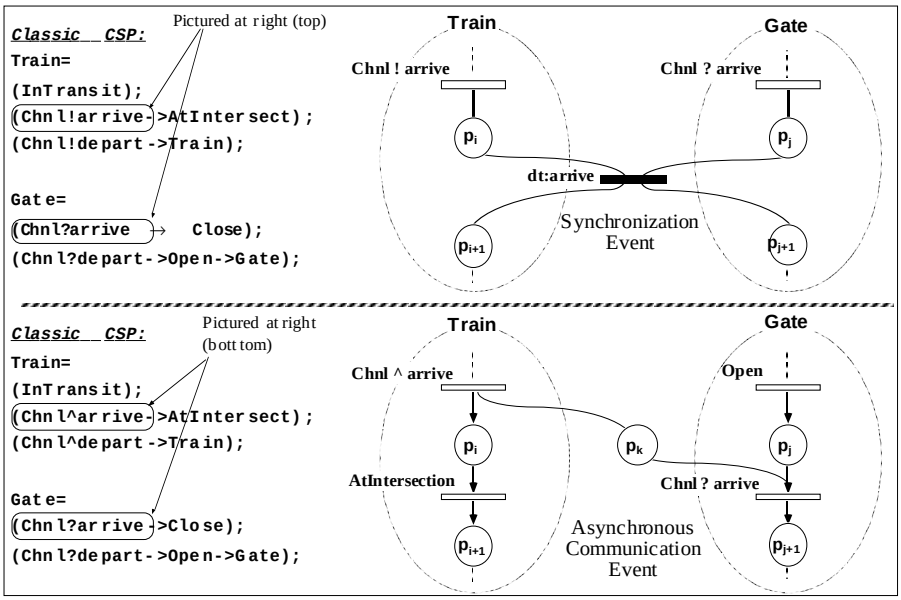


Figure 2b. Communication between CSP process entities.

2.3 Mapping CSP to Petri Nets

The canonical rules for translating CSP models into Petri nets (PNs) are based on the way that CSP model processes move from one action to another. The activities that enable the actions of processes are viewed as events (represented by places), while the actions are viewed as transitions (completing an action is considered an event). Such rules give an associated PN structure for all of the major CSP process primitives and compositions (the bottom half of the rule in Figure 2a gives the textual version of CSP used by the CSPN tool).

The CSP to Petri net translations were designed to facilitate the automatic decomposition of the CSP constructs into Petri net sub-components and subsequent re-composition of the subnet components into a complete *system* Petri net. Once the complete *system* net is obtained, the structure itself is reduced (using equivalent Petri net transformations) to something that is trace equivalent to the CSP model. This is the essential mechanism at the Petri net level for minimizing state space which can be further reduced from an extended reachability graph (including vanishing markings) to one that has the vanishing states eliminated [Allmaier et al. 97, Sorensen et al. 93, Bolch et al. 98].

Figure 2b shows two different types of (codified) synchronizations. In the top half, both the Train and the Gate process are blocked until the "dt:arrive" transition fires while in the bottom half only the gate process may be blocked. The asynchronous translation is used in the following example because it more easily illustrates the race hazard between the deadlines imposed on the train and the gate processes. Either translation may be adopted for consideration during analysis.

3. STOCHASTIC ANALYSIS

How can we model both functionality <u>and</u> reliability (including performance of execution deadlines)? A simple translation is provided to address this question. The Petri net structure facilitates the discovery of a design flaws and problems that may lead to critical and non-critical failures including unwarranted timing dependencies. A timing dependency makes necessary additional synchronization to avoid a safety-critical failure.

3.1 The CSP for a Train Crossing

The gate closes for arriving trains until the train has completely passed by. The problem statement can be extended to handle multiple trains (only one train is specified here):

TRAIN =
(IN_TRANSIT);
(GATE ! a → AT_INTERSECTION);
(GATE ! d → TRAIN)

GATE =
(TRAIN ? a → CLOSE);
(TRAIN ? d → OPEN→ GATE)

RAIL_ROAD_CROSSING = TRAIN $||_{\{a,d\}}$ GATE

Two concurrent processes communicate by sending and receiving messages. The TRAIN sends "a" to inform the GATE of its arrival. After it has passed through the intersection it sends a "d" (departed) to the GATE. The GATE receives the "a" and closes the gate. Once closed, the GATE waits for the "d" before closing the gate. A hazard exists because the TRAIN process could transition to AT_INTERSECTION before the gate closes. Likewise, the train may depart while the gate remains closed. The PN translation in Figure 3 reveals these flaws more readily.

3.2 The Petri Net for the Train Crossing

The train and gate operate independently and communicate to accomplish their mission (i.e., permitting the train to pass safely through the intersection). In Figure 3, the messages exchanged are represented by places P5 and P6. The gate will not close until it receives the “a” message. This process involves markings M3 (send approaching msg), M4 (msg received but gate open), M5 (gate begins to close), and M6 (gate closed). In all four markings (M3-6) the train is "approaching" which constitutes safe operation (see Table 1). Markings M_{cf} and M_{tf} show that the train is at the intersection but the gate is not closed. In M_{cf} either a mechanical or a communication "hard" failure has occurred. In both cases the result is unsafe (i.e., critical [cf]). While in M_{tf} a timing failure has occurred.

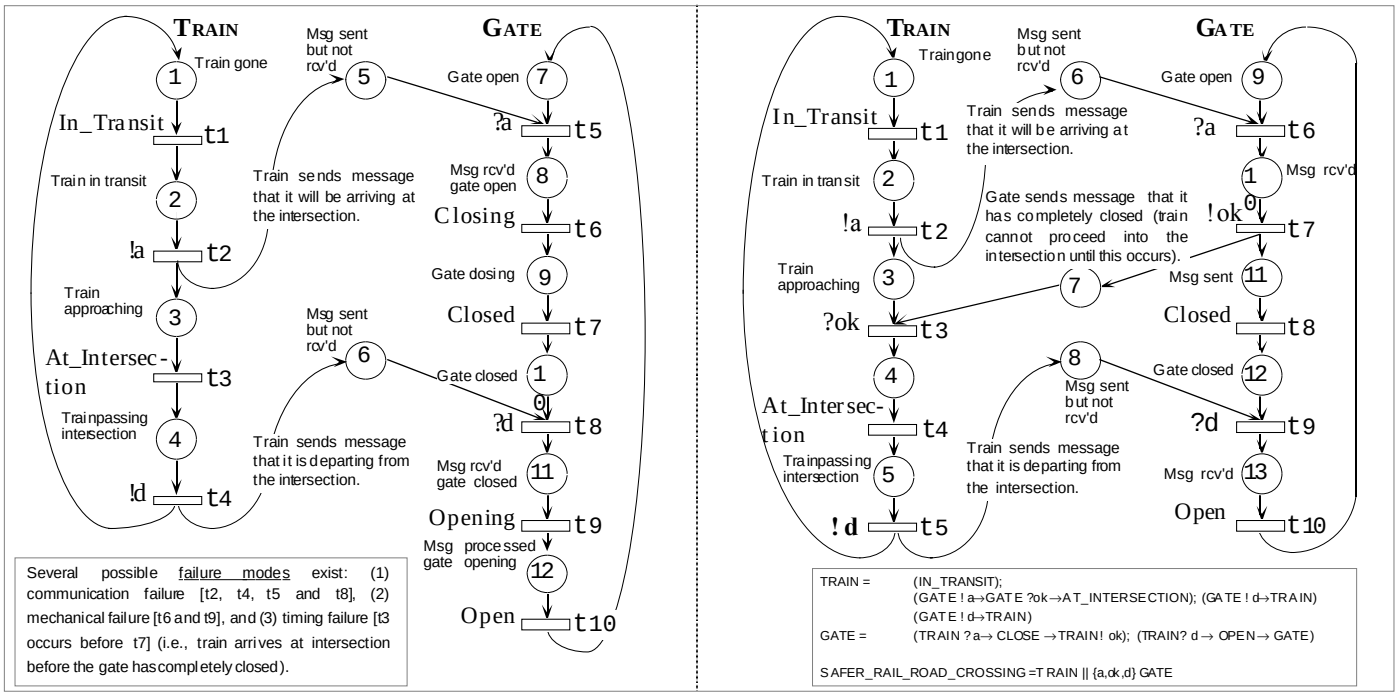


Figure 3. Train crossing with a timing hazard (left) and with hazard removed (right).

Constructing the PN and deriving the feasible markings reveals that the train process could enter the intersection before the gate closes. Some mechanism is needed to ensure the train will not proceed unless the gate is closed. The system could be re-designed to force the train to wait until the gate completely closes and thus provide a safer environment. Neither timing nor hard failures could cause an unsafe condition if additional synchronization is provided to notify the train that the gate is closed (i.e., the train would not proceed unless the notification was received). This assumes only one train can approach until the current train has departed. If this assumption is not made, another train could overcome a train waiting to receive a notification, which itself could be lost! Failure of any communication related actions may lead to a deadlock (train halts), but additional synchronization between the train and gate eliminate the possibility of trains passing through the intersection while the gate is open.[4] It is possible to use stochastic reward nets (and performability analyses) to associate a cost with delays in opening the gate.

Table 1. Feasible markings of the trains crossing with timing hazard (Figure 3).

Mkng	1	2	3	4	5	6	7	8	9	0	1	2	Description of marking (sate of the system)
M1	1	0	0	0	0	0	1	0	0	0	0	0	Train gone (idle state), gate open.
M2	0	1	0	0	0	0	1	0	0	0	0	0	Train in transit (i.e., in transit), gate open.
M3	0	0	1	0	1	0	1	0	0	0	0	0	Train sends *approaching* msg, gate open.
M4	0	0	1	0	0	0	0	1	0	0	0	0	Train approaching, msg rcvd and gate open.
M5	0	0	1	0	0	0	0	0	1	0	0	0	Train approaching, msg rcvd and gate closing
Mcf*	0	0	0	1	1	0	1	0	0	0	0	0	Train at intersect, approaching msg not rcvd!
Mtf*	0	0	0	1	0	0	0	0	1	0	0	0	Train at intersect, msg rcvd and gate closing!
M6	0	0	1	0	0	0	0	0	0	1	0	0	Train approaching, gate closed.
M7	0	0	0	1	0	0	0	0	0	1	0	0	Train passing intersection, gate closed.
M8	1	0	0	0	0	1	0	0	0	1	0	0	Train gone, depart msg sent and gate closed.
M9	1	0	0	0	0	0	0	0	0	0	1	0	Train gone, depart msg rcvd and gate closed.
M10	1	0	0	0	0	0	0	0	0	0	0	1	Train gone, depart msg rcvd, gate opening.
Mncf**	0	1	0	0	0	0	0	0	0	1	0	0	Train gone, but gate fails to open properly!

*Critical condition: Train at intersection but gate is still open **Non-critical condition: Train departed intersection but the gate is still closed

3.3 Failure Modes of the Train

In Figure 3, any transition can fail (at most one token is allowed per place). The PN for the gate is expanded (compared to the train) to emphasize the various phases that occur (places P_8 and P_9 as well as P_{11} and P_{12} could be combined to eliminate transitions t_6 and t_9). These markings show two types of failure (1) critical [M_{cf}] and, (2) non-critical [M_{nc}]. Within the M_{cf} grouping, we've included both *communication-related* and *mechanically-related* failures. One additional category is identified M_{tf} (tf for *timing-related* failure) which can be both critical and non-critical. Actually the M_{nc} [non-critical] category may include failure mechanisms of all three types. Why should different failure mechanisms have separate markings (or states in the corresponding Markov state diagram)? In doing so, it is easier to (1) explain failure causality for the purpose of prevention/avoidance, (2) ascertain their contribution to the overall reliability of the system, and (3) to distinguish if their manifestation could be catastrophic.

To consider timing criticality, note the slow firing of transitions (?a [t_5]), (Closing [t_6]), and (Closed [t_7]) allows the train to enter the intersection prior to complete gate closure. Similarly, transitions (?d [t_8]), (Opening [t_9]), and (Opened [t_{10}]) may incorrectly allow the train to have departed and while the gate is

[4] If we assume that the gate always opens and closes sooner than the time it takes the train to reach the crossing, the PN can be viewed as hazard free (except for the possibility of an unsafe mechanical failure). Also, failure to open the gate is not safety critical, yet should be avoided to avoid traffic congestion.

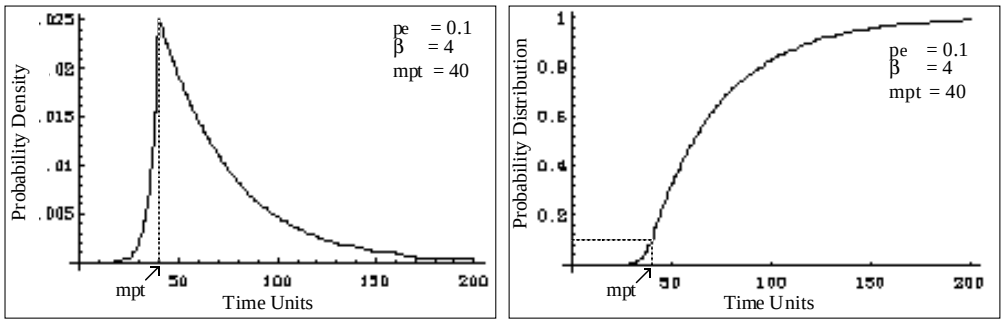


Figure 4. Train pdf and failure distribution function.

closed. Missing from the PN are the corresponding places P_{cf}, P_{tf}, and P_{nc} that consume the failed transition's token. When a transition fails (i.e., its associated action does not complete), the token is consumed but does not fall into an operational place. Instead, the token causes a failure transition to fire and the token falls into an absorbing output place. *For simplicity, these transitions and places are omitted.*

The CSP model (and the corresponding PN) can be augmented to show how such failures should be handled. For example, the communication failures can be handled using time-out and re-transmit. Figure 3-right shows synchronization between the TRAIN and GATE will eliminate the possibility of trains passing through the intersection while the gate is open. Failure to OPEN is not safety critical and is not considered. In Figure 3 (left-to-right), transitions $t_{6,7}$ and $t_{9,10}$ are combined into t_7 and t_{10} (note the augmented CSP model: "SAFER_RAIL_ROAD_CROSSING" shown).

3.4 Sensitivity Analysis

Using standard techniques provided by stochastic PN tools (e.g., SPNP[3,4]), discrete and continuous analyses is performed.[5] The reliability of the train crossing is computed with various failure rates and service rates (e.g., speed of the train, gate close/open rate). The values used in (and hence the results of the analysis) are for illustrating the approach (i.e., no real empirical significance can be attached to the failure rates, MTTFs etc.). These analyses are useful in exploring different fault-handling mechanisms and the cost-benefit of providing fault tolerance. The analysis techniques presented here are generally applicable.

The sensitivity of parameters associated with the train crossing is presented: (1) derivation of timing failure probability, (2) cost function minimization, (3) discrete and (4) continuous Markov reliability analysis. The results of item 1 provide for graphing a cost function and, via item 2, choosing an appropriate timing that will ensure the right level of reliability with respect to the costs. Choosing an some *optimal* timing criteria, we can integrate the timing failure probabilities (or rates) into the overall system model (items 3 and 4). The system model assumes any of the transitions can fail by any of several other failure modes (i.e., mechanical and communication, etc.)

[5]The classic steady-state solution method for stochastic models that maps GSPN models to CTMCs is compared with a method based on DTMCs in [Ciardo et al.

3.4.1 Timing Fault Probability

Analysis of failure rates (and cost functions) requires knowledge of the probability density functions (pdfs) associated with the train travel time and the gate closing time.[6] Lets assume that the pdf for the train travel time (θ) is given by $P_\theta(\theta)$, and the pdf for the gate closing time (η) is $P_\eta(\eta)$. The joint pdf of θ and η is $P_{\theta,\eta}(\theta,\eta)$. If θ and η are statistically independent then $P_{\theta,\eta}(\theta,\eta)= P_\theta(\theta)P_\eta(\eta)$. In this case, since both the train and gate operate independent of each other, it can safely be assumed that their probabilities are independent. The failure condition occurs iff, $\theta < \eta$ (i.e., when the train arrival time is less than the gate closing time). Thus, the probability of failure is,

$$p_{failure} = \int_{\eta=0}^{\infty} \int_{\theta=0}^{\eta} p_{\theta,\eta}(\theta,\eta)\, d\theta d\eta \,.$$

Since θ and η are statistically independent then,

$$p_{failure} = \int_{\eta=0}^{\infty} p_\eta(\eta) \int_{\theta=0}^{\eta} p_\theta(\theta)\, d\theta d\eta \,,$$

$$p_{failure} = \int_{\eta=0}^{\infty} p_\eta(\eta) \cdot F_\theta(\eta)\, d\eta \quad (1).$$

Where, $F_\theta(\eta)$ is the probability distribution function of the train arrival time and presents the probability that $\theta < \eta$ (the gate closing time). The same result can be obtained using conditional probabilities. Lets assume that the gate closing time is fixed and given as H, then the probability of failure can be given as

$$p_{failure}(H) = p(\theta < H \mid \eta = H).$$

Again, if θ and η are statistically independent, then,

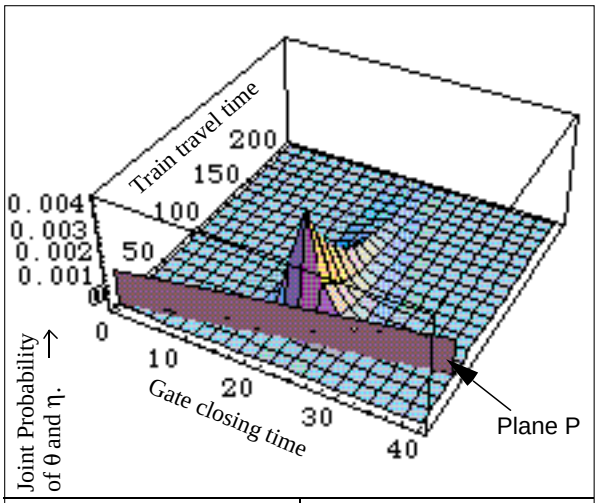
$$p(\theta < H \mid \eta = H) = p(\theta < H) = F_\theta(H)$$

The total probability of failure would be, $p(\theta < \eta) = p_{failure} = \int_{H=0}^{\infty} p_\theta(H) F_\theta(H)\, dH$, same as (1).

In a real circumstance, the pdf of θ and η would be known based on empirical knowledge, but this was not the case for our train example. Yet, lets consider some plausible analytical pdf from the stand point of an engineer who is responsible to procure such a system.

Typically railroad tracks have a set speed limit that would determine the distance from the gate at which the "close gate" message would need to be transmitted. This distance must be set so as to reasonably assure that enough time will elapse for the gate to close. If the speed of the train was constant, then the train arrival time will be a constant. However, there will be some variability from one run to the next which, over a period of say 25 years (with say five trains/day), represents a fair amount of uncertainty. The sources of variability (e.g.,

[6]Train travel time begins from when the open message is transmitted and ends when the train arrives at the intersection.



Gate parameters:	Train parameters:
$p_e = 0.3$, $\beta = 2$, gmpt = 20	$p_e = 0.1$, $\beta = 4$, mpt = 40

Figure 5. Joint pdf for the train crossing.

prevailing winds and other weather effects, time of the day, weight, type of load, and various human factors) can be grouped together as "undetermined" random events. Lets assume that if no random events occur, then the arrival time would be some nominal value, call it MPT (for most probable arrival time).[7] Lets also assume that the deviation around MPT (lower case mpt to shorten equations) is not symmetrical and the pdf of the deviation is exponential given by,

$$p(\theta) = \begin{cases} p_e\left(\frac{1}{\beta}\right)e^{(\theta - mpt)/\beta} & if\ \theta < mpt \\ p_e\left(\frac{1}{\beta}\right)e^{\left(p_e(mpt-\theta)/\beta(1-p_e)\right)} & if\ \theta \geq mpt \end{cases}$$

Where p_e is the probability of the train arriving earlier than the mpt, β^2 is the variance of being early. In addition, lets assume the pdf for the gate $P(\eta)$, is similar except that different parameter values (i.e., different p_e, β, and mpt) are used. Figure 4 shows the pdf for the train travel time for mpt=40, β=4, and p_e=0.1 (area under the curve for $mpt<\theta<\infty$ is 0.9 while the total area under the curve is 1.0) and the probability distribution of θ ($F_\theta(\Theta)$)is shown. Where $F_\theta(\Theta)$ is the probability of train travel time being less than Θ. Figure 5 shows the joint pdf of θ and η. Note that the probability of failure is the volume under that part of the surface where $\theta < \eta$. This is shown by cutting the surface in Figure 5 with plane P. The $p(\theta<\eta)$ is the volume under the surface in the foreground this side of P.

3.4.2 Cost Function

A cost function is useful for finding optimal values of design parameters. The function itself relates a cost to some tangible property of the system (e.g., time, reliability, failures, etc.). In general, calculating the probability of failure is useful in understanding the balance between reliability and the cost of failure. The cost function should be defined with parameters that can be measured and/or altered to minimize the potential losses (or costs). For example, more elaborate fault-handling and fault-recovery mechanisms could be used to tolerate or prevent safety critical failures, while less attention may be paid to non-safety critical failures. Failure to open the gate may cause long delays for waiting traffic but such failures can be handled by providing less expensive mechanisms that may, for instance, allow the gate to be opened manually. Conversely, failure to close the gate is more severe, so the benefit of using more elaborate mechanisms (e.g., increased redundancy, testing, and verification) is worth the expense to ensure the system uses an inherently more reliable design.

Lets assume a simplified cost function with only 3 components, the cost of delaying traffic, the cost of improving the gate reliability or speed, and the cost of a failure. The cost function is,

$$Q = \omega p(failure) + \phi \int_0^\infty \theta p_\theta(\theta) d\theta + \nu \qquad (2)$$

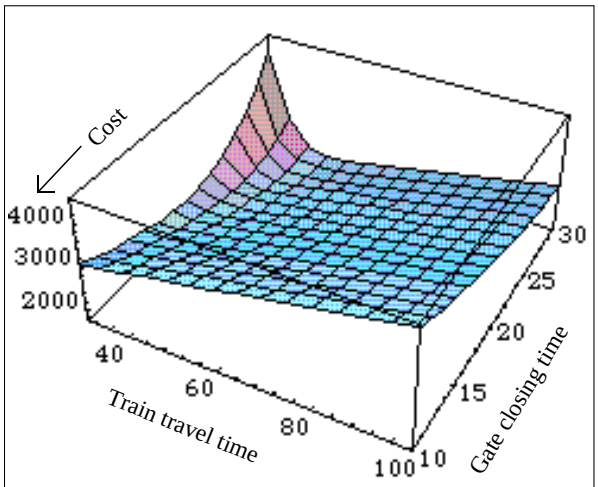
where ω = cost of failure, ϕ = cost of delay/time units, ν = cost of the gate / train passing and, the

$$\text{average train travel time} = \int_0^\infty \theta p_\theta(\theta) d\theta . \qquad (3)$$

The delay is defined from the time the gate receives the arriving message until the train arrives at the intersection. We did not consider the gate opening time and the train length (i.e., assume a train length of zero). Notice that this amount of time only depends on the train parameters. Equation 4 shows the gate cost per run as a function of the gmpt (gate most probable closing time):

$$\nu(gmpt) = \frac{(40 - gmpt)^4 + 20{,}000}{100} . \qquad (4)$$

There is an initial cost associated with installing the gate (20,000 cost units) and gate cost rises as gate speed increases.



Cost parameters: ω = 10,000, ϕ= 20 / time unit.
Train parameters: p_e = 0.1, β = 4
Gate parameters: p_e = 0.3, β = 2

Figure 6. Cost function for train crossing.

[7]The time taken by the train to pass by the intersection (once it has arrived) is ignored since that is dependent on the length of the train and is independent from the design parameters for the Train Crossing.

The denominator's value of 100 is the number of times that the gate operates before failing (a hypothetical equation for demonstration purpose and is not based on empirical knowledge). The total system cost function is multidimensional. It incorporates all of the parameters of the system (mpt, gmpt, β, β_g, p_e, p_{eg} [g subscript stands for gate parameters]). In practice, a subset of the parameters should be selected by the designer in such a way, so as to minimize the overall operating cost. Figure 6 is a 3-D plot of the cost as a function of mpt and gmpt. All other parameters are assumed to be fixed known values. Using this function a designer can minimize the cost by selecting the right values of mpt and/or gmpt. For example, if the gate's closing time was fixed at 30 time units, then the optimal train arrival time is around 52 time units. Knowing the optimal train arrival time and speed, easily gives the distance when to send the open gate request.

For smaller values of mpt and larger values of gmpt the cost is mostly influenced by the cost of failure (mpt = 30 and gmpt = 30). For small values of gmpt (such as gmpt = 10) the cost is mostly influenced by both the cost associated with delaying traffic (i.e., delay cost increases as the train travel time increases) and the cost to build a gate that can close faster. Choosing both mpt and gmpt would give the an optimal mpt $\cong 45$ and gmpt $\cong 25$ (values are exaggerated to make the variations of the cost function more visible).

3.4.3 Discrete Analysis

Table 2 presents the probability assignments for the train crossing ignoring deadline related failures (i.e., P_{tf}=0). Four trials were run with differing failure probabilities (where P_c= communication failure, P_m= mechanical failure). In all runs $P_m > P_c$, and in order to reduce the probability of critical failure in runs 2-4, we set P_m(close) > P_m(open) by the factors of 100, 3 and 5 respectively.

Hypothetically, such reliability improvements are possible using fault-tolerant methods. Consequently, the probability of critical failures (P_{cf}) are reduced by the factors of 17.573, 1.975 and 2.974 respectively. Such analyses showing the magnitude of improvement associated with a given design (or model) option may be useful for deciding what level of fault tolerance is appropriate. Note, P_{ncf} is the non-critical failure probability and the MTTF is given in the number of discrete steps (or time units).

3.4.4 Continuous Analysis

The CTMC results are based on the markings of Table 1. States represented by the markings M1, M2, M6, and M7 are *safe* because these states are transient and do not (directly) give rise to failures (do not transition to absorbing states). The mechanical (λ_m), communication (λ_c) and timing (τ) failure rates are shown associated with their transition arcs. The trade-off between the rate of train arrivals (μ_1), speed of the train (μ_3), rate of the gate mechanism (μ_6, μ_9) and the failure rates were investigated.

Table 2. Discrete analysis (P_{tf}=0).

Description	**Run 1**	**Run 2**	**Run 3**	**Run 4**
$\mathbf{P_c}$	.0001	.00001	.0001	.00001
$\mathbf{P_m}$**(close)**	.01	.00001	.01	.001
$\mathbf{P_m}$**(open)**	.01	.001	.03	.005
Results from DTMC-based analysis:				
$\mathbf{P_{cf}}$	0.5026	0.0286	0.2544	0.1690
$\mathbf{P_{ncf}}$	0.4974	0.9714	0.7456	0.8310
MTTF	490.26	9524.07	248.19	1656.21

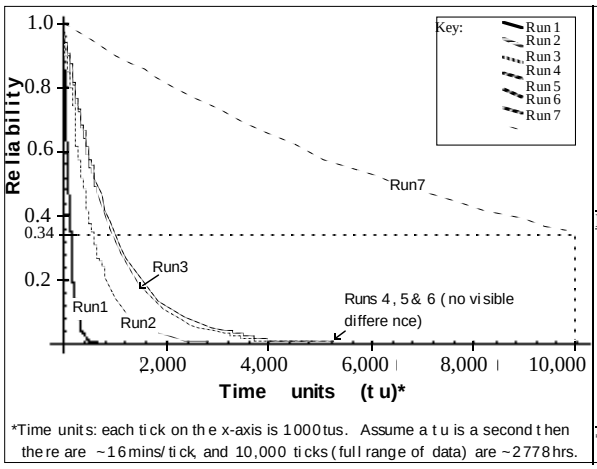
The unreliability of communications do not significantly impact the MTTFs because the failure rates are much less than the mechanical failure rates (i.e., $\lambda_m = 0.0001 > \lambda_c = 0.0000001$). Mechanical failures and the possibility of the gate not closing (opening) in time are assumed to be greater contributors to the unreliability of the system. In looking at the data (input parameters and the results Figure 7) an interesting relation is evident. Observe that, if the train's speed tends to bring it to the intersection sooner than the gate has had time to close, then an improvement in the gate's mechanical reliability does not help. To improve the overall system's reliability it is more important to provide the additional synchronization between the train and gate processes as described in paragraph 3.1, so as to avoid the possibility of having the gate miss its deadline (τ_5).

It is important to see how the least reliable entity impacts the overall reliability. In Figure 7, there are incremental improvements seen in Rel[10,000] (the reliability of the system at 10,000 time units) from 10^{-40} to 10^{-5} (see how τ_5 has been manipulated in runs 1 - 3), but by run 4, a point of diminishing return is reached. The next most significant gain in system reliability comes when the gate's failure rate is improved by a factor of ten (compare run 6 and 7 in the graph). The MTTF has improved by 6 times while the corresponding system reliability changes from ~$2.6\text{x}10^{-5}$ to ~$3.3\text{x}10^{-1}$!

4. SUMMARY AND FUTURE WORK

This paper describes an approach to developing CSP models that are translated into SPNs for the purpose of reliability and performance analyses. Such translations give insight into failure modes and fault/failure handling mechanisms that can be carried back and prescribed as a part of the CSP model. This approach provides feedback to guide the specifier in further refinements. Such guidance can help to avoid certain errors and evaluate the costs of providing certain features and/or structures for fault-avoidance and/or fault-tolerance. This approach is illustrated using an example that considered three different failure mechanisms (communication, mechanical, and timing failures). These failure types were further classified as safety critical or not. The failure rates used in this example (hence the results of the analysis) are for illustrating the approach, no other significance should be attached.

The ***CSP-based*** models are automatically translated into Petri nets (PNs) using the CSPN tool developed by the author and his students. CSPN uses a set of algorithms which codify the canonical CSP-to-PN translations rules based on the CSP grammar discussed in paragraph 2.3 (also see [Kavi et al. 96, 95, 94]). A CSP-based model transformed into a Petri net can



Results

Run1....Rel[10,000]= 4.58042×10^{-40}	Mttf= 1.09934×10^{6}tus
Run2....Rel[10,000]= 4.58554×10^{-9}	Mttf= 5.20472×10^{6}tus
Run3....Rel[10,000]= 1.07427×10^{-5}	Mttf= 8.73755×10^{6}tus
Run4....Rel[10,000]= 2.34974×10^{-5}	Mttf= 9.37937×10^{6}tus
Run5....Rel[10,000]= 2.56342×10^{-5}	Mttf= 9.45662×10^{5}tus
Run6....Rel[10,000]= 2.58888×10^{-5}	Mttf= 9.46547×10^{6}tus
Run7....Rel[10,000]= 3.44604×10^{-1}	Mttf= 6.15169×10^{6}tus

Input Parameters:**

1. $\tau 5$=0.00908	$\lambda 3, 4, 8, 9$=1.0 x10-7	$\lambda 5, 10$=1.0 x10-4
2. $\tau 5$=0.000908	$\lambda 3, 4, 8, 9$=1.0 x10-7	$\lambda 5, 10$=1.0 x10-4
3. $\tau 5$=0.0000908	$\lambda 3, 4, 8, 9$=1.0 x10-7	$\lambda 5, 10$=1.0 x10-4
4. $\tau 5$=0.00000908	$\lambda 3, 4, 8, 9$=1.0 x10-7	$\lambda 5, 10$=1.0 x10-4
5. $\tau 5$=0.0	$\lambda 3, 4, 8, 9$=1.0 x10-7	$\lambda 5, 10$=1.0 x10-4
6. $\tau 5$=0.0	$\lambda 3, 4, 8, 9$=0.0	$\lambda 5, 10$=1.0 x10-4
7. $\tau 5$=0.0	$\lambda 3, 4, 8, 9$=0.0	$\lambda 5, 10$=1.0 x10-5

**Constants: $\mu 1$= 0.0001, $\mu 2$-4, 7, 8= 1.0, $\mu 9$, 10= 1.0, while $\mu 5$ and $\mu 6$ = were held set at 0.1 and 0.01 respectively.

Figure 7. System reliability as a function of operational time.

be more conveniently characterized in terms of time critical behavior using off-the-shelf SPN tools for Markov reward analysis. The results can be related back to the original model and used as a basis for refining the model in light of prior runs. CSPN is being extended into a Prototype Open Toolkit for composing and analyzing models of software and systems so-called DUO. This environment will more easily facilitate the modeling process and support the analyst with regard to the model's sensitivity to different parameter values and structures. Some additional features also include a (portable) graphical user interface, translation of Promela-based models into SPNs (using the C-based Stochastic Petri net Language [CSPL] as a Meta language). This idea is conveyed by Figure 8 [Allmaier 97].

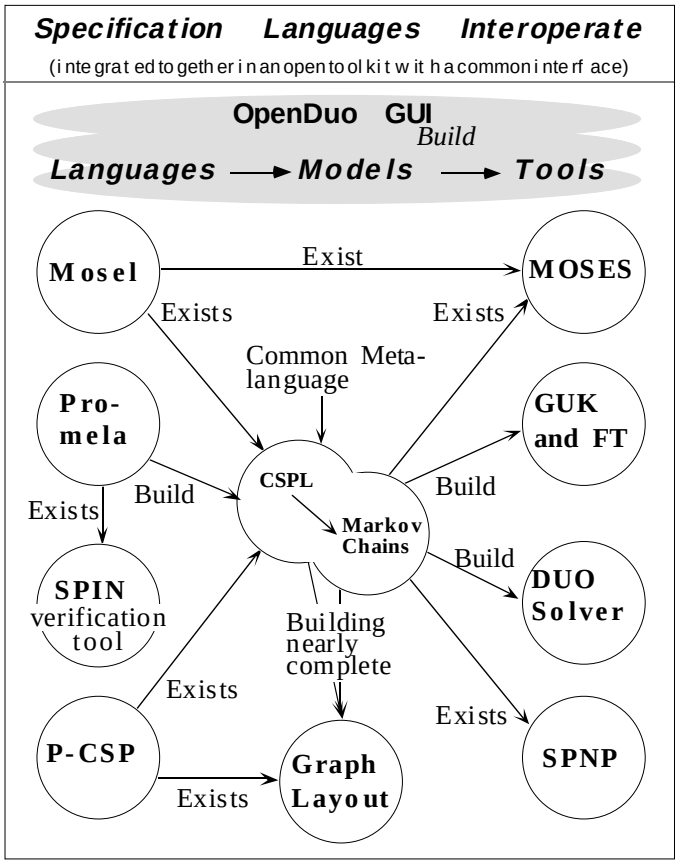


Figure 8. Translations of DUO.

5. REFERENCES

Allmaier, S.C. Kowarschik, M. and Horton, G., "State Space Construction and Steady-State Solution of GSPNs on a Shared-Memory Multiprocessor," *PNPM*, (Durham), 1997.

Bolch G., Greiner S., deMeer H., Trivedi K.S., *Queuing Networks and Markov Chains -- Modeling and Performance Evaluation with Computer Science Applications*, to be published by John Wiley & Sons in Spring 98.

Chiola, G., "A Software Package for the Analysis of Generalized Stochastic Petri Net Models" PNPM (Kyoto), 136-143, Dec. 1989.

Ciardo, G., Muppala J. and Trivedi, K.S., "On the Solution of GSPN Reward Models," *Performance Evaluation*, 12, 237-253, 1991.

Ciardo, G., Muppala J. and Trivedi, K.S., "SPNP: Stochastic Petri Net Package," *PNPM* (Kyoto), 144-151, Dec. '89.

Gerber, R. and Lee, I., "A Proof System for Communicating Shared Resources," *IEEE Proc. Real-Time Systems Symp*., 288-299, 1990.

Hoare, C.A.R., *Communicating sequential processes*, Prentice-Hall Int'l, London, 1985.

Jones, Geraint and Goldsmith, Michael, *Programming in occam 2*, Prentice-Hall Int'l. Series in Comp. Sci., New York, 1988.

Kavi, K.M., Sheldon, F.T. and Reed, S.C., "Specification and Analysis of Real-Time Systems Using CSP and Petri Nets," *Int'l Journal of Software Engineering and Knowledge Engineering*, June 1996.

Kavi, K., Sheldon, F.T., Shirazi, B., and Hurson, A.R., "Reliability Analysis of CSP Specifications Using Petri Nets and Markov Processes," *HICSS-28*, (Honolulu), Jan. 1995.

Kavi, K.M. and Sheldon, F.T., "Specification of Stochastic Properties with CSP," *Int'l. Conf. on Par. and Distr. Sys*., Taiwan, 288-293, Dec. 1994.

Lee, I., Bremond-Gregoire, P., and Gerber, R., "A Process Algebraic Approach to the Specification and Analysis of Resource-Bound Real-Time Systems," *Proc. of the IEEE*, 158-171, Jan. 1994.

Marsan, M.A., Donatelli, S. and Neri, F., "GSPN Models of Multiserver Multiqueue Systems," *PNPM* (Kyoto), 19-28, Dec. 1989.

Marsan, M.A., Balbo, G. and Conte, G., "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems," *ACM Trans. on Computer Systems*, 2(2), 93-122, 1984.

Murata, T. "Petri Nets: Properties, Analysis and Applications," *Proc. of the IEEE*, 541-580, Apr. 1989.

Olderog, Ernst-Rudiger, "Operational Petri Net Semantics for CCSP," *LNCS-266*, Springer-Verlag, 196-223, 1987.

Olderog, E.R., TCSP - Theory of communicating sequential processes, *LNCS-255*, Springer Verlag, 1986.

Ostroff, J.S., "Formal Methods for the Specification and Design of Real-time Safety Critical Systems," *Jr. of Systems and Software*, 33-60, April 1992.

Sheldon, F.T., "Specification and Analysis of Stochastic Properties for Concurrent Systems Expressed Using CSP," Ph.D. Dissertation, CSE Dept., University of Texas at Arlington, May 1996.

Shin, K.G. and Kim, H., "Derivation and Use of Deadline Information in Real-Time Control Systems," *In Foundations of Dependable Computing: Models and Frameworks Dependable for Systems*, Kluwer, 77-110, 1994.

Sorensen, E., Nordahl, J. and Hansen, N., "From CSP Models to Markov Models," *IEEE Trans. on SE*, 554-570, June 1993.

Tomek, L., Mainkar, V., Geist, R.M., and Trivedi, K.S., "Reliability Modeling of Life-Critical Real-Time Systems," *Proc. of the IEEE*, 108-121, Jan. 1994.

BIOGRAPHY

Dr. Sheldon is an Assistant Professor. He received the Ph.D. from the University of Texas at Arlington (UTA) where he was supported by a NASA Fellowship from Langley Research Center and was a NRC Research Associate. He received the Outstanding Research by a UTA Ph.D. Student award in Computer Science and Engineering (1995-96) and the 1996-97 Outstanding Ph.D. Dissertation award from the UTA Chapter for the Society of Sigma Xi Scientific Research Society. He is very pleased to be a 1997-98 Research Associate in the Stanford / NASA Ames ASEE Summer Faculty program. He worked as a Software Engineer while attending graduate school for 9 years in the area of avionics and diagnostics software development for the YF-22, F-16 and Tornado aircraft programs at General Dynamics and Texas Instruments.

He received two Bachelor of Science degrees from the University of Minnesota (Minneapolis-St. Paul) in Microbiology and Computer Science. His reseach interests are to develop and validate methods and supporting tools for the creation of safe and correct software. He is a member of the IEEE Computer and Reliability Societies, ACM, AIAA, and The Tau Beta Pi and Upsilon Pi Epsilon Scholastic Honor Societies.