Department of Electrical and Computer Engineering

SAMPLE DISSERTATION PROPOSAL

on

Algorithm and Specification for Reliability Analysis of Systems with Dependent Failures

Submitted by:

Signature: _____ *Date:*

Graduate Committee:

Signature: _____ *Date:*

Committee Member:

Signature: _____ *Date:*

1. Background

Many real-world systems are subject to dynamic behavior such as sequence-dependence in which the order that fault events occur is important, and common-cause failures which are multiple dependent component failures resulting from a shared common-cause. As an example of sequence-dependent failures, consider a system shown in Figure 1. The system has one primary unit (*M*) and one standby spare unit (*S*) connected with a switch controller (*Sw*). The system can continue to operate when the switch controller fails after the primary unit fails as the standby is already in use. However, if the switch controller fails before the primary unit fails, then the system fails upon the failure of the primary unit as the standby unit cannot be switched into active operation [1]. Thus, the failure criteria of the system depend not only on the combinations of events, but also on the sequence in which events occur. Traditional static fault trees [1, 2] cannot capture such behavior. In order to model the sequence-dependent behavior, a priority-AND (pAND) gate has been proposed in the dynamic fault tree (DFT) reliability analysis [1, 3, 4].

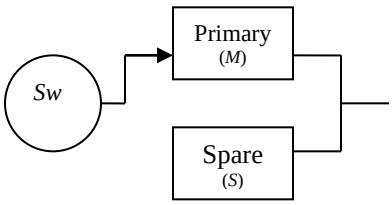


Figure 1. An example of sequence dependent systems

The pAND gate is a dynamic gate that is logically equivalent to an AND gate along with an added condition that events must occur in a specific order. As shown in Figure 2, a pAND gate has two inputs *A* and *B* whose output is true if both *A* & *B* have occurred, and *A* occurred before *B*. The gate will not fire if either of the two events has not occured, or if *B* occured before *A*.

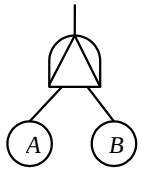


Figure 2. The pAND gate

Figure 3 illustrates the DFT model of the sequence-dependent system of Figure 1. It shows that the system fails when both the primary unit and the standby unit have failed, or when both the primary unit and the switch have failed and the switch fails before the primary unit fails.

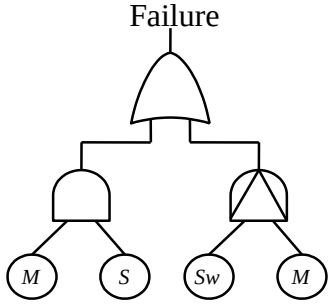


Figure 3. DFT of the example sequence dependent system

Common cause failures (*CCF*) [5] are simultaneous failures of multiple components due to a common cause (*CC*) like sabotage, flood, earthquake, power outage or human errors. It tends to increase the joint-failure probabilities and can have dominant contribution to the system unreliability and performance. In this project, we will perform formal specification of an efficient algorithm proposed for addressing *CCF* in the system reliability analysis.

Formal methods [6] are mathematics-based techniques for describing system properties. They provide frameworks for specifying, developing, and verifying systems in a systematic, rather than ad hoc manner. A method is formal if it has a sound mathematical basis, typically given by a formal specification language. A formal specification language [6, 7] provides a notation (its syntactic domain), a universe of objects (its semantic domain), and precise rules defining objects that satisfy certain specification. It also provides the means of precisely defining properties like consistency, completeness, and more relevantly, specification, implementation, and correctness.

2. Technical Discussion

Sequence -Dependent Failures Analysis

Typically, the DFT with pAND gates can be solved by automatic conversion to an equivalent Markov model. However, the Markov-based methods are subject to the well-known state-space explosion problem and typically require exponential time-to-failure distribution for the system components. Therefore, they are generally applicable for systems with very limited size.

To mitigate the state-space explosion problem of the Markov-based methods, a modularization technique [2, 8, 9] has been proposed to analyze a large dynamic system via a divide-and-conquer strategy, where the system is divided into smaller and independent subtrees. If a subtree contains a dynamic gate, then it will be solved using a Markov model; otherwise, it will be solved using a combinatorial method called Binary Decision Diagrams (BDD) [10, 11]. Solutions of all the subtrees will be integrated to obtain the solution for the entire fault tree model, *i.e.*, the entire system unreliability. However, in practice the modularization technique may not work well for complex systems with lots of repeated or shared events and a high degree of interdependence [12].

Monte Carlo simulation [13] represents another class of methods that have been used to solve dynamic fault trees. The simulation-based methods can offer great generality in representation and solution to highly complex and dynamic systems. However, the simulation-based methods have certain limitations. They can only offer approximate results. They often involve long computational time, especially when results with high degree of accuracy are desired. They also require a completely new simulation to be performed whenever the input failure parameter value changes. Bayesian network approach is another method proposed for the DFT analysis [14]. However, it has the same complexity problem as the Markov-based methods.

Recently, an analytical method based on inclusion-exclusion (IE) formulation [15,16] has been proposed to analyze a DFT with pAND gates, where the set of minimal cut sets and/or cut sequences is first generated from the DFT specifications, and is then combined using the IE formula to obtain the system unreliability. The major problem of this method is it requires enumeration or *a priori* knowledge of the minimal cut sets/ sequences, which is often a process with exponential complexity. Also, the IE-based method in [15] assumes the exponential time-to-failure distribution for the system components.

In order to overcome the limitations of the above described existing methods, a combinatorial and analytical method [12] has been proposed. The method can offer an exact and efficient solution to the reliability analysis of non-repairable systems with the sequence dependent behavior, without requiring the enumeration or *a prior* knowledge of the minimal cut sets/sequences. Also, this method does not have any limitation on the type

of time-to-failure distributions for the system components. In [12], the necessity of an efficient algorithm to generate the complete orders/sequences from the partial oders/sequences for the evaluation of the system unreliability has been pointed out. Hence, in this research work, we address the above need by proposing a complete sequence generation algorithm based on topological sorting. Several examples are given to show the applications of the proposed algorithm.

Common-Cause Failure Analysis

Basically there are two different approaches for incorporating *CCF* into system analysis: explicit and implicit methods. The explicit method models *CCF* as shared basic events in the system fault tree, then applies the conventional fault tree analysis approach to analyze the fault tree with *CCF* basic events. In the implicit method, the fault trees are built without considering *CCF*, and the algebraic system unreliability expression is derived in a certain form. Such an expression is then evaluated in a way that the contribution of *CCF* is correctly included [5]. An example of the implicit method is the efficient decomposition and aggregation (EDA) approach [17].

EDA approach uses the divide and conquers strategy. It decomposes an original system with *CCF* into a number of reduced systems based on the total probability theorem. The set of reduced systems needs not consider the dependencies introduced by *CCF* as the effects are factored out. The results of all reduced system reliability can be aggregated to obtain the entire system reliability measure considering the *CCF*. The EDA approach can be summarized as shown in Figure 4.

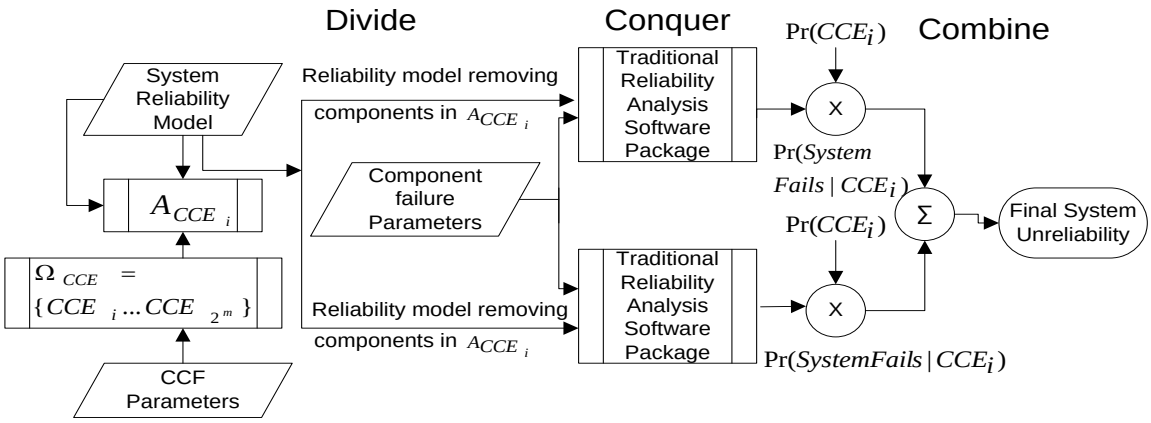


Figure 4. The EDA approach

The approach consists of three major steps as discussed below:

Step 1. Build a Common Cause Event (*CCE*) Space

Suppose there are '*m*' Common Cause (*CC*)s existing in the system, the '*m*' *CC*s partition the event space into the following 2^m disjoint subsets, each called a common-cause event (*CCE*):

$$CCE_1 = \overline{CC}_1 \cap \overline{CC}_2 \cap \ldots \cap \overline{CC}_m$$
$$CCE_2 = CC_1 \cap \overline{CC}_2 \cap \ldots \cap \overline{CC}_m$$
$$\ldots\ldots\ldots\ldots$$
$$CCE_{2^m} = CC_1 \cap CC_2 \cap \ldots \cap CC_m$$

Hence, the *CCE* space can be denoted by $\Omega_{CCE} = \{CCE_1, CCE_2, \ldots, CCE_{2^m}\}$. Let A_{CCEi} denote the set of components affected by CCE_i which is the union of a set of components affected by each occurring *CC*.

Step 2. Generate and Solve Reduced Problems

The unreliability of the system can be calculated using the total probability theorem given by:

$$U_{sys} = \sum_{i=1}^{2^m} \Pr[\, systemfails \mid CCE_i]P[CCE_i]$$

Here, the problems to be solved (i.e., conditional probabilities Pr[*systemfails*| *CCE*$_i$]) are actually reduced reliability problems in which the components affected by *CCE*$_i$ are removed and *CCF* needs not to be considered. Fault tree of each reduced problem is generated by replacing each basic event affected by *CCE*$_i$ with a constant logic value '1' (*True*), and then applying a Boolean reduction to the system fault tree in which all the components affected by *CCEi* do not appear.

Step 3. Aggregation

Finally, the combination for system reliability considering *CCF* can be calculated using $U_{sys} = \sum_{i=1}^{2m} \Pr[\, systemfails \mid CCE_i]P[CCE_i]$

Formal Methods

Formal methods, such as assume-guarantee based compositional reasoning methods have been widely used in formal verification of large-scale and complex system. They have been used to present new rules for assume-guarantee reasoning and ensure the soundness and completeness of the system. Compositional reasoning [18] is a reduced approach for reasoning about a system's components which are carried out in assume-guarantee paradigm [19] where each component guarantees certain properties based on assumptions about the other components.

Earlier work [20] contributes a case study on the use of formal specification in the collaborative development of a dynamic fault tree analysis tool. Similar work on formalizing DFTs can be found in [21], where a semantics is described using the Z specification language. However, no precise specification for modeling how sequence of failure events cause system failures like common cause failures (*CCF*) has been developed before. Hence, another part of this research work is an attempt to bridge the gap between formal methods and dynamic system reliability analysis for systems subject to common-cause failures. In particular, the divide and conquer paradigm that has been used in formal methods is applied to dynamic system reliability analysis in the form of the EDA approach, which will be formally specified using the Z formal specification language, as described next.

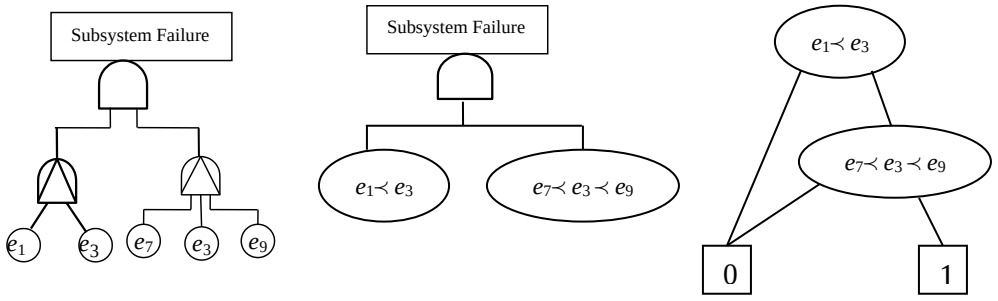
The Z notation [21, 22] is a language for expressing formal specifications of computing systems. It supports structuring and composing complex expressions in the first-order typed set theory. The typical structure of a Z Schema contains schema name, schema signature and schema predicate. The schema name specifies the name of the Z schema. The schema signature introduces state variables, and the schema predicate defines the constraints on them, which is written in the lower part of a schema. These constraints should always be true for the state variables. The operations performed on the state variables are defined in operation schemas.

3. Problem Statement

The combinatorial apporach proposed in [12] for addressing sequence dependent failures integrates an analytical solution for considering pAND dependence at the lower level, and a Sequential BDD (SBDD)-based solution for representing the system structure function at the upper level. This approach can be implemented as three-step process :

a. Transformation of system DFT model
b. Generation of the system SBDD model
c. Evaluation of the system SBDD model

Based on this approach, while evaluating the system SBDD model, we may obtain a path that involves more than one sequential event. For example, Figure 5(a) shows the DFT model of a subsystem with two PAND gates, each representing a sequential event. Figure 5(b) shows the transformed DFT model. Figure 5(c) is the SBDD model generated from the transformed DFT model.



(a) DFT model *(b) Transformed DFT* *(c) SBDD model*

Figure 5. An example of systems with two sequential events.

In Figure 5(c), there is one path to the sink node '1': ($e_1 \prec e_3$)→ ($e_7 \prec e_3 \prec e_9$)→'1'. And the two events involved in this path are not

independent becasue they share the same event e_3. For such cases, when we calculate the path probability, we must generate the complete sequences from the presented partial sequences for considering the dependence between them. For example, to calculate $\Pr\{(e_1 \prec e_3).(e_7 \prec e_3 \prec e_9)\}$, we must expand the partial sequences $(e_1 \prec e_3)$ and $(e_7 \prec e_3 \prec e_9)$ into complete sequences over all the four basic events as $(e_1 \prec e_7 \prec e_3 \prec e_9)$ and $(e_7 \prec e_1 \prec e_3 \prec e_9)$. The resultant complete sequences must preserve all the ordering constraints imposed by the input partial sequences.

One major component of this project ***is to propose an algorithm to implement the generation of the complete orders/sequences from the partial orders/sequences***, which is a necessary and significant task for accomplishing the reliability analysis of sequence dependent systems using the approach of [12].

Little work has been done to link formal methods and dynamic system reliability analysis for complex systems, though formal methods can potentially provide more reliable and efficient solutions than existing approaches to the reliability analysis of dynamic systems. Another component of this work is ***to bridge the gap between formal methods and dynamic system reliability analysis via the formal specification of algorithms for systems subject to common-cause failures***.

4. Approach

The proposed complete sequence generation algorithm is based on topological sort [23], which is a method of arranging the vertices in a directed acyclic graph (DAG) as a sequence such that no vertex appears in the sequence before its predecessor. For a DAG, we define **in-degree** of a vertex as the number of arrows going into the vertex and **out-degree** as the number of arrows coming out of the vertex. In the context of precedence constraints, the in-degree refers to the number of predecessors of a vertex and the out-degree refers to the number of successors of the vertex. Next, we describe the proposed algorithm as a five-step procedure.

Generation Algorithm:

1. Initialization: set up an array *R* that records the in-degree value of each vertex in a DAG. Initially, the in-degree values are all set to zero.
2. Update the in-degree of each vertex according to partial sequences. Specifically, search each partial sequence. Except the first vertex in the sequence, for each of the remaining vertex appearing in the sequence, increase its in-degree by 1 and update the array *R*.

3. Let Q be the queue used to keep track of vertices with in-degree of zero. If Q contains more than one element with in-degree of zero, then split Q into $n!$ queues Q_i, where n is the number of elements in Q. Each Q_i contains a distinct permutation of the n elements.
4. For each Q_i, define $R_i = R$, as long as Q_i is not empty, do the following
   a. Visit each vertex of the queue and move it to an array F_i.
   b. Delete that vertex from the array R_i.
   c. If the vertex has a successor, then decrease the in-degree of its successor and update array R_i.
   d. If the in-degree of that successor becomes zero then keep that successor at the end of queue Q_i . If Q_i contains more than one element with in-degree of zero, then do the following:
      - Split Q_i into $n_i!$ queues Q_{ij}, where $j = 1, ..., n_i!$ and n_i is the number of elements with in-degree of zero in Q_i. Each Q_{ij} contains a distinct permutation of those n_i elements.
      - Set up an array F_{ij} for each Q_{ij}, and initialize it to be the current F_i. Then delete F_i.
      - Also, set up an array R_{ij} for each Q_{ij}, and initialize it to be the current R_i. Then delete R_i.
      - Go back to 4(a).

   Note that when iteration in Step 4 is performed for Q_{ij}, then corresponding arrays F_{ij} and R_{ij} will be used in those four sub-steps. In addition, it is possible that Q_{ij} can be split further into Q_{ijk}. Thus, corresponding arrays F_{ijk} and R_{ijk} will be set up and used in the subsequent operations. Similar split can be done further for Q_{ijk}.

5. Output the array F, each corresponding to a complete sequence.

Formal specification of the EDA approach:

To bridge the gap between formal methods and dynamic system reliability analysis, we illustrate our idea by formally specifying the EDA approach using Z notations. As already discussed in the previous section, the formal specification consists of a set of state schema and operation schema.

Figure 6 shows an example schema named “Common Cause Failure” which defines a set of *CC* (denoted as ‘causes’), and a set of components affected by a *CC* (denoted as ‘group’). It defines *CC(i)* as a type of Common Cause and a function *CCG* which maps the components affected for a certain *CC*. It also defines the array size in use as hwm (high water mark). The schema predicate defines the conditions on the state variables, which are invariants. The first predicate states that the set ‘causes’ consists of those CCs that occur somewhere among *CC(1)* to *CC(hwm)*. The case for the set ‘group’ is also similar. The second predicate state that the set ‘causes’ is same as the domain of the function *CCG*, i.e., the set of *CC* to which it can be validly applied. Similarly, Z state schema will also be defined for Common Cause Event Space.

Common Cause Failure

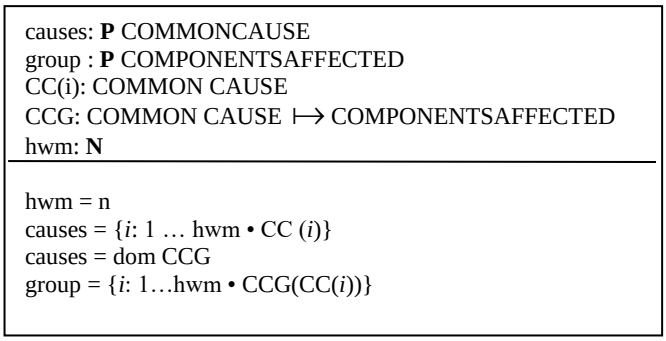


Figure 6. The Z state schema 'Common Cause Failure'

The Z operation schema will be defined for generating all the *CCE*s and getting *CCE* for each *CC*. The operation schema will also be defined for generating the probability of those *CCE*s and getting the components affected by those *CC*s. Finally, operation schema will be defined for evaluating the system unreliability. Hence, all the steps of the EDA approach can be formally defined using the Z notations.

5. Schedule/ Milestones

Feb 2009	Study on Compositional Reasoning and EDA Approach.
March 2009	Study on Formal Specification and Z language.
April-May 2009	Preparation of formal Specification for EDA Approach using Z language.
June-August 2009	Preparation of the paper “Bridging The Gap Between Formal Methods and Dynamic Reliability Analysis”. Working model of Phased Mission System Reliability Analysis Software (implemented in Java platform).
Sep 2009	Literature survey on Sorting algorithm and complete sequence generation.
Oct 2009	Development of the sequence generation algorithm for pAND gate.
Nov-Dec 2009	Presentation of the algorithm to DCN group. Refinement of the generation algorithm. Modeling of the algorithm for implementation.
Jan 2010	Draft preparation of the paper “Complete Sequence Generation Algorithm for Reliability Analysis of Dynamic Systems with Sequence-Dependent Failures”. Draft preparation of Thesis Proposal.
Feb 2010 (First two weeks)	Refinements of the draft paper. Refinements of the proposal.
Feb 15-Apr 2010	Software Implementation of the generation algorithm

	(in Java platform).
May 2010	Testing of the developed software.
June-July 2010	Thesis documentation.
Aug 2010	Thesis defense.

6.Conclusion

The proposed complete sequence generation algorithm is based on topological sort [23], which is a method of arranging the vertices in a directed acyclic graph (DAG) as a sequence such that no vertex appears in the sequence before its predecessor. For a DAG, we define **in-degree** of a vertex as the number of arrows going into the vertex and **out-degree** as the number of arrows coming out of the vertex. In the context of precedence constraints, the in-degree refers to the number of predecessors of a vertex and the out-degree refers to the number of successors of the vertex. Next, we describe the proposed algorithm as a five-step procedure.

7. Bibliography

[1] J. B. Dugan and S. A. Doyle, “New results in fault-tree analysis,” *Tutorial notes of the Annual Reliability & Maintainability Symposium*, Jan 1997.

[2] K. B. Misra (Editor), *Handbook of Performability Engineering*, Springer-Verlag, London, ISBN: 978-1-84800-130-5, Oct. 2008.

[3] J. B. Dugan, S. J. Bavuso, and M. A. Boyd, “Dynamic fault-tree models for fault-tolerant computer systems,” *IEEE Transactions on Reliability*, vol. 41, no. 3, pp. 363-377, Sep 1992.

[4] J. B. Fussell, E. F. Aber, and R. G. Rahl, “On the quantitative analysis of priority-AND failure logic”, *IEEE Transactions on Reliability*, vol. R-25, pp 324-326, Dec 1976.

[5]J. K. Vaurio, "An implicit method for incorporating common-cause failures in system analysis," *IEEE Transactions on Reliability*, vol.47, no.2, pp.173-180, Jun 1998.

[6] J. M. Wing, "A specifier's introduction to formal methods," *Computer*, vol.23, no.9, pp.8, 10-22, 24, Sep 1990.

[7] D. Coppit and K. J. Sullivan, "Formal specification in collaborative design of critical software tools," *Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium*, Washington, D.C., November 13-14, 1998, pp. 13-20.

[8] R. Gulati and J. B. Dugan, “A modular approach for analyzing static and dynamic fault trees,” *Proceedings of the Annual Reliability &*

Maintainability Symposium, Philadelphia, PA. pp. 568-573, Jan 1997.

[9] S. V. Amari, G. Dill, and E. Howald, “A new approach to solve dynamic fault trees,” *Proceedings of Annual Reliability & Maintainability Symposium*, pp. 374-379, Jan. 2003.

[10] R. Bryant, “Graph based algorithms for Boolean function manipulation,” *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677-691, Aug 1986.

[11] L. Xing and J. B. Dugan, "Analysis of generalized phased mission system reliability, performance and sensitivity," *IEEE Transactions on Reliability*, vol. 51, no. 2, pp. 199-211, Jun. 2002.

[12] L. Xing, A. Shrestha, and Y. Dai, "Exact Combinatorial Analysis of Dynamic Systems with Sequence-Dependent Failures," *IEEE Transactions on Dependable and Secure Computing* (under review).

[13] W. Long, T. L. Zhang, Y. F. Lu, and M. Oshima, “On the quantitative analysis of sequential failure logic using Monte Carlo method for different distributions,” *Proceedings of Probabilistic Safety Assessment and Management*, pp. 391-396, 2002.

[14] H. Boudali and J. B. Dugan, “A discrete-time Bayesian network reliability modeling and analysis framework,” *Reliability Engineering & System Safety*, vol. 87, no. 3, pp. 337-349, Mar 2005.

[15] T. Yuge and S. Yanagi, “Quantitative analysis of a fault tree with priority AND gates,” *Reliability Engineering & System Safety*, vol. 93, no. 11, pp. 1577-1583, Nov. 2008.

[16] D. Liu, C. Zhang, W. Xing, R. Li, and H. Li, “Quantification of cut sequence set for fault tree analysis,” *HPCC2007, Lecture Notes in Computer Science,* no. 4782, pp. 755-765, Springer-Verlag, 2007.

[17] L. Xing, “Reliability modeling and analysis of complex hierarchical systems”, *International Journal of Reliability, Quality and Safety Engineering (IJRQSE),* Vol. 12, No. 6, December 2005.

[18] N. Amla, E. A. Emerson, K. S. Namjoshi and R. J. Trefler, "Assume-Guarantee Based Compositional Reasoning for Synchronous Timing Diagrams”, *TACAS* 2001: Pages 465-479.

[19] T. A. Henzinger, M. Minea, and V. Prabhu, “Assume-guarantee reasoning for hierarchical hybrid systems,” *Proceedings of the Fourth International Workshop on Hybrid Systems: Computation and Control (HSCC)*, LNCS 2034, Springer, 2001, pp. 275-290.

[20] D. Coppit and K. J. Sullivan, "Formal Specification in Collaborative Design of Critical Software Tools," *Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium*, 1998

[21] D. Coppit, K. J. Sullivan, and J. B. Dugan, ”Formal semantics of models for computational engineering: A case study on dynamic fault trees,” *Proceedings of the International Symposium on Software Reliability Engineering*, pages 270–282. IEEE, Oct 2000.

[22] J. M. Spivey, *The Z Notation: A Reference Manual*, *International Series in Computer Science*. Prentice-Hall, New York, N.Y., second edition, 1992.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (2nd Edition)*. The MIT Press, 2001.