

HIMAP: Architecture, Features, and Hierarchical Model Specification Techniques

Murari Sridharan, Srinivasan Ramasubramanian, and Arun K. Somani
Dependable Computing Laboratory
Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011
email: {murari, rsrini, arun}@iastate.edu

Abstract

Accurate analysis of complex systems for their reliability and performance is a challenging task. Several tools for such analysis have been developed. Combinatorial methods such as fault trees and reliability block diagrams have been used to analyze reliability of systems employing components with two states. For components with more complex behavior such as including repairs, or different fault and error handling methods, Markov Chains have been used. Fault trees, fault trees with fault and error handling models, fault trees with repair specification, and Petri Nets are used to describe the behavior of such complex systems. In many cases, this behavior is converted into Markov Chains to analyze. In other cases, simulation-based methods are employed. Development of powerful and easy-to-use tools allow engineers and system designers to accurately model such systems. Several tools have been developed addressing these needs, HIMAP is one such tool that helps analyze system behavior that are used in multi-phased missions and in systems that employ scheduled maintenance. The tool accounts for the effects of phase changes, that may include configuration and behavior changes, and maintenance of components. In this paper, we describe the most important features of HIMAP and hierarchical model specification techniques that allow easy development and solution of models.

1 Introduction

As systems become more complex, it is important to ensure reliable operation in critical applications. Developing powerful and easy-to-use tools which allow engineers and system designers to model such systems is the need of the hour. Modeling helps the designer in making necessary changes to the configuration of the system or parameters governing the reliability of the system, to achieve desired reliability levels. This calls for powerful methods to specify, model and analyze the system in an efficient and user-friendly manner. State-space based description for modeling system becomes cumbersome as systems grow larger. Fault trees, fault trees with specification of component repairs, and Petri nets can serve as representations of Markov chains and are easier to manage.

HIMAP is an X-Windows based graphical modeling and analysis tool, implemented in the C

programming language. HIMAP is the product of over 5 years of research in building and improving tools for modeling and analyzing the reliability of complex systems. This work started at the University of Washington and is now being continued at Iowa State University. HIMAP provides an ideal modeling environment as it allows the designer to specify the system as a Fault tree, a Fault tree with component repairs, a Markov chain or a Petri net. The modeler has the flexibility to choose the modeling method as appropriate to the system. HIMAP allows these high level representations to be solved directly or convert them to a markov chain and use existing tools to solve them. HIMAP is portable on all platforms supporting X-Windows. We are currently developing a Windows version of HIMAP in Visual C++ for PC's.

This paper is arranged as follows. We describe specification techniques for modeling hierarchically in the HIMAP modeling environment and build an example to illustrate the modeling process. Section 2 gives an overview of the features in HIMAP modeling environment [10]. Section 3 talks about specification techniques for hierarchical modeling. Section 4 concludes this paper.

2 The HIMAP Modeling Environment

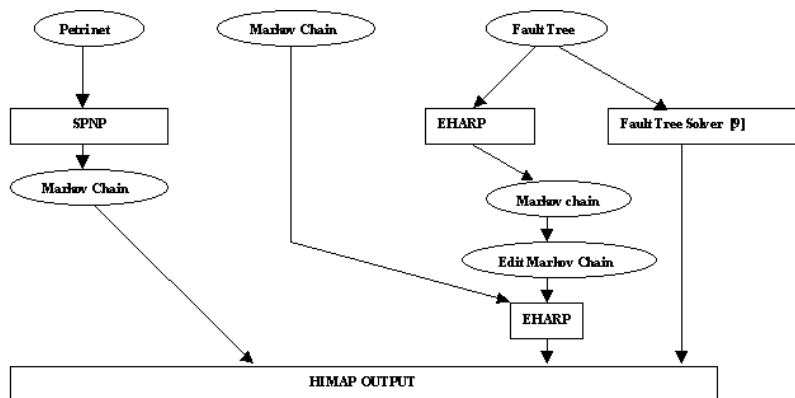


Figure 1: The HIMAP Modeling Environment

Several tools have been developed to analyze systems. These tools accept specification of systems in one of the following ways:

- Fault Tree : HARP [1], EHARP [2], SDM [3], CAFTA [4], SETS [5]
- Markov Chain : HARP [1], EHARP [2]
- Petri net : UltraSAN (uses Stochastic Activity Nets) [6], SPNP [7], FIGARO [8].

Most of these tools are text based and not very user friendly except a few like UltraSAN which provide a user-friendly modeling environment. HIMAP allows the designer to model the system as a Markov chain, a Fault tree, a fault tree with component repairs or as a Petri net as shown in Figure 1. Each of these modeling techniques has its advantages and HIMAP exploits these to provide the designer with maximum flexibility in modeling.

2.1 The Markov Chain Environment

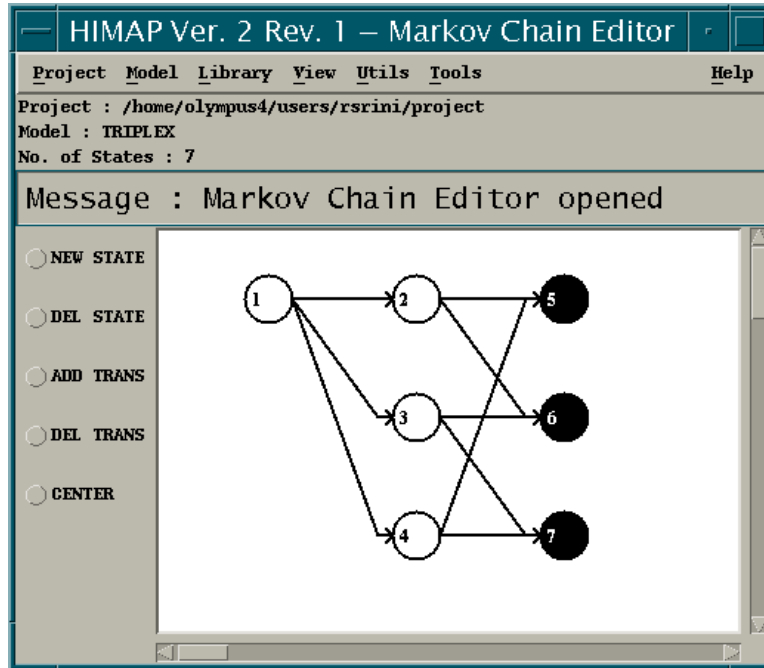


Figure 2: Markov Chain For Triplex System Without Replication

Markov chain is a popular mathematical tool for modeling and analyzing systems. Graphically as shown in Figure 2, a markov model consists of states and transitions. The states contain information about the number of operational components, and the transitions are rates at which specific components or subsystems fail causing a change in the state of the system. Computations are done to determine the probability of being in a state at different times. The reliability of the system can then be determined by adding the probabilities of the operational states. HIMAP provides a user-friendly environment for specifying and solving a Markov chain description of a system. Since the Markov chain can be considerably large, apart from the standard view where the whole Markov chain generated is displayed, HIMAP has a provision of viewing the transitions emanating from any state, the single state view, as shown

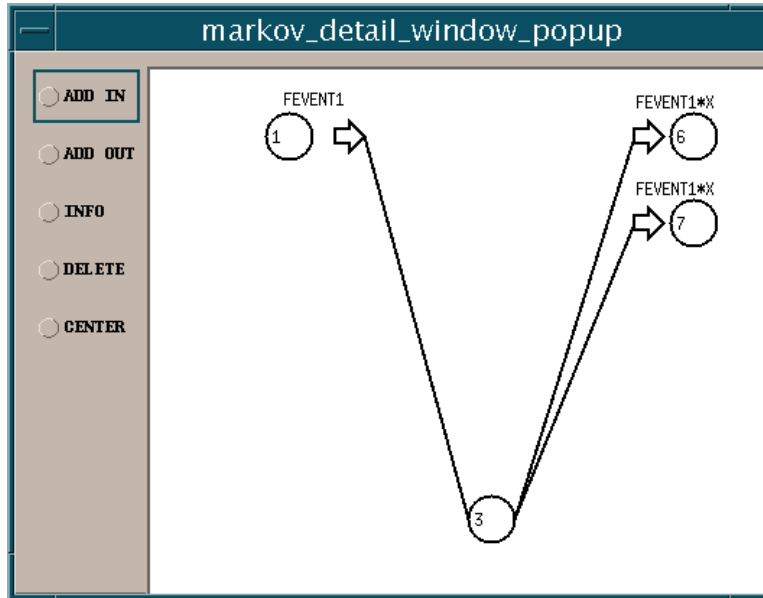


Figure 3: A Single-state view of the Markov Chain

in Figure 3.

2.2 The Fault Tree Environment

A fault tree provides a formal graphical representation of the combination of events, which can lead to system failure [Figure 4]. Fault trees are easy to understand and create as they represent the logical construction of the system. The top event (system fail state) is resolved into its constituent causes connected by the AND, OR and M-out-of-N, and other logic states which are then further resolved until basic events are identified. Basic Events directly relate to physical components in the system. HIMAP also supports the Functional dependency gate, the sequence dependence gate, and the Cold Spare gate. To ease the effort needed in the modeling of highly complex systems HIMAP has a feature called IN gate. This feature allows the designer to divide the system hierarchically into simpler subsystems, which can be linked through IN gates.

The modeler is provided with the flexibility of reusing models by instantiating models already created. Instantiation is the process of deriving a specific model from a base model. These instantiations can share events or they can be independent. The instantiation process is described in Section 3. To reduce the size of a model, HIMAP allows statistically identical components to be combined into a single basic event. A replicated basic event is labeled with

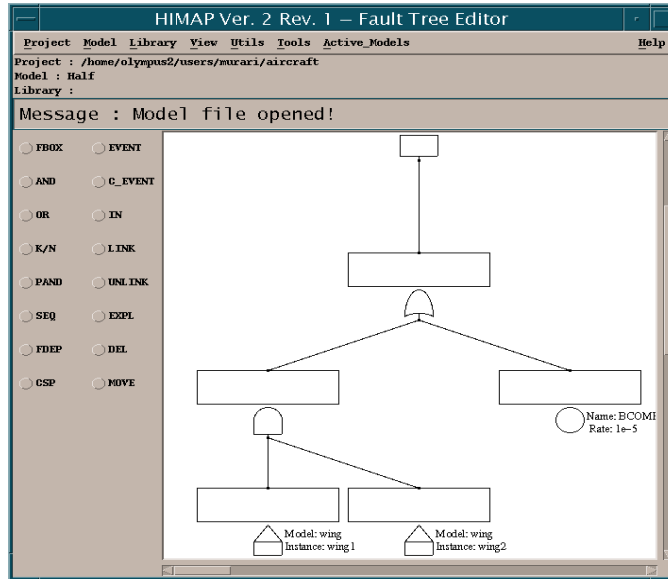


Figure 4: The Fault Tree Editor

and expression of the form $m*n$, representing 'm' replications of redundant, functionally identical components of type 'n'. Replication is useful in modeling statistically identical components with the same failure rate value. Replication considerably reduces the size of the Markov chain representation of the fault tree. Figure 2 shows the Markov chain representation of Triplex system without replication. The basic events in fault tree are associated with components from the library and the IN gates are instantiated from existing models.

2.3 The Library Concept

The Library concept is unique to HIMAP. The library forms the main specification for HIMAP. The information in the library defines the characteristics of each component, which is going to be used in the mission. HIMAP *allows importing of components from any of the compatible database packages and it chooses the relevant fields for the appropriate component.*

Components are individual entities, which contribute to the overall reliability of the system. Parameters like name, version, failure rate, repair rate are associated with each component [Figure 5]. The idea of version is described in Section 2.7 under Phase Mission System. Failure distribution can be either exponential or weibull, or a fixed probability value or value from any other sub-model. The components are chosen from a library database depending on system requirement. The modeler has the choice of creating a new library for the system he is modeling

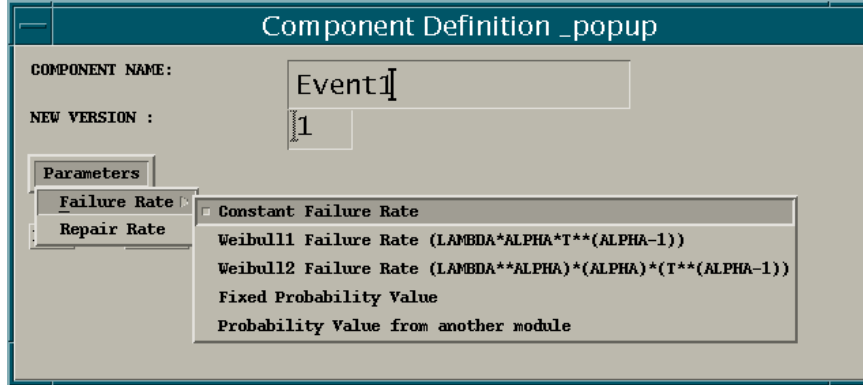


Figure 5: The Library Component Definition Window

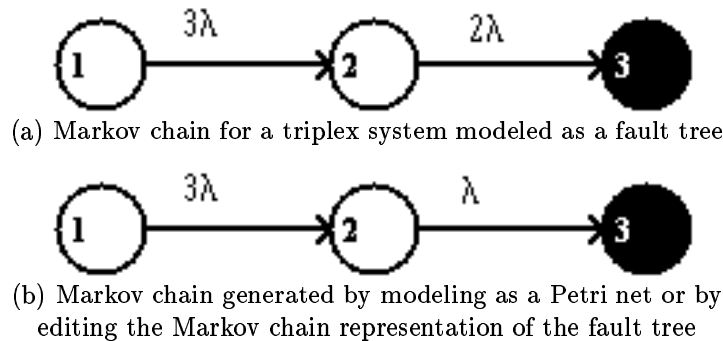


Figure 6: A Triplex System

or he could import the components from an existing library.

2.4 Fault Trees with component repairs

Fault trees, by themselves, cannot be used to model repairs in the system as they are inherently acyclic. HIMAP tackles this problem in the following ways:

- by allowing the Markov Chain (generated from the fault tree representation) to be edited to introduce repair arcs,
- by allowing the designer to declare whether the component can be repaired. (at the time of associating the pictorial view with the physical components).

HIMAP allows the specification of repair rate as well as the number of repair persons associated with the repair of each component or a group of components. When this is done, HIMAP automatically edits the Markov chain description, generated using the fault tree, and adds the appropriate repair arcs with the associated repair rates. HIMAP also supports the sharing of repair person among components, which can be repaired. This characteristic of HIMAP helps us model systems in a more realistic fashion.

2.5 The Petri net Environment

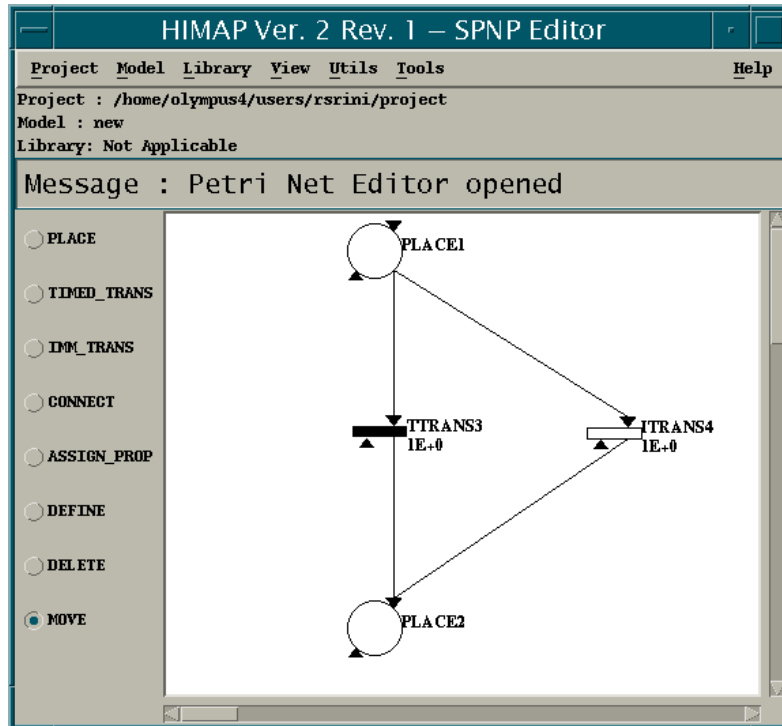


Figure 7: Petri net representation of a Triplex System

Petri nets though more complicated than fault trees in representation form a very powerful paradigm to model complex systems. HIMAP provides a very user-friendly graphical interface in which such modeling can be done. HIMAP supports constructs used by SPNP [7] in the specification of Petri nets. SPNP is the underlying Petri net solver used by HIMAP. In SPNP, models are specified in CSPL, which is an extension of the C programming language.

To make the modeling process easier, HIMAP supports constructs like *places* and *transitions* (*timed* and *immediate*), and relationship between them in the pictorial mode. To model these systems as Petri net more efficiently, HIMAP allows the declaration of global variables, constants, and functions that include reward type, rate type, probabilistic type and enabling functions (associated with each transition) *places* represent the state of the system, which may contain *tokens*. Each transition is associated with an exponentially distributed (*timed*) or a δ -function-shaped (*immediate*) firing time. A *priority* can also be associated with each transition. Reward functions are output functions, which are used to find the probability that the system is in a specified configuration. Enabling functions determine the conditions when a transition can fire. Petri nets are more powerful than the fault trees in specifying the behavior of the

system. Some systems cannot be specified using fault tree; Petri nets have no such limitations. For example consider a triplex system, If a channel fails, the designer may require the system to become a simplex system rather than duplex system as the duplex system is less reliable than the simplex system. This is easily modeled using a Petri net using immediate transitions [shown in Figure 7]. In the triplex system case, the immediate transition is modeled such that it puts one token from PLACE1 into PLACE2 whenever there are two tokens in PLACE1. This modeling is not possible using fault trees directly, however the designer can model the system as a fault tree(markov chain representation shown in Figure 6(a)) and edit the Markov chain generated by it (as shown in Figure 6(b)). Petri nets however lose out to fault trees in the simplicity of representation.

Stochastic Reward Nets (SRNs) are adopted as modeling tools because of their flexible, concise, and intuitive forms in comparison with other modeling methods. Hierarchical Stochastic Reward nets Solver Package (HSP)[14] has been developed and it provides a comprehensive environment to facilitate Stochastic Reward Net (SRN) analysis. This package is currently being included into HIMAP to facilitate SRN analysis.

2.6 The Model Development Process

The model development process in HIMAP consists of the following two steps:

- Creating a pictorial view
- Associating physical components with pictorial view

Pictorial view is used to specify the relationship among the components that leads to desired (or undesired) behavior. This can be specified using fault trees or Petri nets. HIMAP stores this information in a PIC file. Description of physical components used in designing a system is specified using a component database (described in next section). Association process relates the physical component properties with the pictorial view. Any complex behavior associated with the component can be described during this process. The modeler can specify redundancy, its Fault and Error Handling Model (FEHM), and repair information. The IN gates, which represent sub-models in the hierarchical model development process, have to be mapped to their corresponding models. The IN gate thus mapped is called as an instantiation of the models from which it is derived. The Associate Components window is shown in Figure 8. The information about every model is maintained in a MODEL file. In reality, the MODEL file is a pair of files one holding the pictorial information and the other one which relates the physical component properties with the pictorial view.

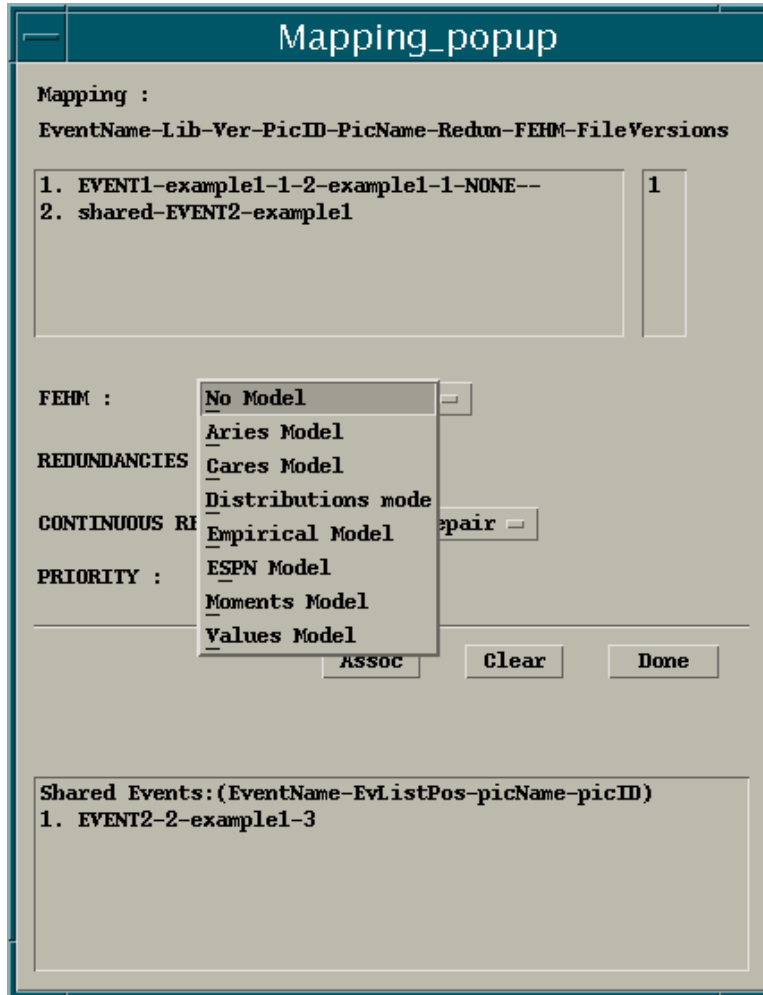


Figure 8: The Associate Components Window

2.7 Phase Mission System

In a real life scenario we expect the complete operation of a system (mission) to comprise of different phases. Each phase may require the functioning of a different set of components of the system. A component could also have different failure rates in different phases. For example, the failure rates of components in lift-off and cruise stages of a space shuttle mission are very different. HIMAP has the capability to divide the mission into phases and select components needed for each phase, use proper failure rates and find reliability of each phase [12]. This helps the analyst to get a more accurate picture of the reliability of the system rather than being very conservative or optimistic about it. HIMAP allows the designer to specify the starting and ending time of a phase with a certain mean and variance using the Beta distribution.

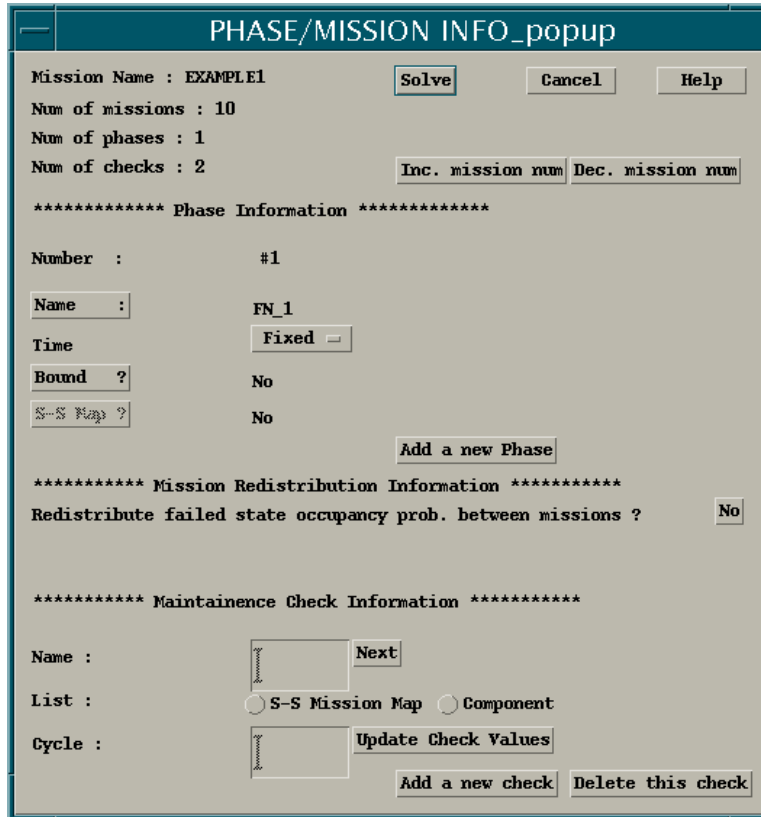


Figure 9: The Phase Mission Information Window

These individual phase reliabilities are then combined to obtain the reliability of the overall system. Phased-mission analysis cannot be accomplished by modeling the system as a Petri net in HIMAP.

In HIMAP the user first creates a generic model that uses all the components in the system. The 'Create Phase' facility in HIMAP is used to generate models for each phase. The modeler specifies the duration of each phase as a fixed or random interval. The phased-mission analysis is done using Markov chains. When system enters a phase from the previous one, a state in the previous phase corresponds to a state in the next phase. The user then has to specify whether there is a state-to-state mapping. The state occupancy probability between missions can also be specified. For multi-mission analysis, advanced users may be able to perform state-to-state mapping between missions by choosing S-S-Mission Map option. This is however only recommended for small state space system and very specific scenarios. The phase mission information window is shown in Figure 9.

2.8 Scheduled Maintenance (SM)

To maintain error free operation and avoid excessive faulty components, most redundant systems are checked and repaired at the end of regular pre-specified intervals of time, called maintenance check times to repair and restore faulty components. SM systems have a maintenance schedule, which determines the intervals at which its components are maintained during the life cycle of a system. Accurate modeling of SM systems is an important and complex problem in reliability analysis. HIMAP is provided with various techniques as described in [15] to accurately model SM systems.

More information on the HIMAP environment is provided in [13].

3 Hierarchical Modeling

Modeling complex systems is very involved, both in view of the amount of work needed to model it, as well as the size of the model that needs to be solved. This makes the solution process computationally intensive. The hierarchical modeling principles can be employed in two ways.

1. A system can be decomposed into smaller models based on component types or subsystems. These models can be solved separately, and the results can be combined to produce large system solution. This approach helps solve the largeness problem. This also makes modeling of large systems simpler and the modeler has the flexibility to analyze individual subsystems independently and be able to abstract their behavior at the next higher level. HIMAP allows the modeler to choose the environment (ie. fault tree or Petri net) most suited to specify each individual subsystem. This assumes that the behavior of the subsystem is independent of each other [11].
2. Even when subsystems are not independent, the model description may be identical. Moreover, the system may consist of several identical components or subsystems. This is likely to be the case when a fault tolerant or a system employing parallelism is being analyzed. In such cases, the subsystem description can be specified once, and used through instantiations, as many times as required. It is also conceivable that the pictorial view of the subsystem model is identical, but the physical association of components is different. In most cases, even most of the physical components may also be identical and only a few descriptions change. A modeling environment must allow specification of such systems as well. HIMAP specification approach allows a modeler to follow this methodology.

A sub-model can be used in two ways:

- It may be associated with more than one physical subsystems. In VLSI systems for example, modules A and B may be driven by different clocks. In this case the clock model is associated with 2 physical subsystems, one as input to module A and other as input to module B.
- The model may represent a single physical subsystem that is an input to many other subsystems. Take a variation of the example used in the previous case, the only difference now is that the same clock drives both the modules. In this case the clock model is associated with one physical subsystem which goes as input to both the modules.

The specification technique for hierarchical modeling in HIMAP provides a powerful modeling aid, which allows events to be shared among the instantiations of a model. We demonstrate this technique using an example. It is desired to specify a model only once. The modeler can then instantiate the unique model as many times as required.

Hierarchical modeling using fault trees is implemented in HIMAP by integrating specification of different subsystems into the main system through IN Gates (Fault tree mode) which serve as pipes transferring reliability characteristics of the subsystem into the main system.

To use hierarchical specification, the modeler decomposes the system behavior, creating sub-models which can be combined to solve the whole system. These sub-models may have some events that are shared. A sub-model itself can be common in different parts of the system. HIMAP facilitates specification of such commonalities and use of common sub-models. When a model is instantiated, the modeler has to specify whether it is a common sub-model which can be carried over different hierarchies. In such cases the IN gates which represent these sub-models are termed as 'Common IN gates'. HIMAP also maintains a list of all the models created in the model development process and the modeler can select from this list whenever a model which is already developed needs to be used. If a model has certain inputs which remain common with other models of the same type, the model need not be specified every time. The modeler can instead instantiate from the list of models which were already developed and can also selectively maintain commonality between them.

3.1 Example

To demonstrate our modeling technique, consider the example of modeling a Research Center. The Research Center has four Labs. The Research center would like all the labs running. Each Lab deploys Computer Systems of four types numbered 1-4. For simplicity, we assume that there are seven Computer Systems(CS) in each lab. The Lab operation requires at least 3 out-of 7 to remain fully operational. Each Computer System in turn has two processors, a

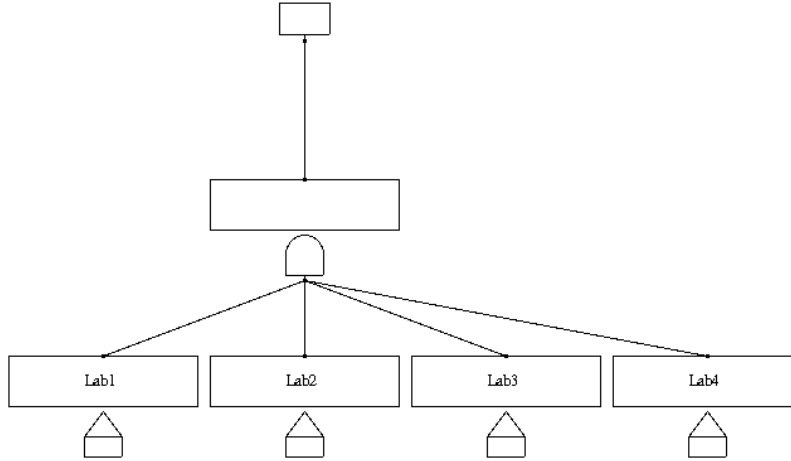


Figure 10: The Research Center Model

Power Supply(PS) and connects to an Ethernet (EC). The Computer System will fail when both the processors fail or the PS or the EC fails. For the operation of all labs there are 2 Power supplies (PS1,PS2) and 2 Ethernet Cables (EC1, EC2) which are shared among the Computer Systems. Computer Systems of type 1,2 (CS1,CS2) use PS1 and type 3,4(CS3,CS4) use PS2. Computer Systems of type 1,3 use EC1 and type 2,4 use EC2. Lab1 consists of 3 CS of Type 1 and 4 CS of Type 2. Lab2 consists of 3 CS of type 3 and 4 CS of type 4. Lab3 consists of 3 CS of Type 1 and 4 CS of type 4. Lab4 consists of 3 CS of Type 3 and 4 CS of Type 2. Our specification technique simplifies the modeling process as the modeler needs to describe each model only once.

The Research Center will fail even if one of the Labs fail. To describe this behavior, The Lab models are connected to an OR gate as shown in Figure 10. The Labs are modeled as IN gates and the modeler needs to associate the properties of each Lab type to the corresponding IN gate.

The Lab will fail if 4 out-of-7 Computer Systems fail. The Lab model is shown in Figure 11. The Computer Systems are modeled as IN gates and the modeler needs to associate the properties of the Computer System types to the corresponding IN gates. The IN gates are connected to a M-out-of-N gate. The values of M and N are specified by the modeler during the association process.

The Computer System model is shown in Figure 12. The Computer System will fail when both the processors (Processor1,Processor2) fail or Power Supply or Ethernet fails.

The basic models of Power Supply and Ethernet are shown in Figure 13. The failure rates of

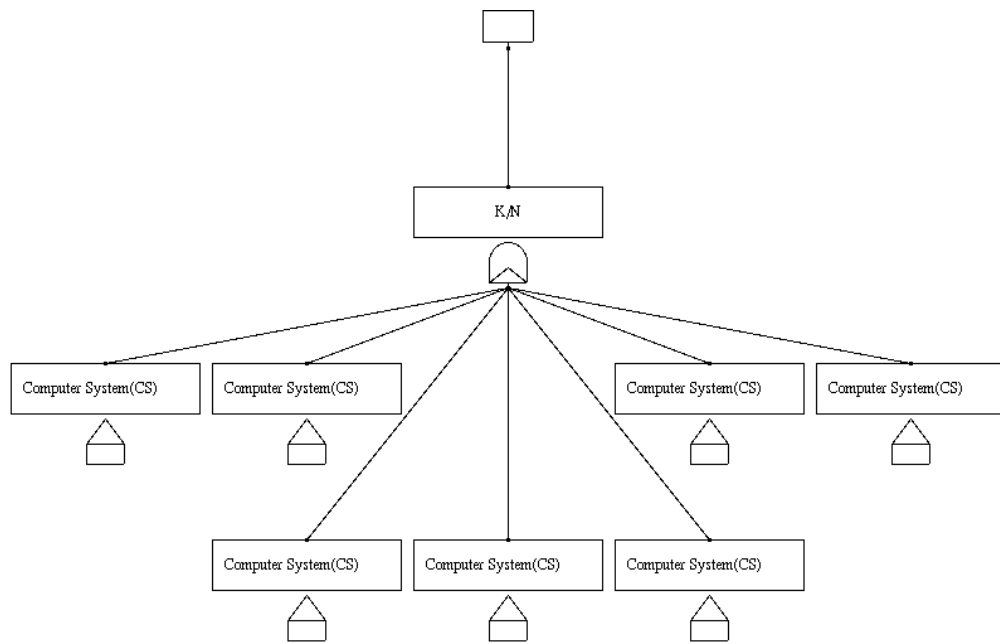


Figure 11: The Lab Model

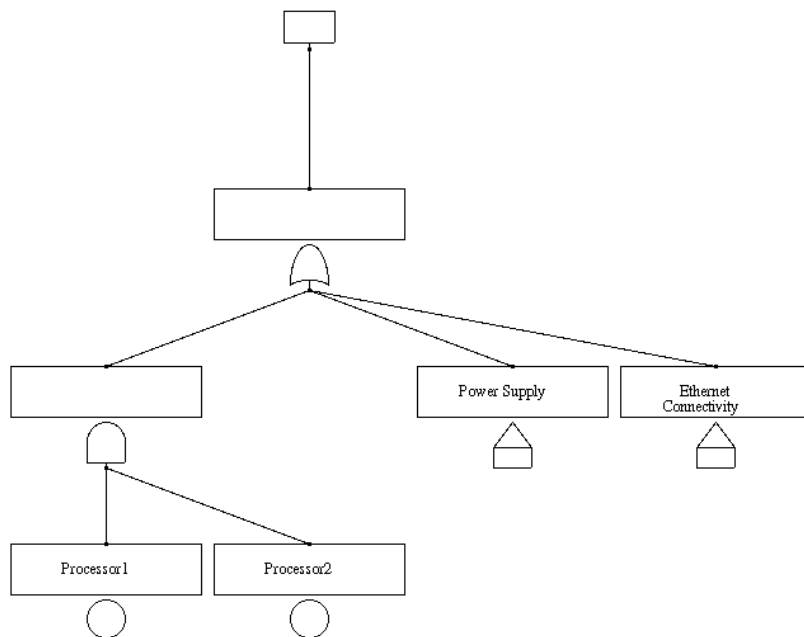


Figure 12: The Computer System Model

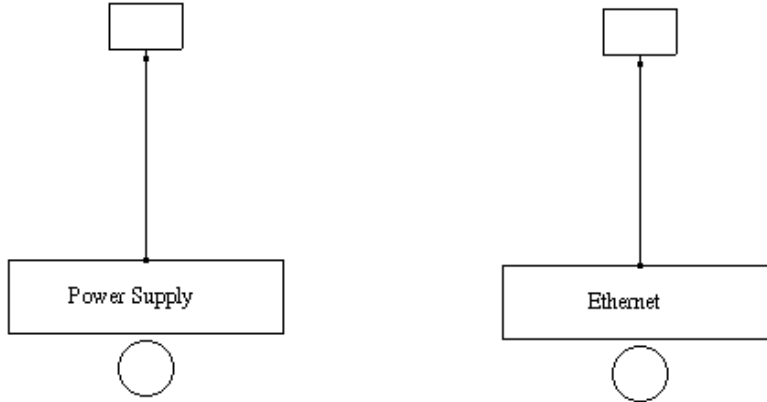


Figure 13: Power Supply and Ethernet Models

the Power Supply and Ethernet models are specified during the association process.

While associating the CS model (CS1) the IN gates for power supply and Ethernet are mapped as PS1 and EC1 where PS1, EC1 are instantiated from models PS and EC. To create a model for CS Type 2, the modeler need not re-specify. The *Copy* feature in HIMAP allows the modeler to copy an already developed model. The modeler then associates the IN gates corresponding to PS and EC which are different in the instantiated model. In this case CS1 is copied and saved as CS2 and the Ethernet is associated with EC2. The PS is not associated again as PS1 remains common between CS1 and CS2. The same procedure is followed to create CS3 and CS4. The instantiations are shown in Figure 14.

Now consider creation of of model for Lab1. It is instantiated from the Lab model and there are 3 CS1 and 4 CS2 in Lab1. Each of the computer systems of type 1 is instantiated from the already developed CS1 model and the instantiations (CS11, CS12, CS13) use PS1 and EC1. Similarly each of the Computer Systems of type 2 is instantiated from CS2 and the instantiations (CS21, CS22, CS23, CS24) use PS1 and EC2. The Lab1 model is shown in Figure 15. The Lab1 model is copied using the *Copy* routine and Lab2, Lab3, Lab4 models are created using the same technique as explained in the Computer System case.

This model can now be solved by solving the sub-models independently and combining the results to get the final solution, or as a single fault tree. The former approach gives an optimistic view of the reliability of the system as the sub-models are treated independently. The latter approach gives an accurate solution by maintaining dependencies among the sub-models. HIMAP allows the model to be solved in both ways.

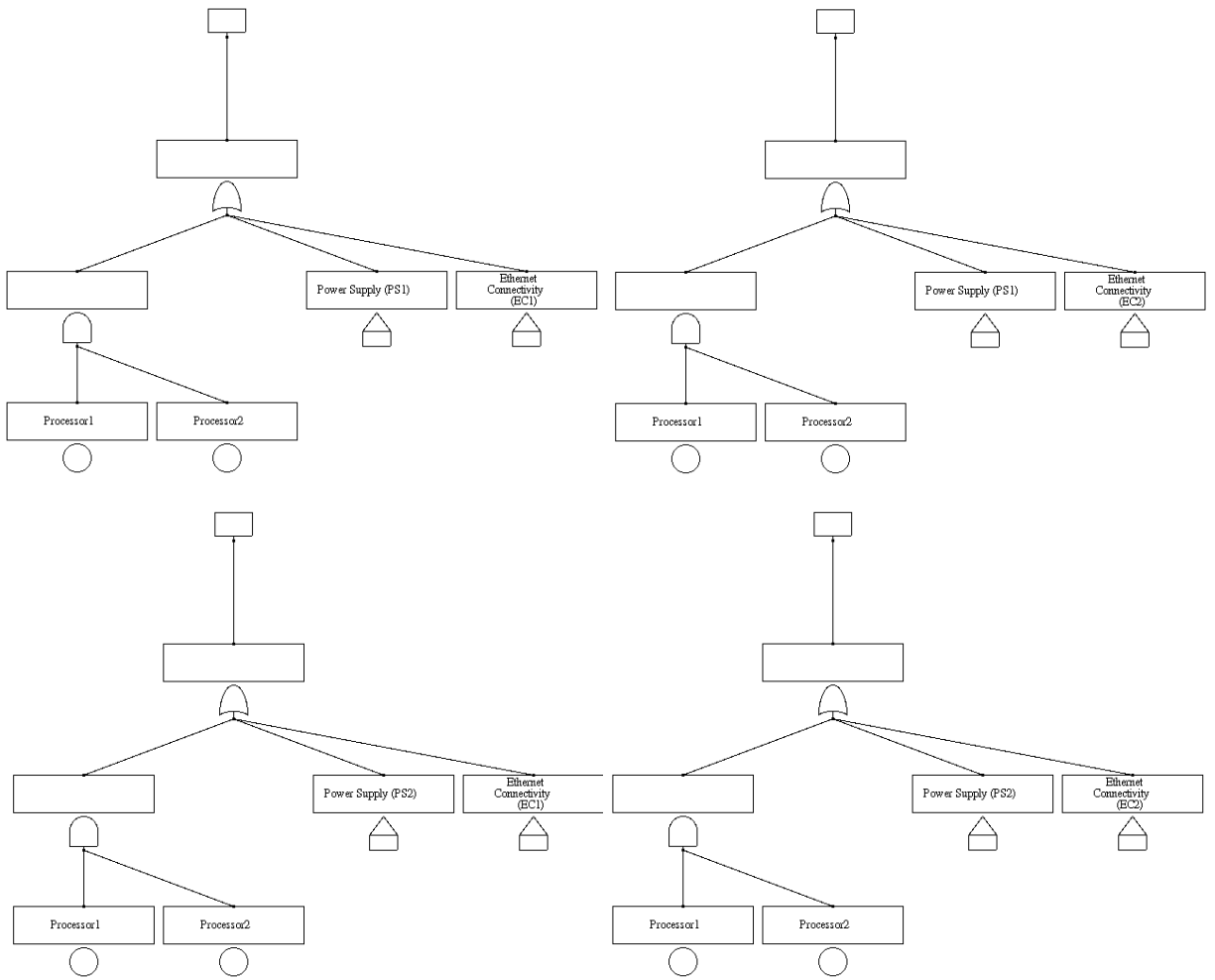


Figure 14: Instantiations from the Computer System Model using *Copy* Routine

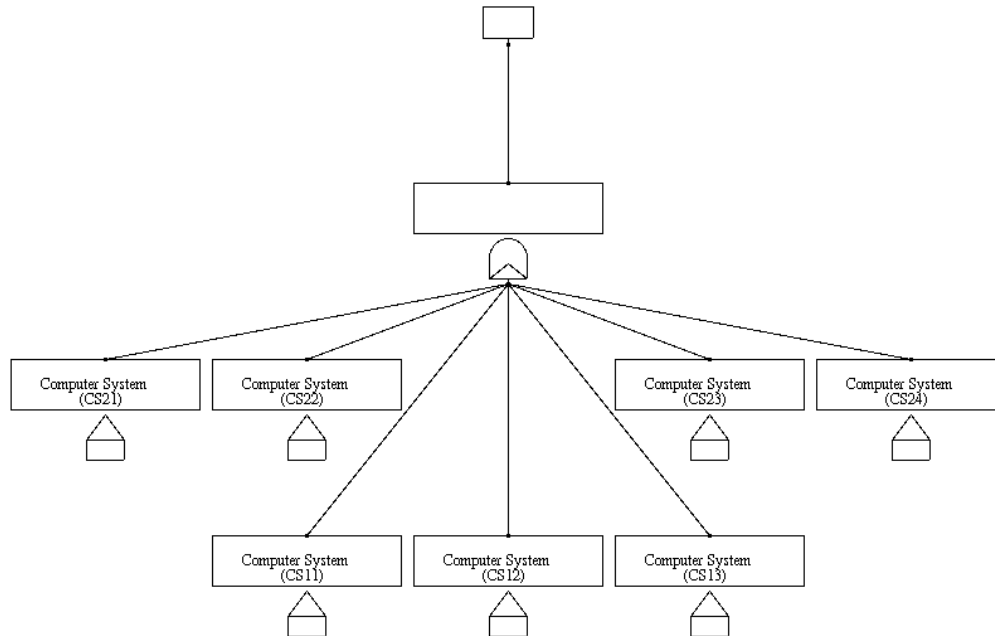


Figure 15: Lab1 Instantiated from the Lab Model

4 Conclusion

In this paper we have presented the modeling features in HIMAP and specification techniques for hierarchical modeling. Complex systems can be modeled in an easy and user-friendly manner, without having to compromise on the accuracy of the model. HIMAP allows systems to be modeled hierarchically, and facilitates individual sub-models to be analyzed separately or as one model. As implemented currently, HIMAP has the following important features:

- Systems can be modeled as a fault tree, fault tree with repairable components, Markov chain or a Petri net.
- Phased Mission Analysis and Schedule maintenance system modeling are incorporated to model realistic scenarios.
- Hierarchical modeling process has been greatly simplified and designers are provided with specification techniques that reduce the amount of work needed to model complex systems.
- An elegant and easy-to-use graphical user interface make HIMAP an ideal tool to model real-world systems. We are currently developing a Windows version of HIMAP for PC users.

References

- [1] J.B. Dugan, K.S. Trivedi, M.K. Smotherman, and R.M. Geist. *The Hybrid Automated Reliability Predictor*. AIAA Journal of Guidance, Control and Dynamics, vol. 9, no. 3, May-June 1986, pp. 319-331.
- [2] A.K. Somani, U.R. Sandadi, A. Gupta, P.C. Leung. *EHARP: Enhanced Hybrid Automated Reliability Predictor*. Tech Report, DPCNL, University of Washington, Seattle, Dec 1993.
- [3] K.S. Trivedi. *SDM: User Manual*. Dept of Electrical Engineering, Duke University.
- [4] J.M. Koren and J. Gaertner. *CAFTA: A fault tree analysis tool designed for PSA*. Proc. Of Probabilistic Safety Assessment and Risk Management: PSA '87, Zurich, vol 2, pp. 588-92, 1987.
- [5] The Boeing Company D6-55312-4. *SETS: System Users Manual*. 1990.
- [6] W.H. Sadnders, W.D. Oball II, M.A. Quereshi, and F.K. Widjanarko. *UltraSAN Version 3: Architecture, Features and Implementation*. Proc AIAA Comp. In aerospace 10 Conf., March 1995.
- [7] G. Ciardo, J. Muppala, and K.S. Trivedi. *SPNP: Stochastic Petri net Package*. Intl. Conf. On Petri nets and performance models, Kyoto, Japan, 1989.
- [8] M. Bouissou, H. Bouhadana, M. Bannelier, and N. Villate. *Knowledge modeling and reliability processing: The FIGARO language and associated tools*. Proc. Of the FTCS-23, pp. 680-685, June 1993.
- [9] A. Anand and A. K. Somani. *Hierarchical Analysis of Fault Trees with Dependencies, Using Decomposition*. in the Proc. of RAMS-1998, Los Angeles, CA, pp , January 1998.
- [10] G. Krishnamurthi, A. Gupta, and A. K. Somani. *The HIMAP Modeling Environment*. in Proc. of PDCS-96, Sept 24-27, 1996, Dijon, France.
- [11] R.M. Sinnamon and J.D. Andrews. *Fault Tree Analysis and Binary Decision Diagrams*. Proc. RAMS, pp.215-220, Jan 1996.
- [12] A.K. Somani, J. Ritcey, and S.Au. *Computationally efficient Phased-Mission reliability analysis for systems with variable configurations*. IEEE Trans on Reliability, Vol 41, No 4, pp. 504-511, Dec 1992.
- [13] G. Krishnamurthi, A. Gupta, and A.K. Somani. *The HIMAP Modeling Environment*. DCL Tech Report, Iowa State University, Ames, Mar 1998.
- [14] T. Sakaguchi. *Development of the hierarchical Stochastic Reward Net Solver Package*. Master's Thesis, Dept. of Electrical Engineering, University of Washington, Seattle, 1997.

- [15] A.K. Somani, Samir Palnitkar, and Tilak Sharma. *Reliability Modeling of Systems with Latent Failures Using Markov Chains*. Proc. on Annual Reliability and Maintainability Symposium 1993.