Dynamic Fault Tree Analysis for Software-Based Systems

Joanne Bechta Dugan
Professor of Electrical & Computer Engineering
University of Virginia
jbd@Virginia.edu

Dynamic Fault Tree Analysis

- Fault tree analysis (FTA) is a widely accepted methodology for reliability analysis.
- Recent advances in FTA have allowed the analysis of FTC systems which are characterized by:
  - high levels of redundancy; large numbers of components
  - complex redundancy management
  - spares (cold, warm, pooled)
  - functional and sequence dependencies
  - hardware and software
  - imperfect coverage

Static Fault Trees

- Combinatorial model (models combinations of events)
  - AND gates
  - OR gates
  - K-of-M gates
- New approach for solution: BDD (Binary Decision Diagrams)
- Advantages:
  - Exact analysis without cutsets
  - Can include repeated events
  - can include coverage modeling
  - Fast solution for very large models
- Disadvantage:
  - Static model: cannot include sequence dependencies

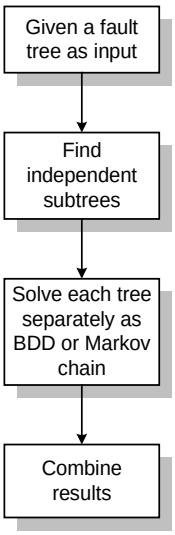
Dynamic Fault Trees

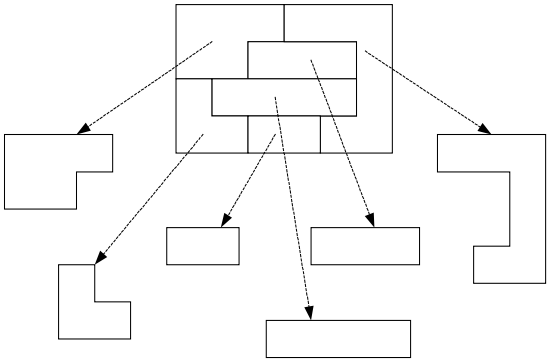
- Include special constructs for modeling sequence dependencies
  - functional dependencies
  - hot, warm and cold spares
  - priority-AND
  - sequence enforcing
- Solution: convert to Markov chain
- Advantages:
  - easier to use fault tree than Markov model directly
  - can model dynamic redundancy, shared pools of spares, etc
- .Disadvantage:
  - state space explosion -- worst case exponential in number of basic events

Coverage Modeling

- Adaptive (computer-based) systems can exhibit multiple failure modes
- *Covered* (benign) failure can be handled automatically
  - error is detected and located
  - switch in spare or bypass faulty component
  - system can continue operation without manual intervention
- *Uncovered* failure is globally malicious
  - undetected error escapes from embedded system
  - faulted component cannot be disabled
  - malicious behavior confuses recovery procedures
- System dependability measures are very sensitive to coverage
- Good techniques exist for incorporating coverage into static and dynamic fault trees.

Modular approach to fault tree analysis

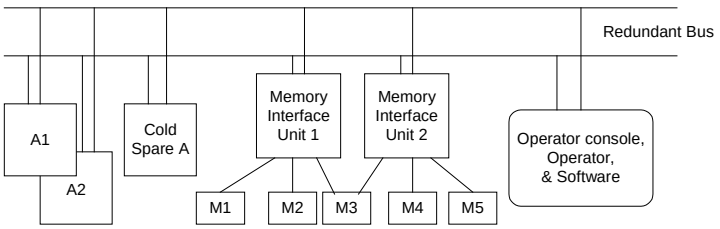




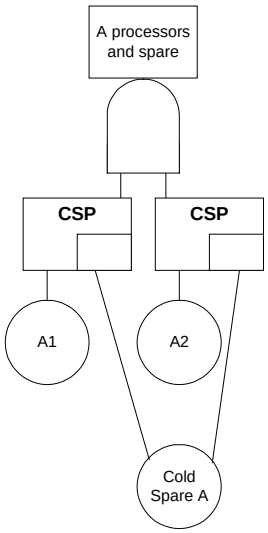
The best of both worlds

- Modularization combines the best of Boolean (BDD) and Markov approaches
  - Use fast and efficient BDD approach for static modules
  - Build Markov chain automatically when needed for dynamic behavior
  - divide-and-conquer helps avoid models which are too large to solve
- Different solution methods can aid in testing and validation
- Modularization allows alternative solution approaches

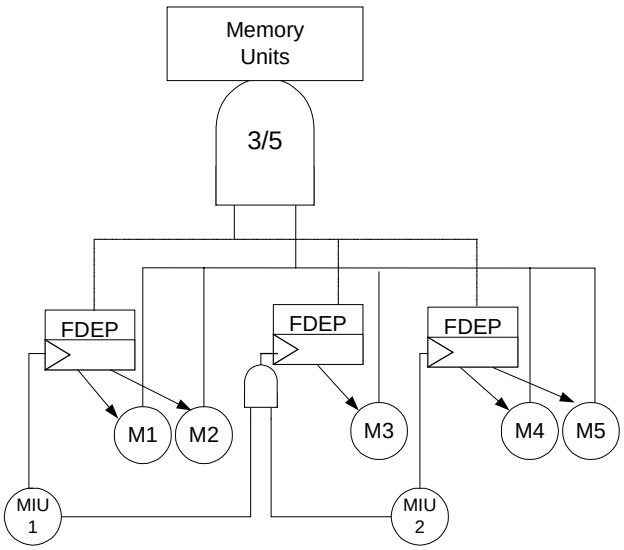
HECS: Hypothetical Example Computer System



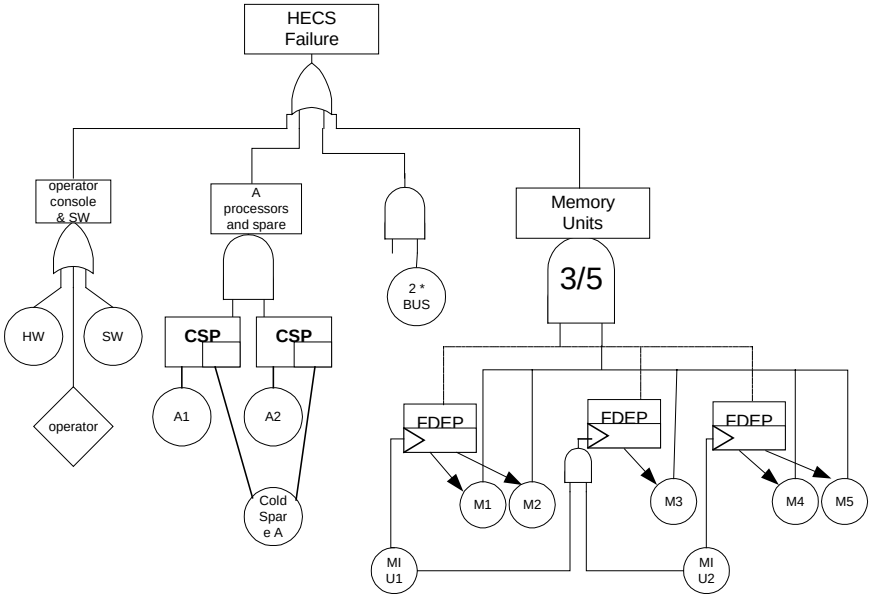
Modeling the processors



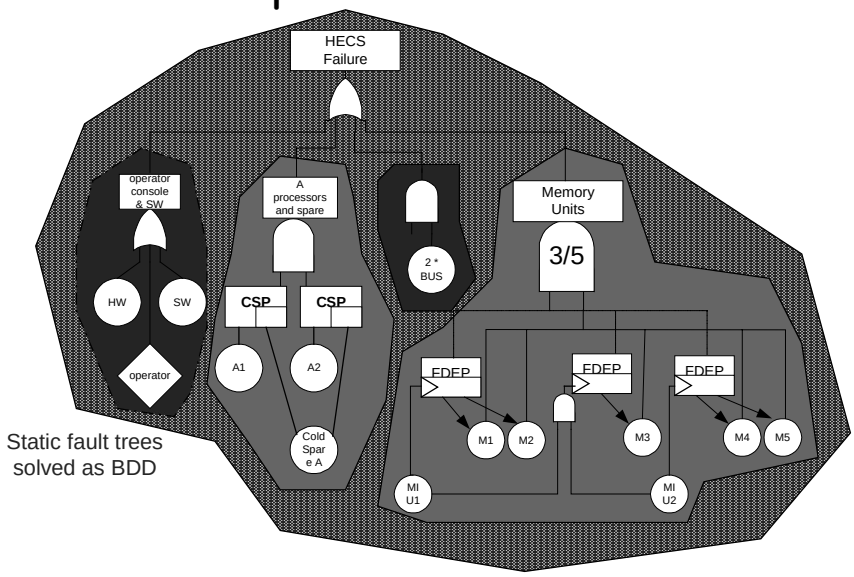
Modeling the memories



Full dynamic fault tree model



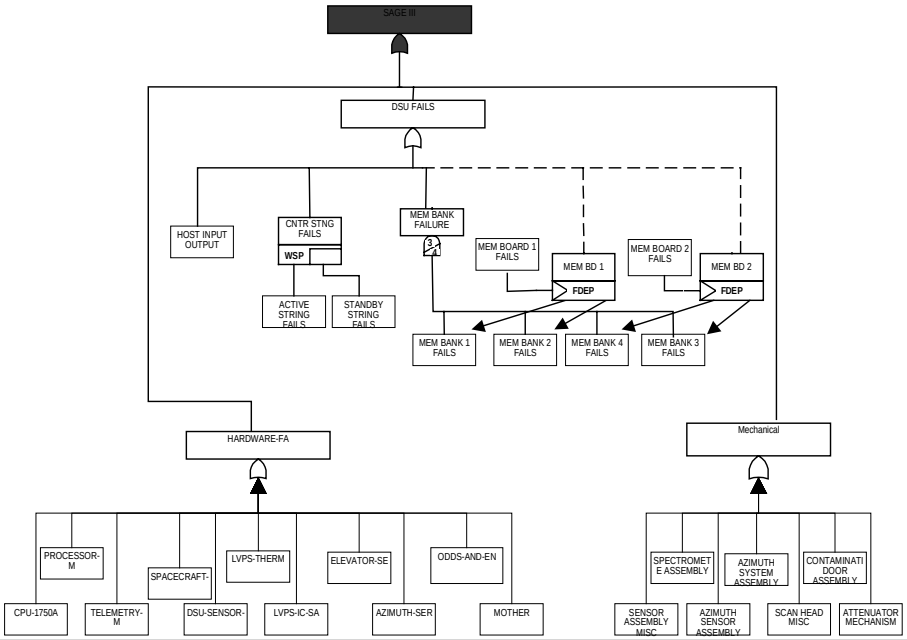
Independent subtrees



Static fault trees solved as BDD

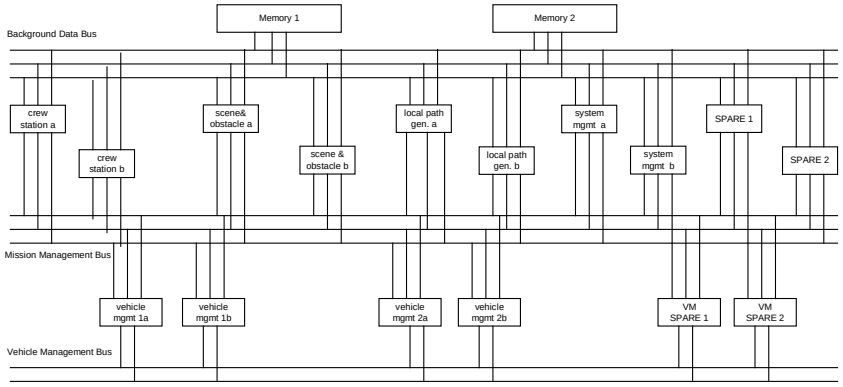
Dynamic Fault trees solved as Markov chains

SAGE III Stratospheric Aerosol & Gas Experiment)



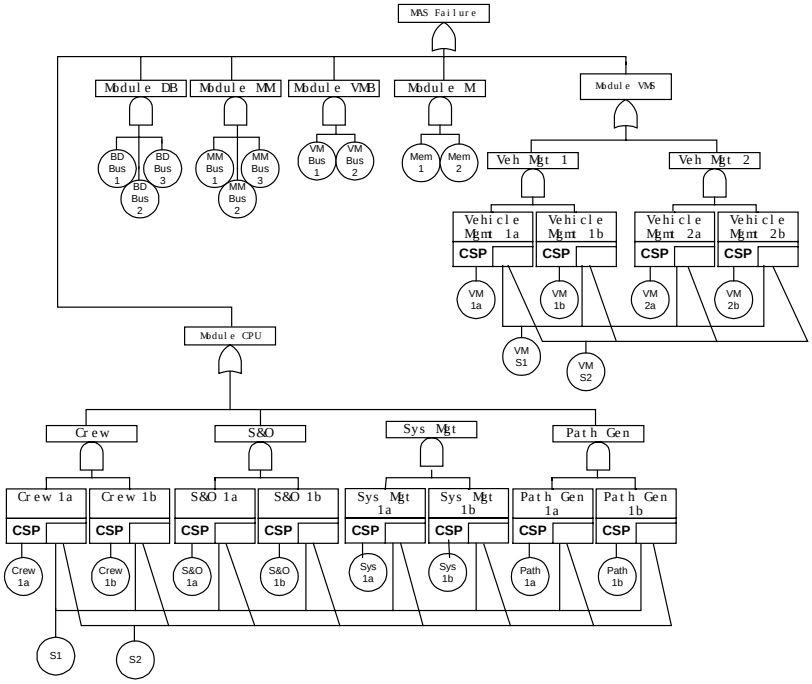
Mission Avionics System Example

- The example mission avionics system provides several key software-based functions on a fault tolerant hardware architecture.
- Each processing element is really a pair of processors whose results are compared; disagreements result in reconfiguration to a spare.
- There is a hot-spare for each processing element, as well as a pool of cold spares which are shared by several processors.



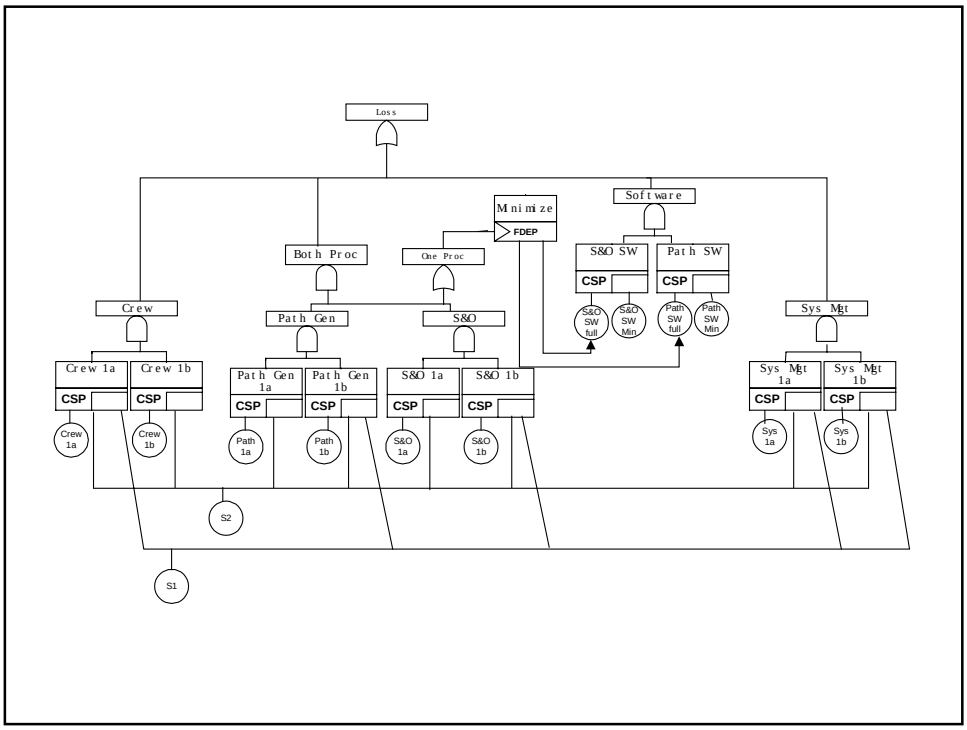
System Requirements

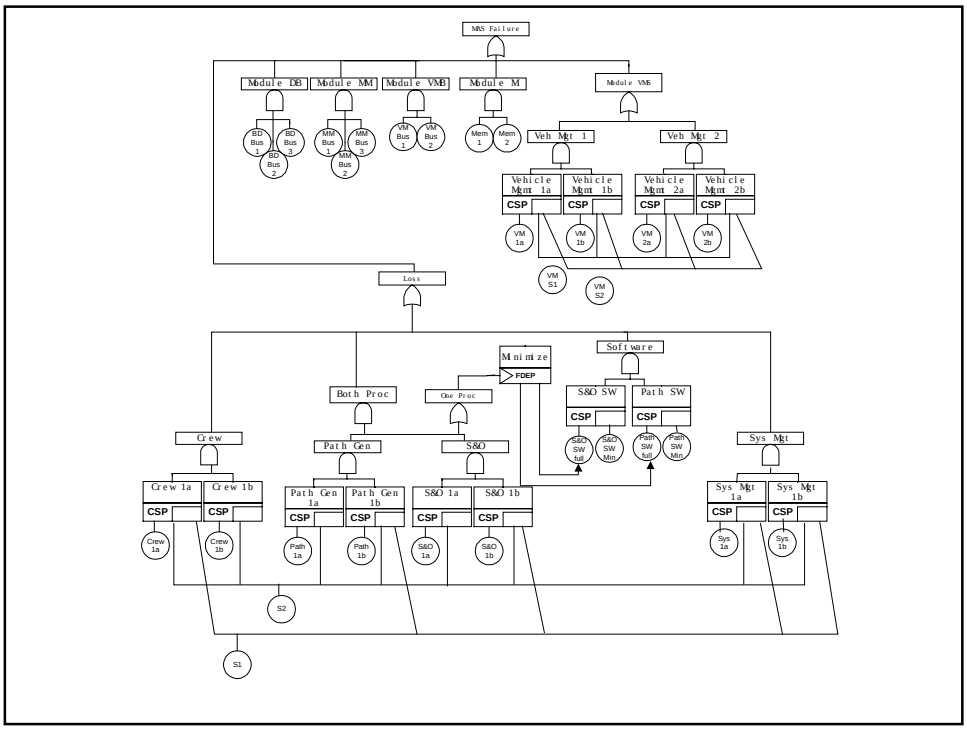
- The success/failure of the system is driven by the need to provide certain software functionality
  - crew station management
  - scene & obstacle processing
  - local path generation
  - system management functions
  - vehicle management



Redundant software architecture

- Next consider the path generation (PathGen) and scene&obstacle (S&O) functions
- Each function needs a single processing element to provide full functionality
- There is also a reduced version of each function that can provide minimum functionality (PathGenMin & S&Omin).
- In the event of a detected software fault in PathGen, the system can switch to PathGenMin (similarly for S&O).
- Further, if there are no longer 2 full processing elements available, the system will switch to PathGenMin and S&Omin running on a single processing element.


Loss
Minimize
FDEP
Software
Both Proc
One Proc
S&O SW
Path SW
CSP
S&O SW full
S&O SW Min
Path SW full
Path SW Min
Crew
Path Gen
S&O
Sys Mgt
Crew 1a
Crew 1b
Path Gen 1a
Path Gen 1b
S&O 1a
S&O 1b
Sys Mgt 1a
Sys Mgt 1b
Crew 1a
Crew 1b
Path 1a
Path 1b
S&O 1a
S&O 1b
Sys 1a
Sys 1b
S2
S1


MNS Failure
Module DB
Module MM
Module VMB
Module M
Module VMS
BD Bus 1
BD Bus 3
BD Bus 2
MM Bus 1
MM Bus 3
MM Bus 2
VM Bus 1
VM Bus 2
Mem 1
Mem 2
Veh Mgt 1
Veh Mgt 2
Vehicle Mgmt 1a
Vehicle Mgmt 1b
Vehicle Mgmt 2a
Vehicle Mgmt 2b
CSP
VM 1a
VM 1b
VM 2a
VM 2b
VM S1
VM S2
Loss
Minimize
FDEP
Software
Both Proc
One Proc
S&O SW
Path SW
CSP
S&O SW full
S&O SW Min
Path SW full
Path SW Min
Crew
Path Gen
S&O
Sys Mgt
Crew 1a
Crew 1b
Path Gen 1a
Path Gen 1b
S&O 1a
S&O 1b
Sys Mgt 1a
Sys Mgt 1b
Crew 1a
Crew 1b
Path 1a
Path 1b
S&O 1a
S&O 1b
Sys 1a
Sys 1b
S2
S1

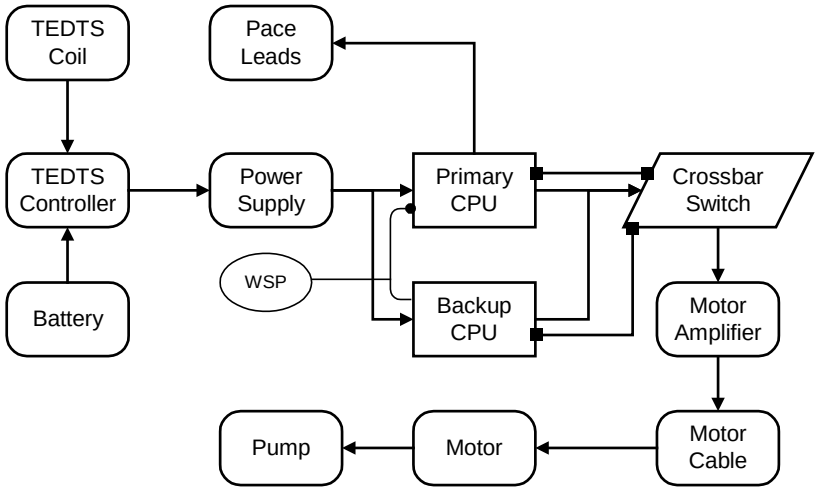
Sensitivity analysis

- Reliability analysis tells only part of the story
- "What are the weak points in the system?"
- "How do results change with changing input parameters?"
- "What is most cost effective way to improve reliability?"
- How much uncertainty in result?
- These questions require sensitivity analysis of reliability analysis results

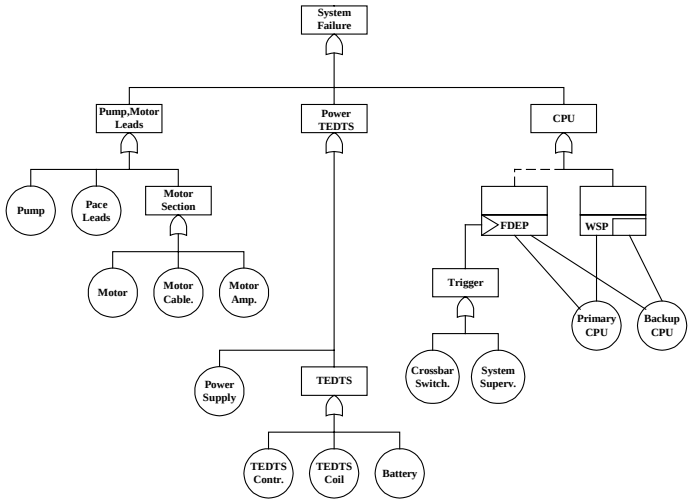
Modular approach to sensitivity

- Sensitivity analysis (also called importance analysis) can use partial derivative
- Sensitivity results from different submodels can be easily combined using "chain-rule"
- Sensitivity analysis for BDD is almost free while calculating reliability
- Sensitivity analysis for Markov chain is more troublesome but we've developed an interesting, efficient approximation

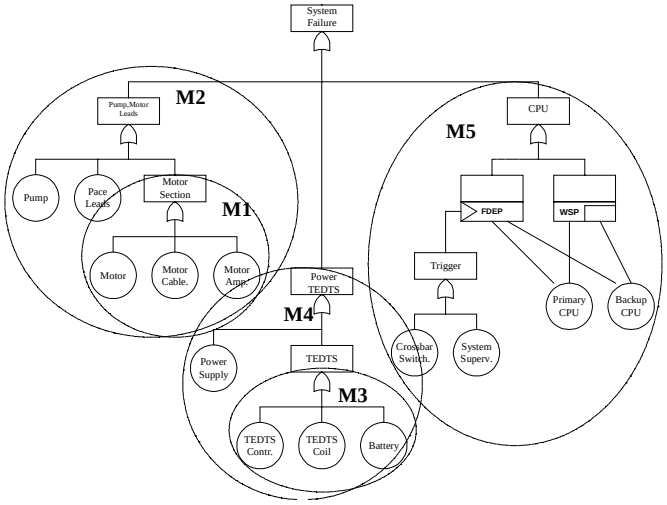
Example - Cardiac Assist System



Fault Tree Model for CAS



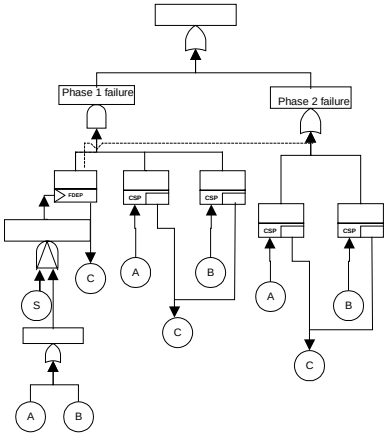
Modularization & sensitivity analysis



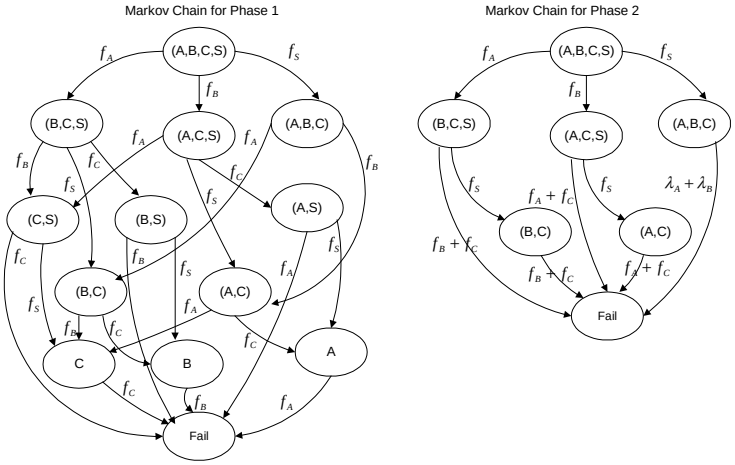
Phased Mission Systems

- System is subject to several consecutive (non-overlapping in time) phases of operation
- Reliability requirements can be different in each phase
- Failure parameters may be different in each phase
- Some components may not be used in every phase, but may nonetheless fail
- System must succeed in each phase
- Components are dependent across phases

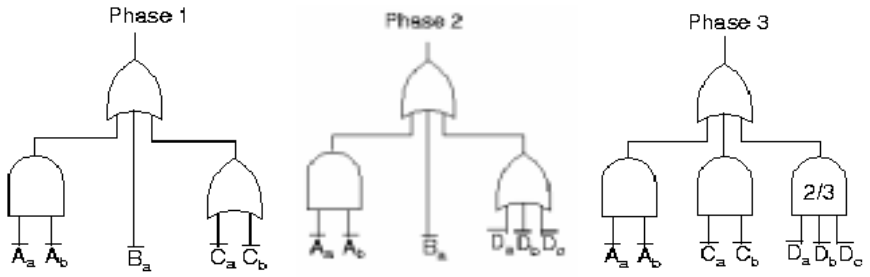
Dynamic Phased Mission System



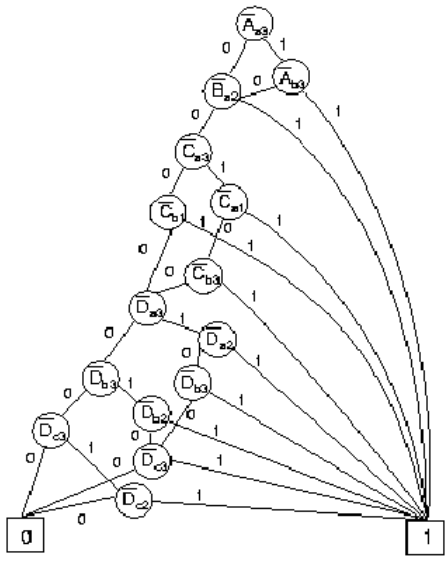
Solve as Markov model



Static Phased Mission System



Solve as BDD (using Phase Algebra)



Benchmarks for Evaluation

- We are developing a set of benchmarks for evaluation of Galileo
- Includes "scrubbed" versions of real systems from industry
- Determination of "correct" result by alternative means
- Paper accepted for publication & presentation at RAMS (Reliability and Maintainability Symposium) January 2001

Unique Features in our Methodology

- Modularization allows solution of large systems
- Combines Markov analysis with BDD analysis automatically
- Automatic generation of Markov model or BDD
- Coverage modeling for computer-based systems
- Sensitivity analysis for static and dynamic models
- Phased mission analysis
- Exact results (no hidden approximations)
- Benchmarks to show applicability and correctness