REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (*LEAVE BLANK*)	2. REPORT DATE 27 October 1995	3. REPORT TYPE AND DATES COVERED Professional Paper

4. TITLE AND SUBTITLE Flight Control Computer Development Through Application of Software Safety Technology	5. FUNDING NUMBERS
6. AUTHOR(S) Janet A. Gill	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Commander Naval Air Warfare Center Aircraft Division 22541 Millstone Road Patuxent River, Maryland 20670-5304	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Air Systems Command Department of the Navy 1421 Jefferson Davis Highway Arlington, VA 22243	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release, distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This presentation provides an overview of Software Safety Assurance, and includes a practical example as applied to the Vehicle Management System as a part of the V-22 Engineering Manufacturing Development Program. The following topics are addressed:

What is Software Safety Assurance?
Software Safety Assurance Mission
Software Safety Assurance Program
Analysis Techniques

14. SUBJECT TERMS V-22, Engineering Manufacturing Development Program, Vehicle Management System	15. NUMBER OF PAGES 34 16. PRICE CODE		
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT N/A	20. LIMITATION OF ABSTRACT N/A

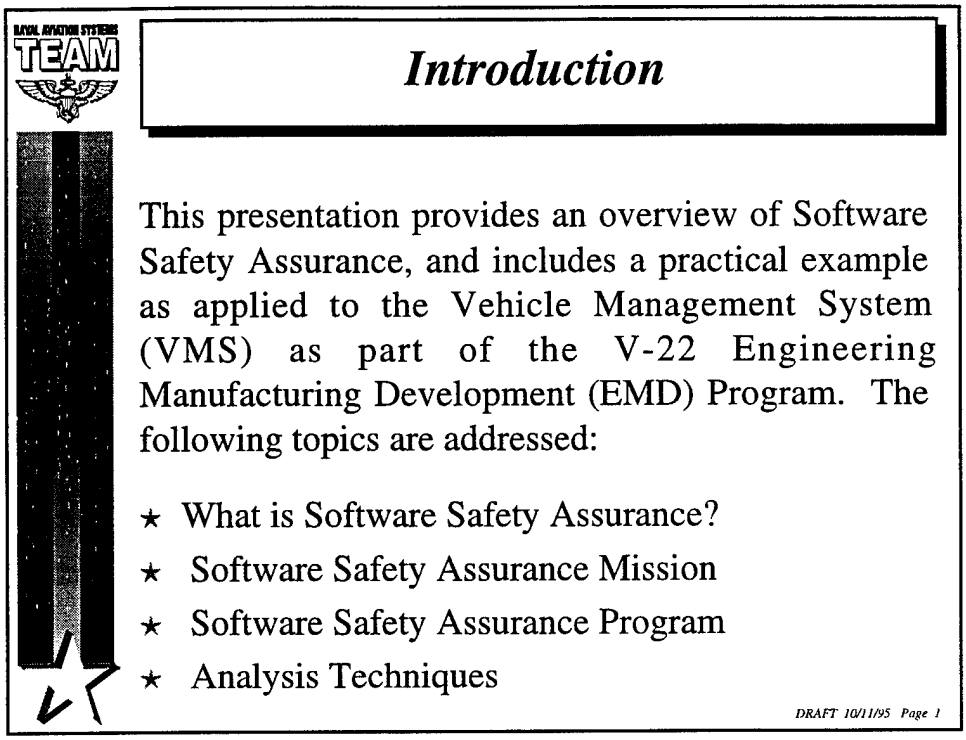
Enc(8)

Flight Control Computer Development Through Application of Software Safety Technology

Presented by

Janet A. Gill
Software Safety Assurance
Competency 4.5.7.4
(301) 342-3133 ext. 230
October 26, 1995

Enclosure (1)



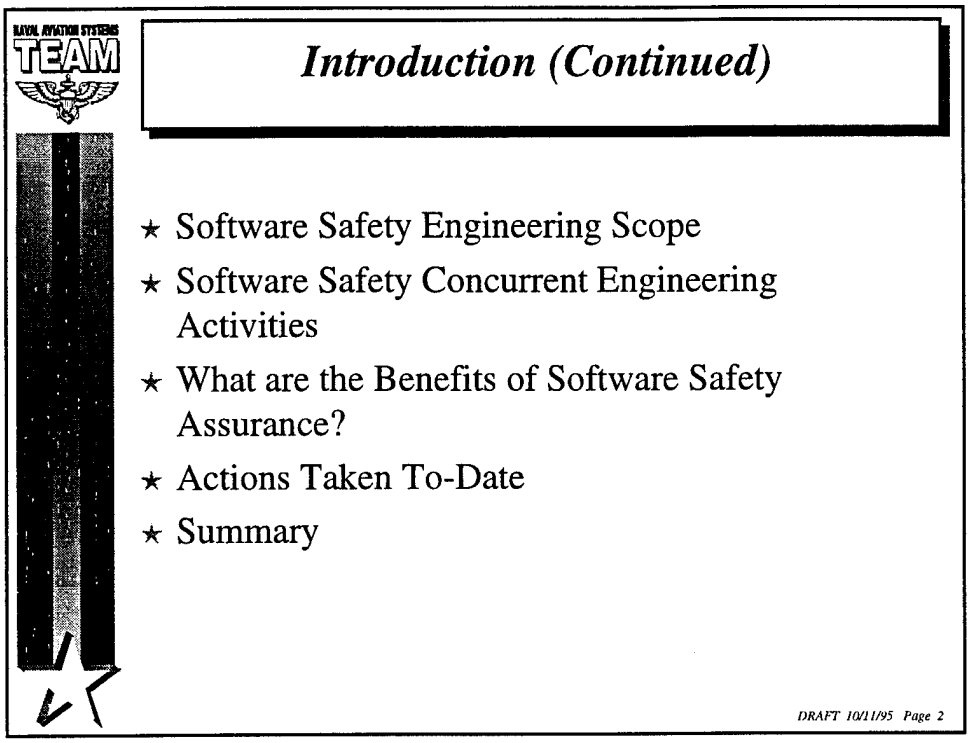
This presentation provides an overview of Software Safety Assurance, and includes a practical example as applied to the Vehicle Management System (VMS) as part of the V-22 Engineering Manufacturing Development (EMD) Program. The following topics are addressed:

★ What is Software Safety Assurance?

★ Software Safety Assurance Mission

★ Software Safety Assurance Program

★ Analysis Techniques



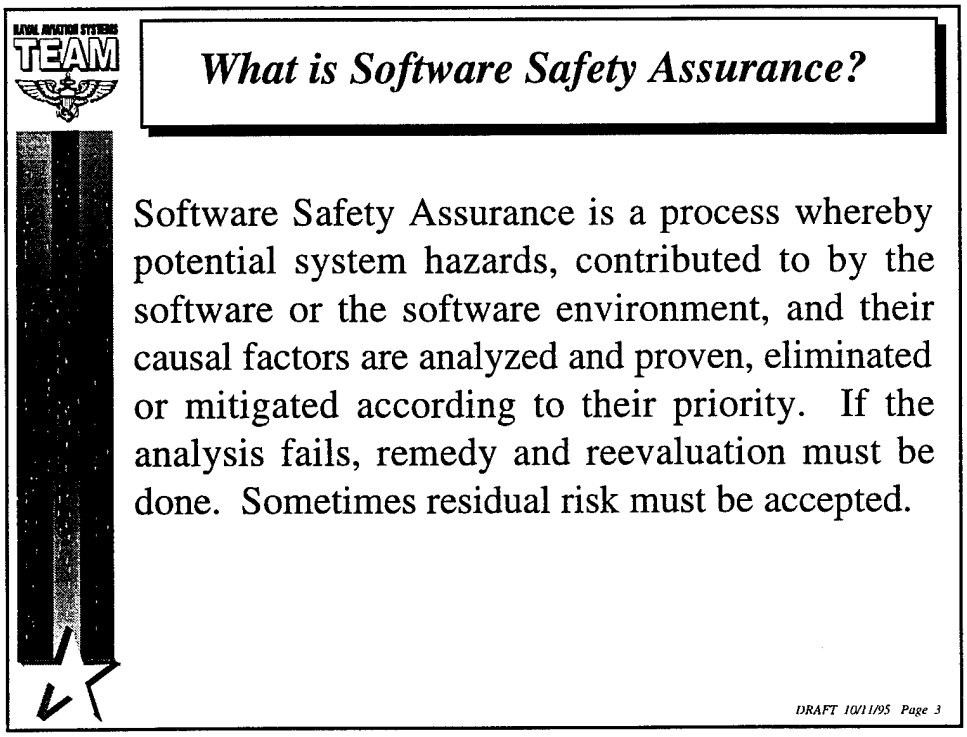
★ Software Safety Engineering Scope

★ Software Safety Concurrent Engineering Activities

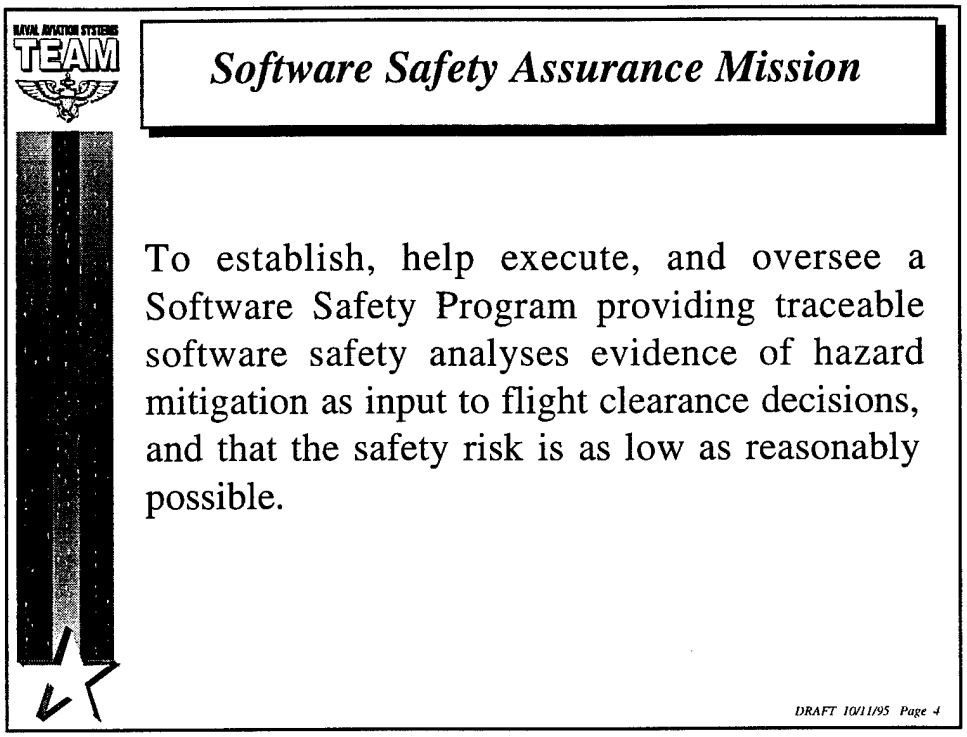
★ What are the Benefits of Software Safety Assurance?

★ Actions Taken To-Date

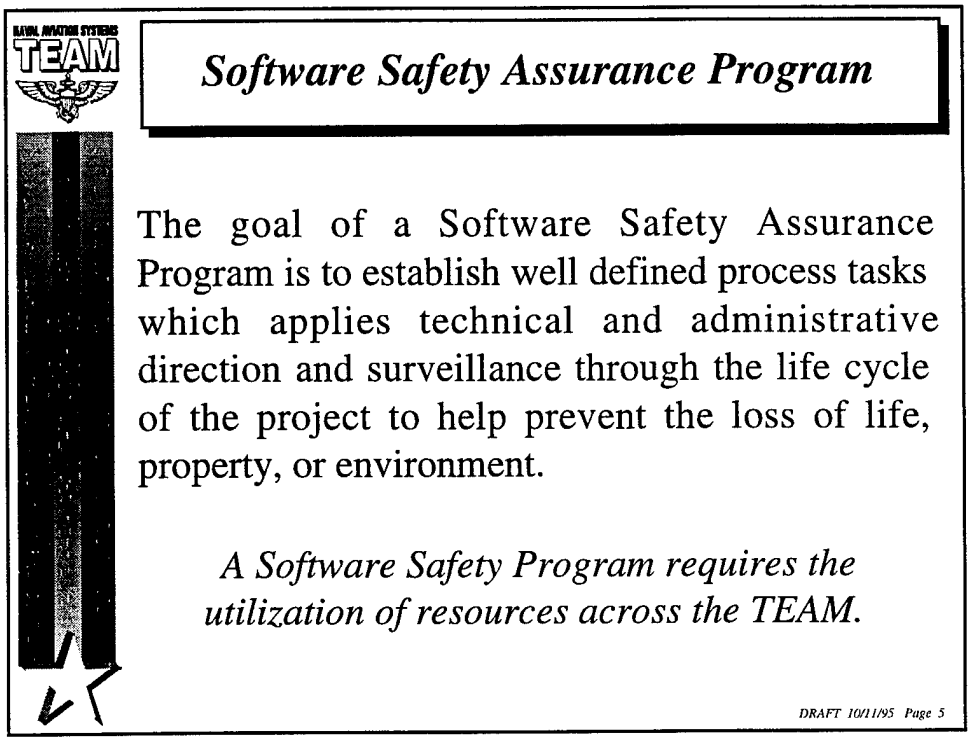
★ Summary



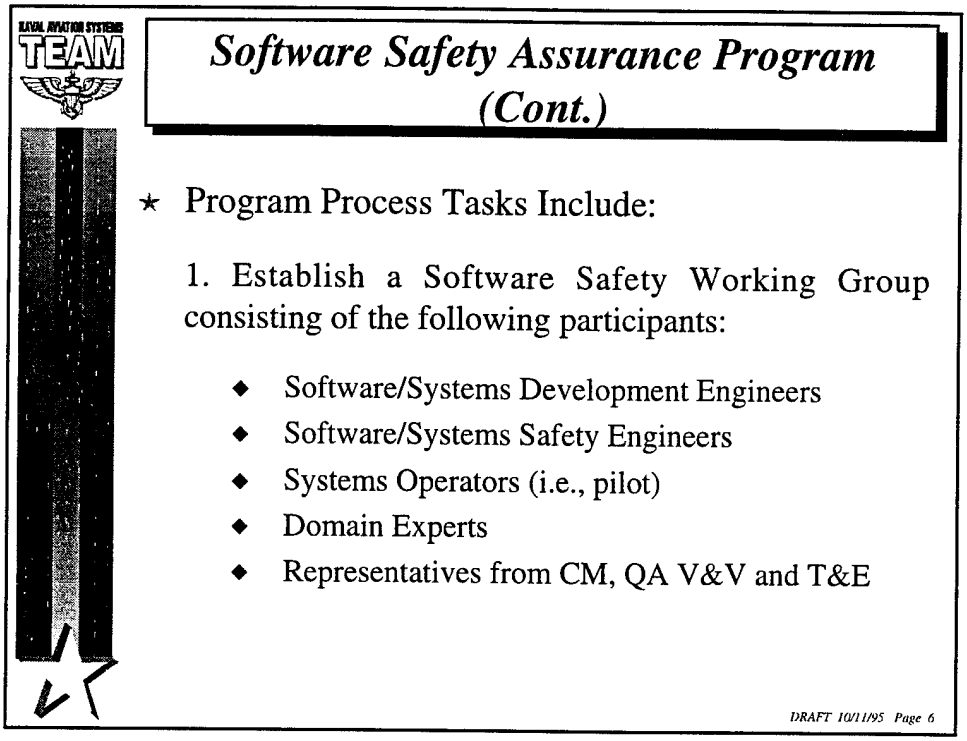
Software Safety Assurance is a process whereby potential system hazards, contributed to by the software or the software environment, and their causal factors are analyzed and proven, eliminated or mitigated according to their priority. If the analysis fails, remedy and reevaluation must be done. Sometimes residual risk must be accepted.



To establish, help execute, and oversee a Software Safety Program providing traceable software safety analyses evidence of hazard mitigation as input to flight clearance decisions, and that the safety risk is as low as reasonably possible.

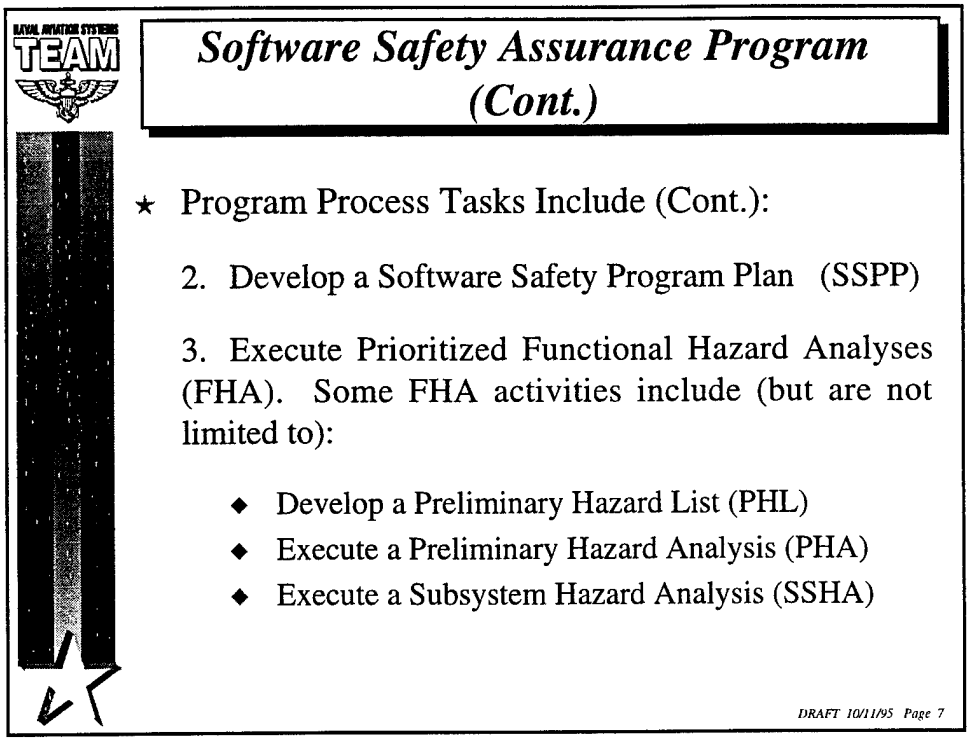


The goal of a Software Safety Assurance Program is to establish well defined process tasks which applies technical and administrative direction and surveillance through the life cycle of the project to help prevent the loss of life, property, or environment. A Software Safety Program requires the utilization of resources across the TEAM.



Establish a Software Safety Working Group consisting of the following participants:

- ★ Software/Systems Development Engineers
- ★ Software/Systems Safety Engineers
- ★ Systems Operators (i.e., pilot)
- ★ Domain Experts
- ★ Representatives from CM, QA V&V and T&E



Functional Hazard Analyses is Defined as - The identification, evaluation, and management of the system's potential functional hazards that may be contributed to by software requirements/design algorithms, or lack thereof, (i.e., premature stores release).

PHL (Preliminary Hazards Lists) consists of:

★ Holding a brainstorming session to determine all potential hazards to be analyzed and,

★ Working the list for appropriateness of each potential hazard.

PHA (Preliminary Hazard Analysis) consists of:

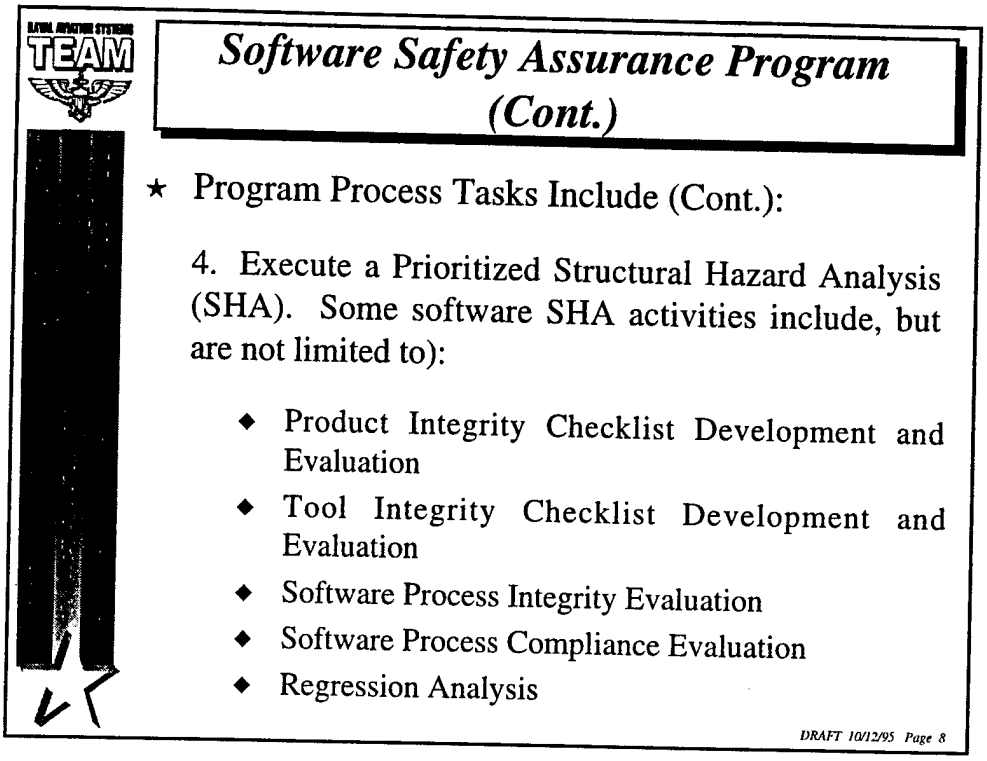
★ Categorizing and Prioritize the list according to System Safety MIL-STD-882C and,

★ Having management determine the list of potential hazards to be analyzed.

SSHA (SubSystem Hazard Analysis consists of:

★ Executing first-level design Hazard Analysis and,

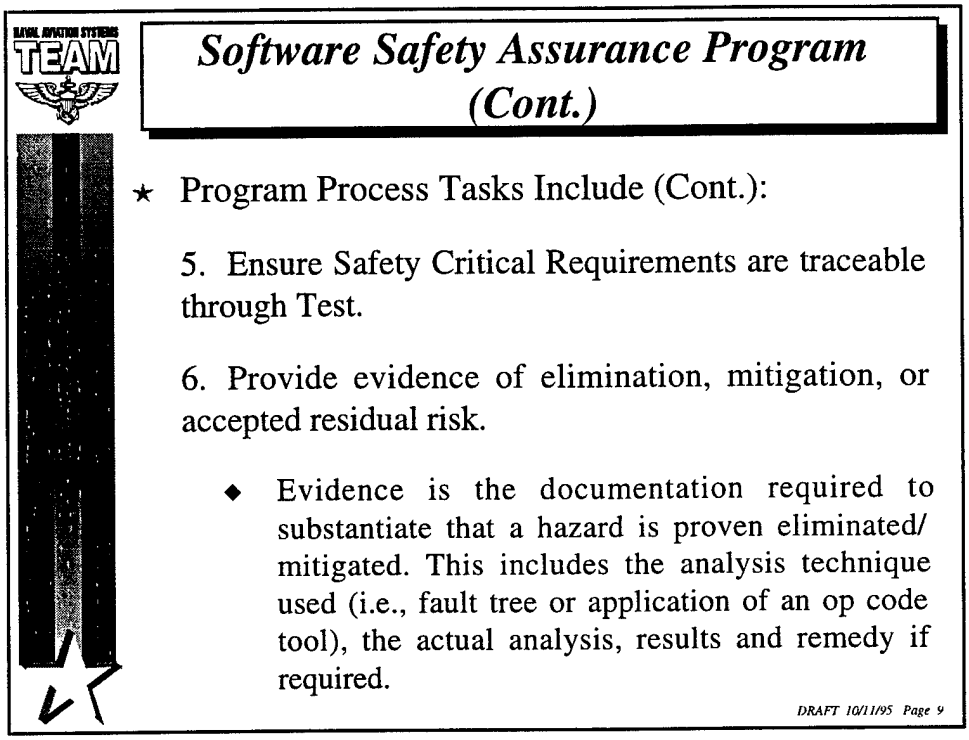
★ Execute detailed level design Hazard Analysis.



Structural Hazard Analysis is Defined as - The on-going analysis of development or maintenance processes and products during the entire life cycle with respect to software engineering practices/ design features of a system containing safety critical software, (i.e., no restricted op codes).

Execute a Prioritized Structural Hazard Analysis (SHA). Some software SHA activities include, but are not limited to):

★ Product Integrity Checklist Development and Evaluation
★ Tool Integrity Checklist Development and Evaluation
★ Software Process Integrity Evaluation
★ Software Process Compliance Evaluation
★ Regression Analysis



If a hazard can be reached, it must be mitigated and reevaluated.

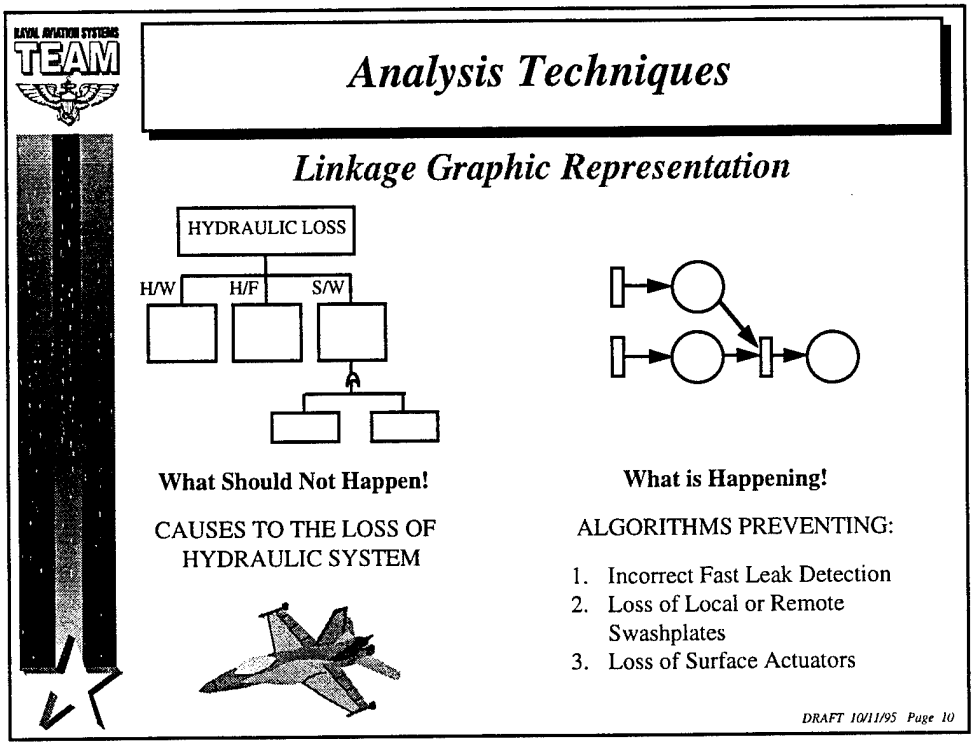
Ensure Safety Critical Requirements are traceable through Test.

Provide evidence of elimination, mitigation, or accepted residual risk.

★ Evidence is the documentation required to substantiate that a hazard is proven eliminated/mitigated. This includes the analysis technique used (i.e., fault tree or application of an op code tool), the actual analysis, results and remedy if required.

Many Structural Hazards are evaluated in a good software development program. Collecting the previously documented evidence may be all that is necessary.

Many of the elements of these analyses are already considerations in the software life cycle development. We do not want to duplicate effort; however, there is a need to accumulate an audit trail of evidence that we are at the lowest possible risk and that we cannot reach the hazards identified.

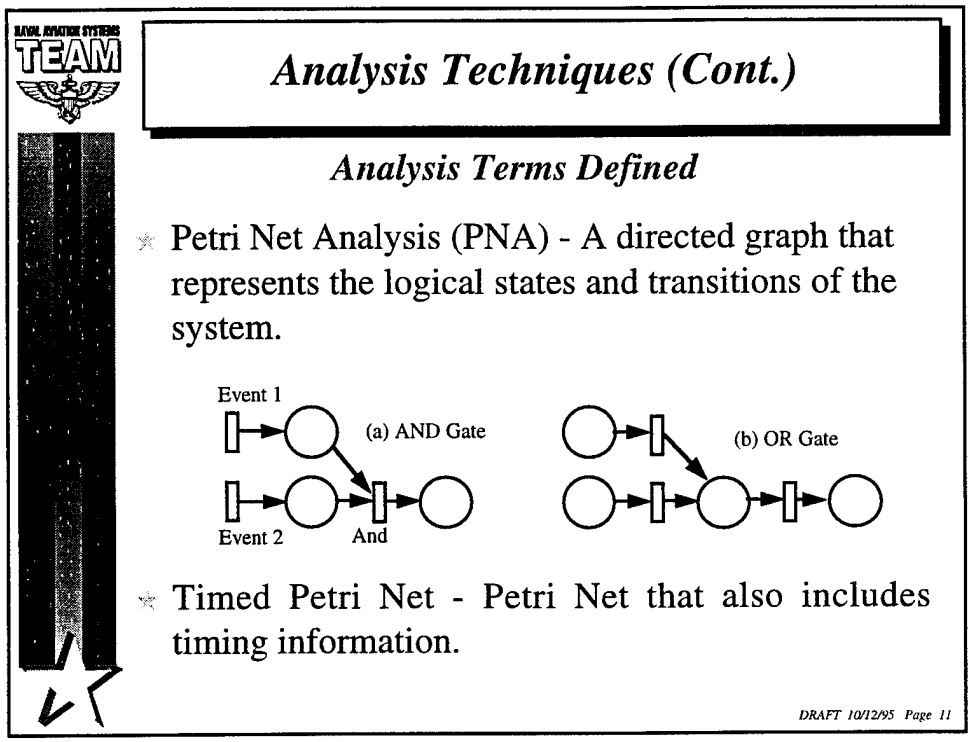


The key to the analysis concept is to determine all of the causes to the hazard (What Should Not Happen!) and compare that to (what is happening). If you can link a path to the root node of the fault tree you have a problem that needs corrected.

The example identifies a possible algorithm problem.

ALGORITHMS PREVENTING:

- ★ Incorrect Fast Leak Detection.
- ★ Loss of Local or Remote Swashplates.
- ★ Loss of Surface Actuators must be in place to ensure these causes to the loss of an hydraulic system are not occurring.

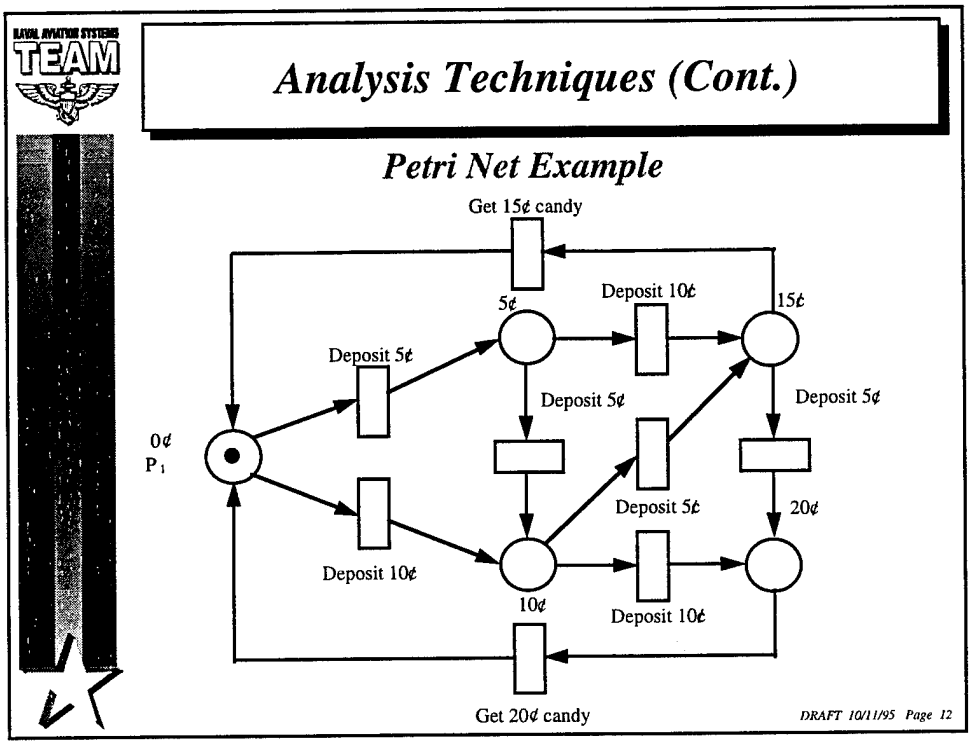


★ Petri Net Analysis (PNA) is a directed graph that represents the logical states and transitions of the system.

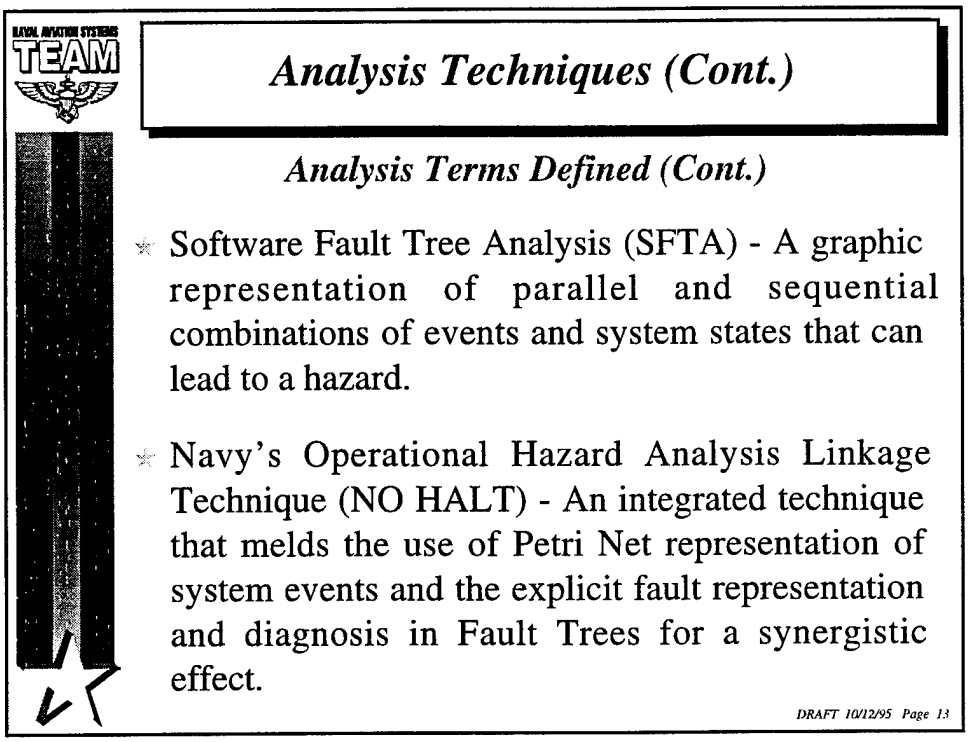
★ Timed Petri Net is a Petri Net that also includes timing information.

★ These are the parts of a petri net:

- ◆ Places/states
- ◆ Transitions
- ◆ Directional flow arrows

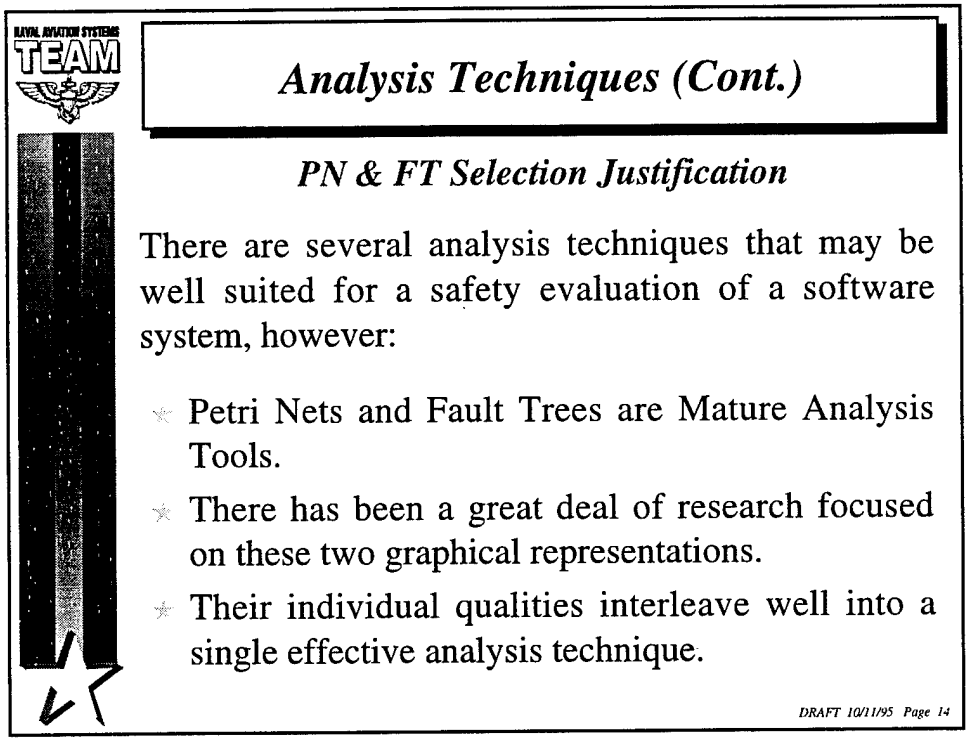


This is a Petri net of a candy machine. The dark dot signifies an initial state. As the activity flows, the dot will move throughout the system. Note there isn't any money in the slot, therefore, the state is 0 until the transition of depositing a nickel in the machine changes the state to 5 cents. Another transition of depositing a nickel will change the state to 10 cents and another to 15 cents. Then a decision must be made to pull the lever, getting a 15 cent candy bar and setting the state back to 0, or depositing another nickel and so on.



★ Software Fault Tree Analysis (SFTA) is a graphic representation of parallel and sequential combinations of events and system states that can lead to a hazard.

★ Navy's Operational Hazard Analysis Linkage Technique (NO HALT) is an integrated technique that melds the use of Petri Net representation of system events and the explicit fault representation and diagnosis in Fault Trees for a synergistic effect. I have an entire brief describing this technique for anyone that is interested.

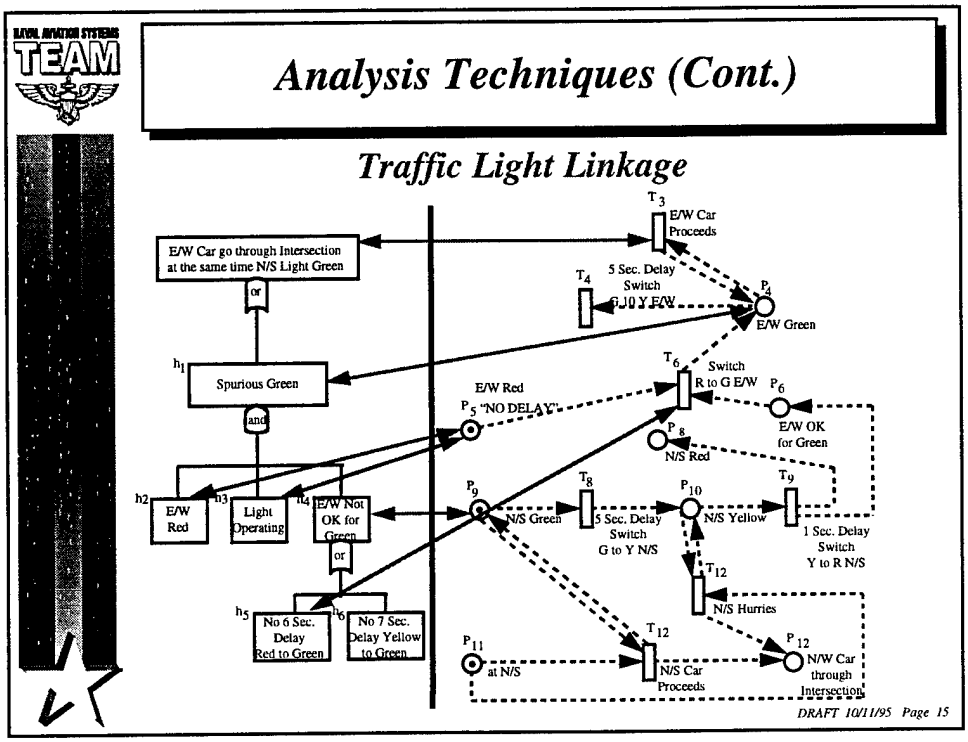


There are several analysis techniques that may be well suited for a safety evaluation of a software system, however:

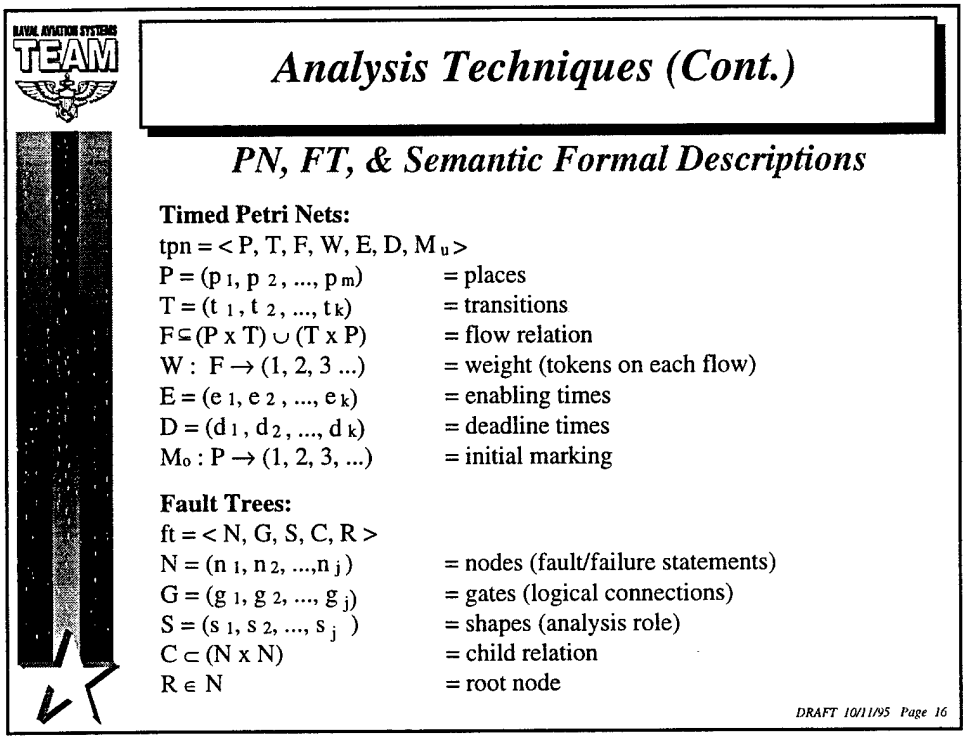
★ Petri Nets and Fault Trees are Mature Analysis Tools.

★ There has been a great deal of research focused on these two graphical representations.

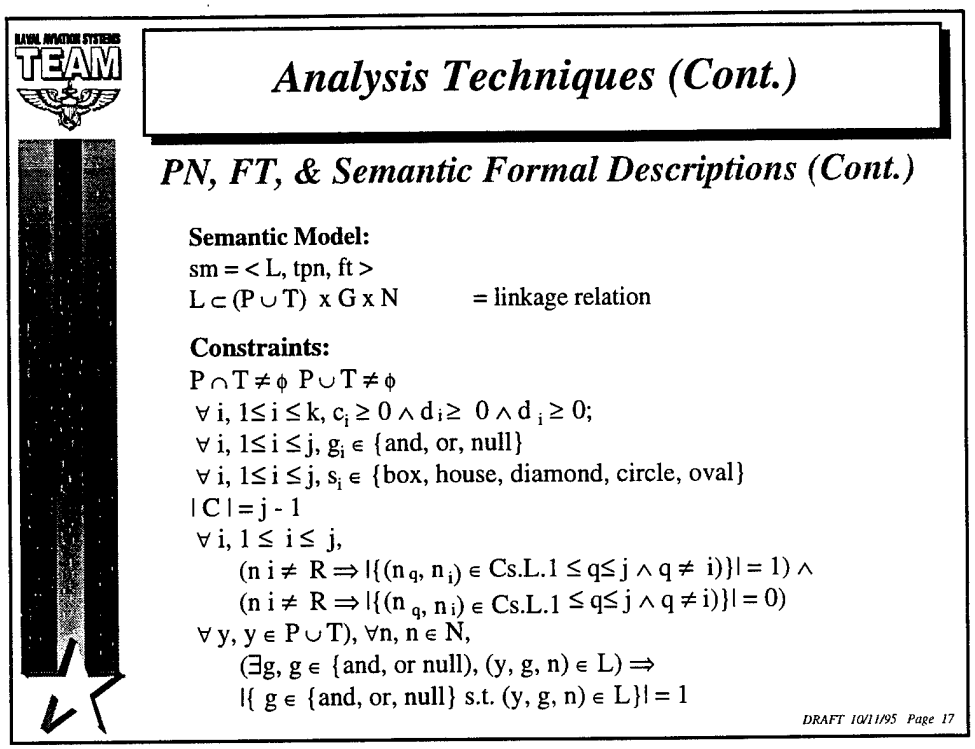
★ Their individual qualities interleave well into a single effective analysis technique.



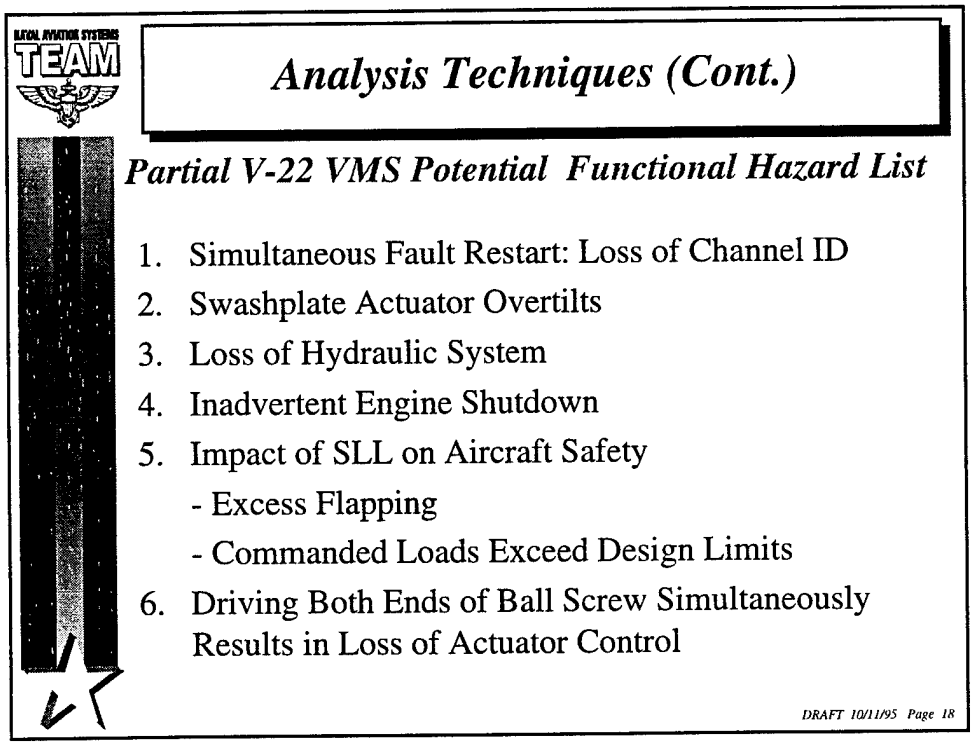
This is the linking of a fault tree to a Petri net in an academic example of a traffic light sequence. We should all be familiar with this. It was obvious that there was no second delay going from red to green on the east/west traffic light when we applied this technique. Again, when you can reach the root of the hazard through this linkage, you have a problem.



Here is the rigorous semantic model that reflects this linkage for the mathematicians. Even though graphics is the preferred representation, this textual representation will give an equivalency.

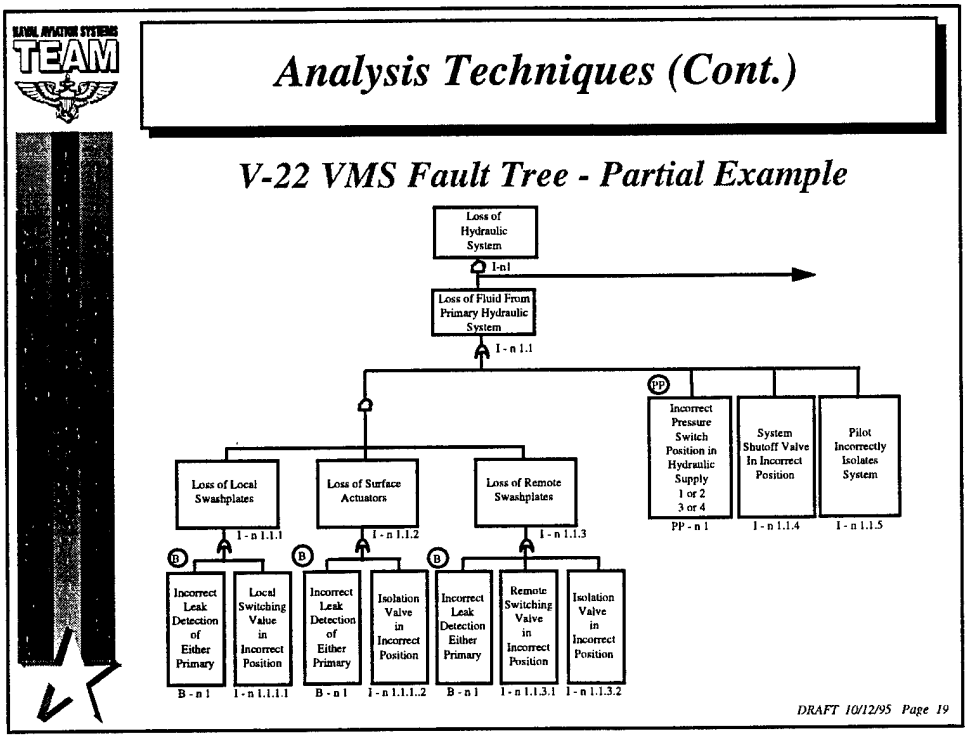


Here are the constraints.

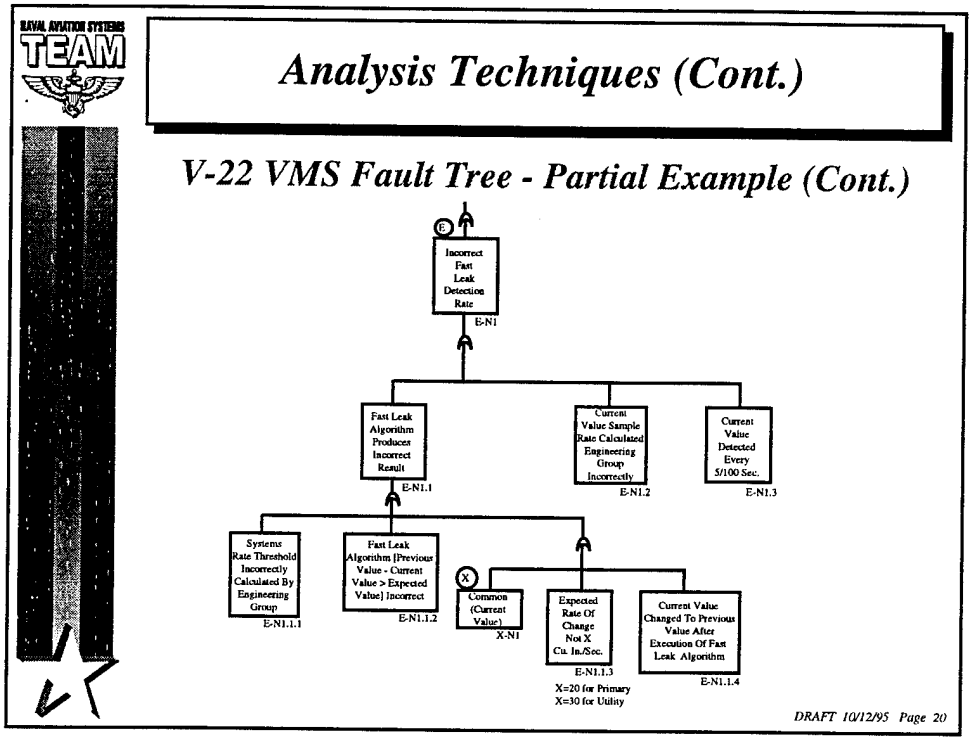


Here are a few of the potential functional hazards that were prioritized and categorized to be the most critical to analyze:

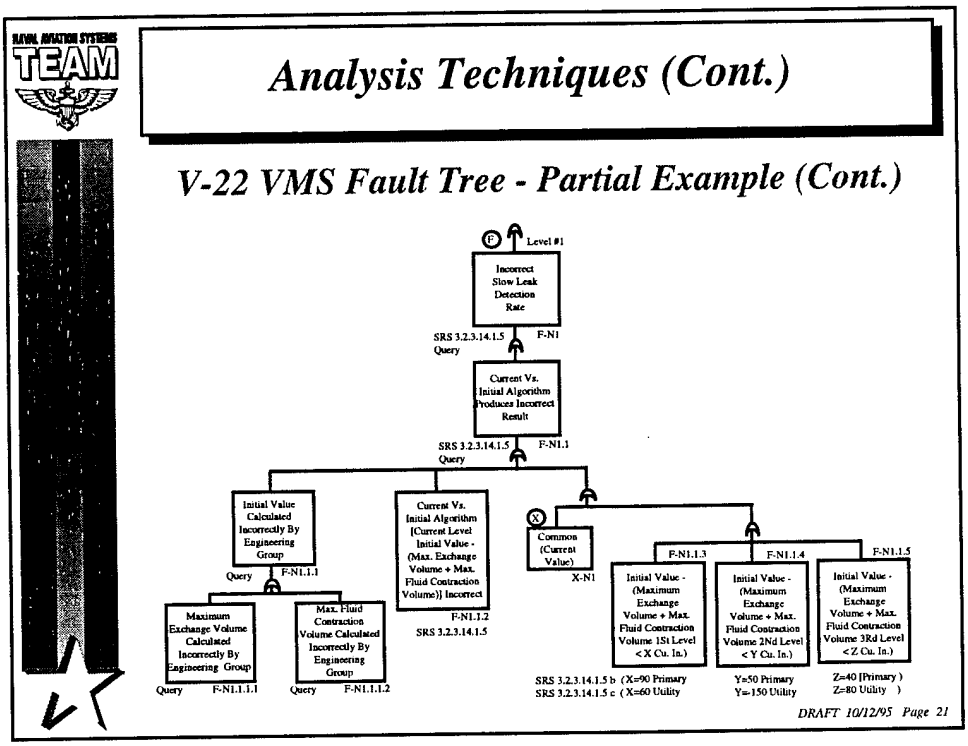
1. Simultaneous Fault Restart: Loss of Channel ID
2. Swashplate Actuator Overtilts
3. Loss of Hydraulic System
4. Inadvertent Engine Shutdown
5. Impact of SLL on Aircraft Safety
   - Excess Flapping
   - Commanded Loads Exceed Design Limits
6. Driving Both Ends of Ball Screw Simultaneously Results in Loss of Actuator Control



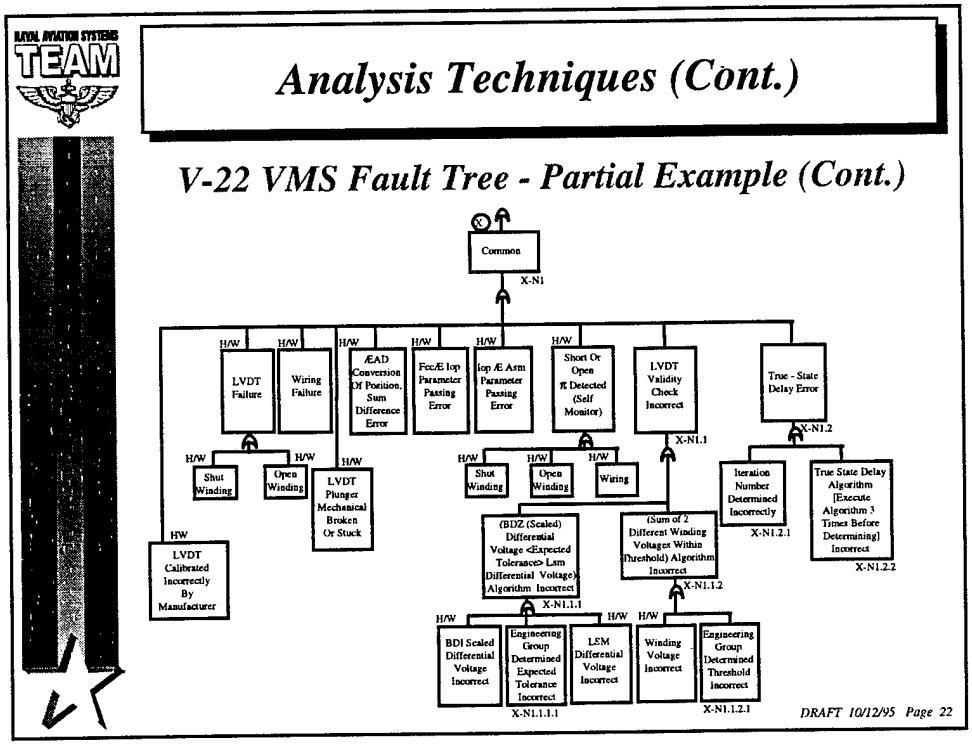
Here is a portion of the fault tree addressing the loss of hydraulics. Remember this page is the format--how we can take the analysis to additional pages. This was the first page.



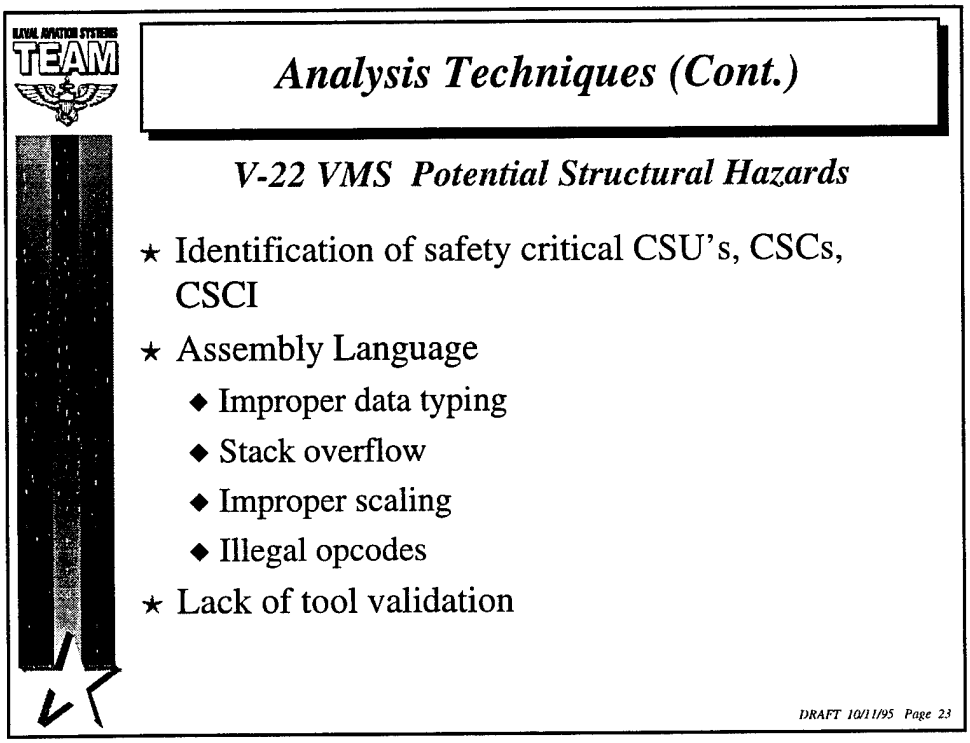
This portion shows that you can go all the way to the algorithm level. Note that you can check all the pieces of the algorithm as well as the frequency at which it executes.



This portion shows where we connected the fault tree to documentation. We were unable to graph out the associated Petri nets due to time constraints, but we are able to progressively connect them to the appropriate documentation, specifications, requirements, design, and even code.



This final example graph shows that we can easily represent hardware, human factors, and software nodes and their interoperability within the causal factor nodes.



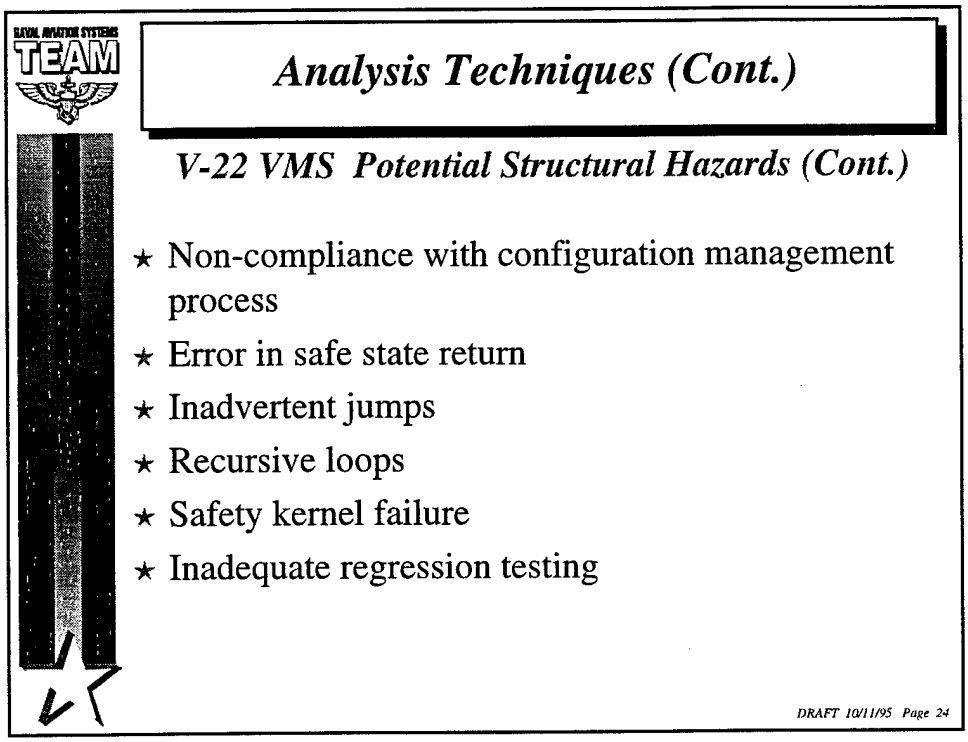
Some of the structural hazards identified and addressed in the V-22 safety program are:

★ Identification of safety critical CSU's, CSCs, CSCI

★ Assembly Language

- ◆ Improper data typing
- ◆ Stack overflow
- ◆ Improper scaling
- ◆ Illegal opcodes

★ Lack of tool validation



More potential structural hazards are :

★ Non-compliance with configuration management process

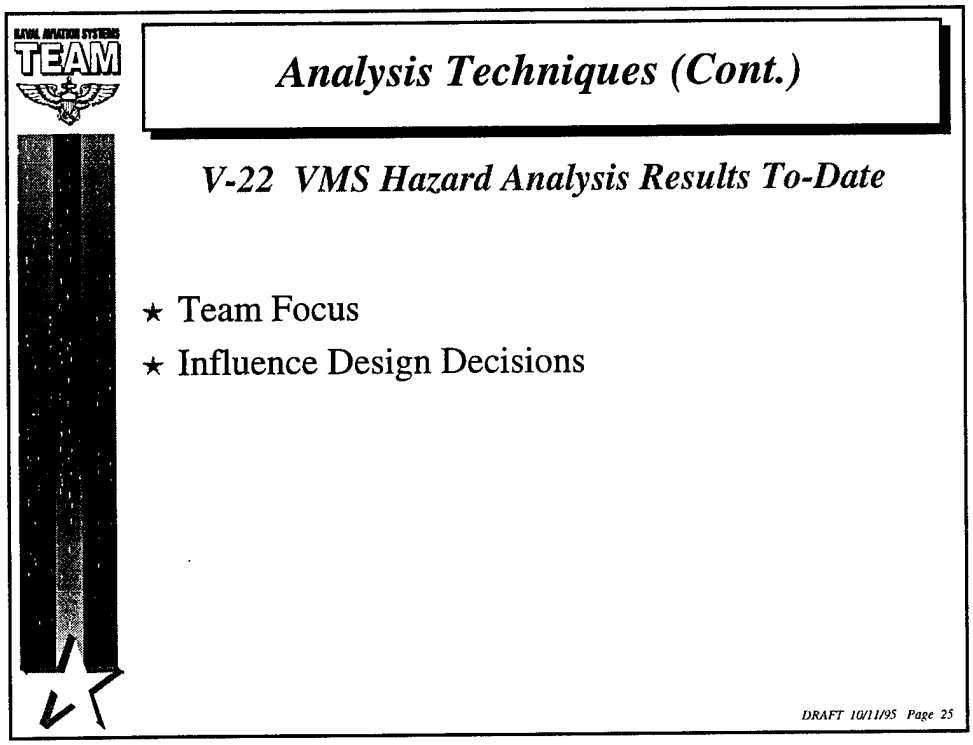
★ Error in safe state return

★ Inadvertent jumps

★ Recursive loops

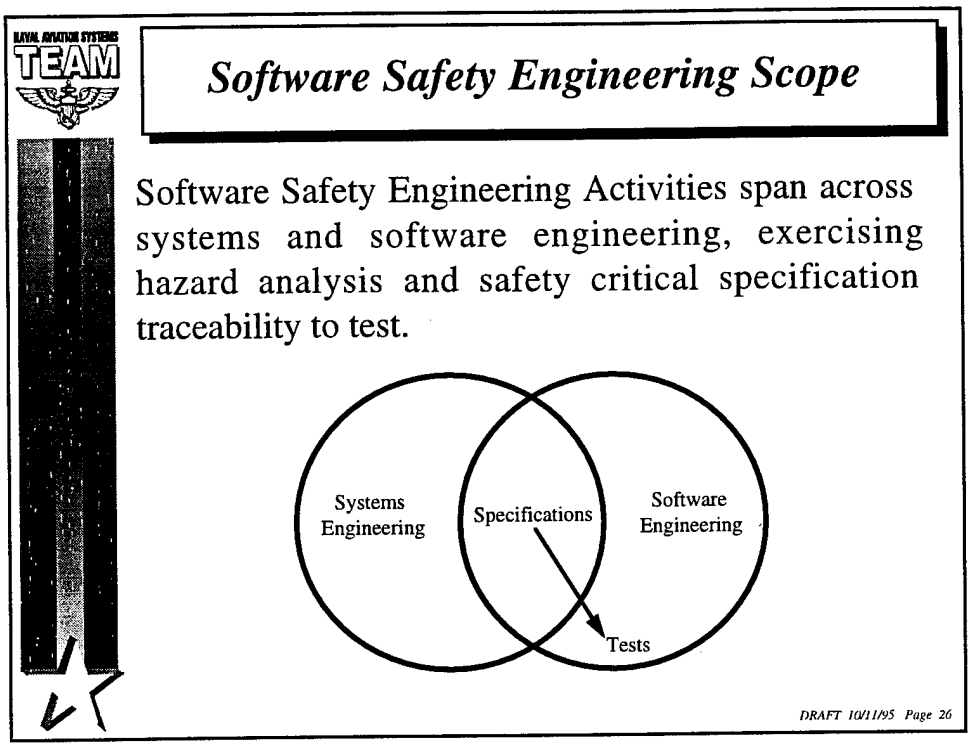
★ Safety kernel failure

★ Inadequate regression testing



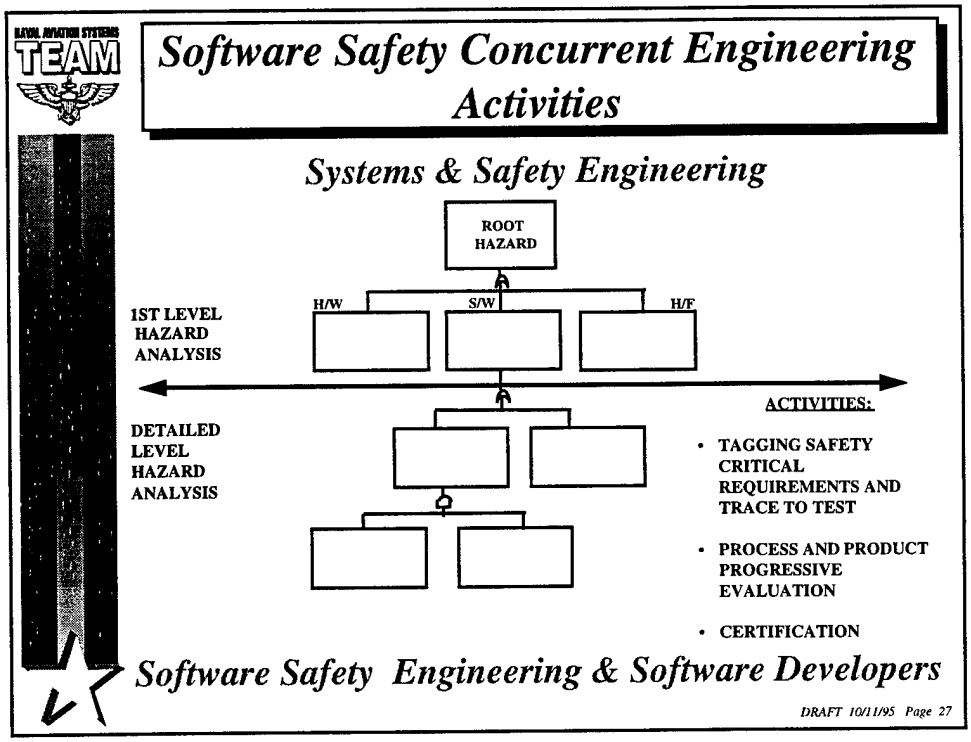
We really have executed the team concept within our safety program. There has been some reluctance; however, when they understand the goal, there has been good participation. All of the domains have been exercised. This has been a merging of the software system engineers and the safety engineers. *It takes both. Neither can execute software safety assurance alone.*

We have been able to influence the design at the most advantageous time, the beginning. While gathering data for the hydraulic hazard, systems engineers found a problem and were able to correct it in the pen and ink phase. There could be no more ideal time.

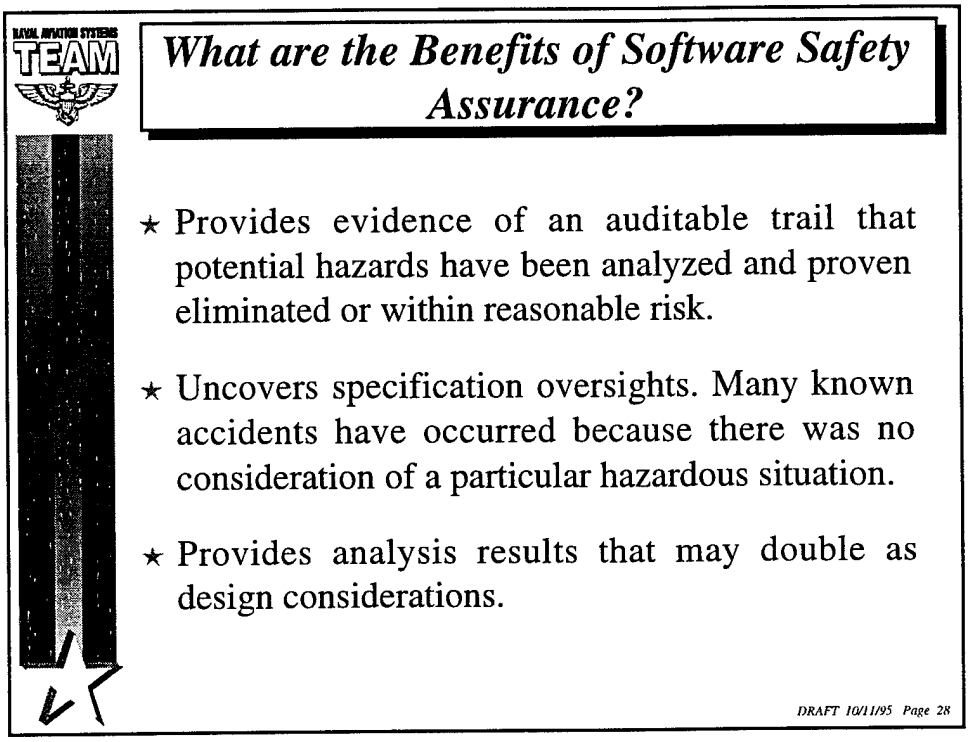


Software Safety Engineering Activities span across systems and software engineering, exercising hazard analysis and safety critical specification traceability to test.

The disciplines merge when going from system design to software design.



Usually the safety engineer determines the general (first level) hardware, human factors, and software causes to the root hazard. The software systems designers can then assist his analysis by taking the analysis into the software design. They can also participate in the safety program by executing the tagging of safety critical requirements and tracing to test, process and product progressive evaluation, and certification portions of the software safety assurance process.

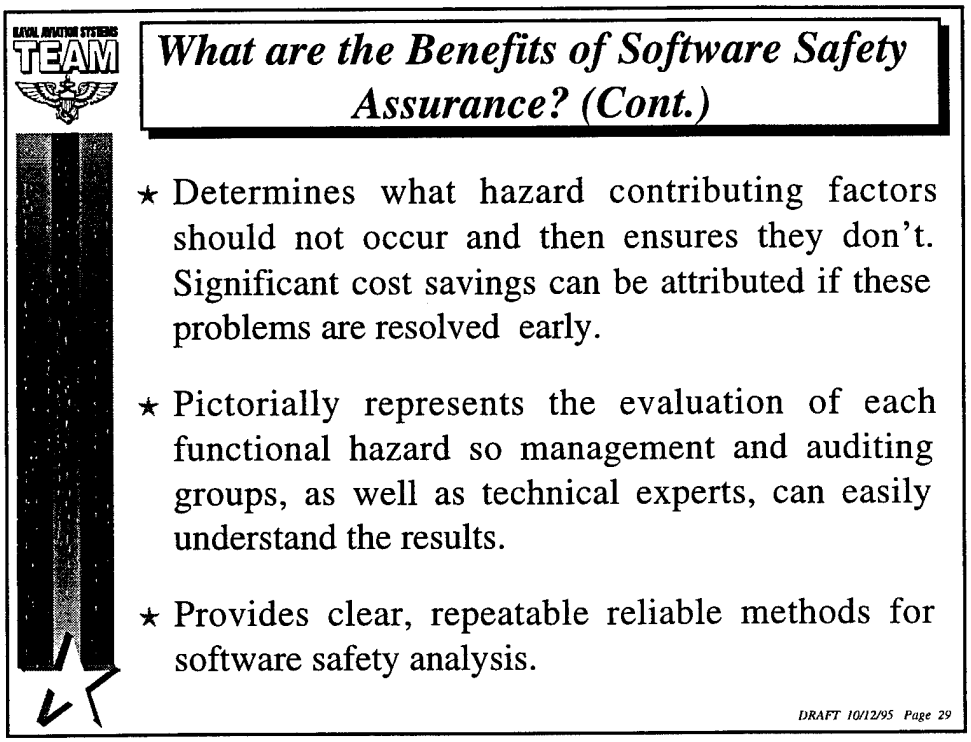


Some benefits of structured software safety assurance are:

★ It provides evidence of an auditable trail that potential hazards have been analyzed and proven eliminated or within reasonable risk.

★ It uncovers specification oversights. Many known accidents have occurred because there was no consideration of a particular hazardous situation.

★ Provides analysis results that may double as design considerations.

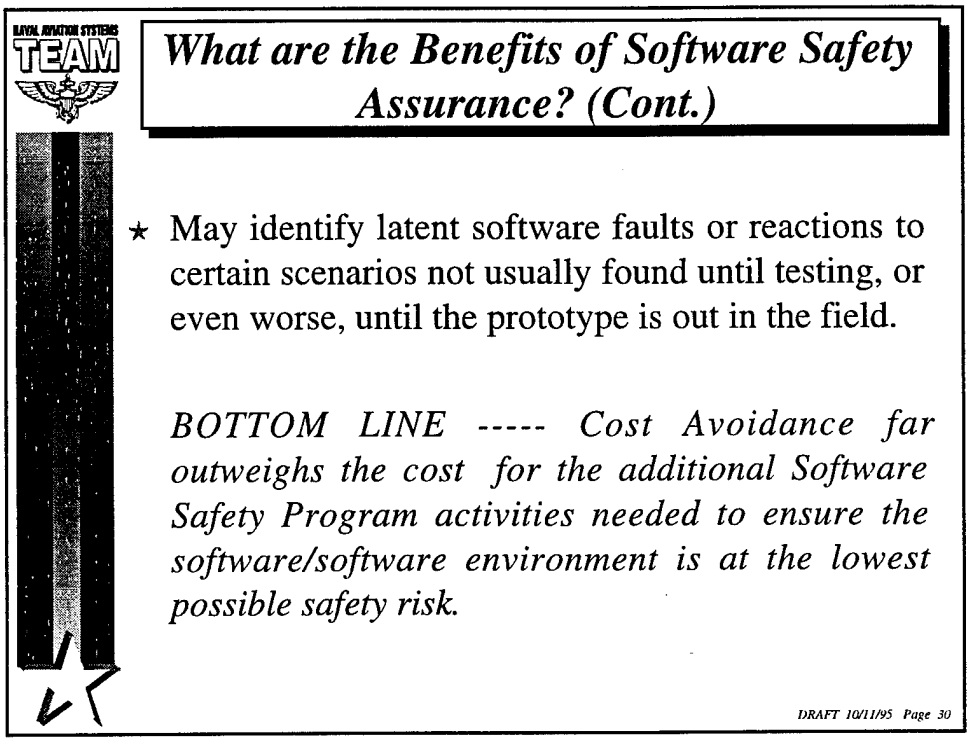


Also it helps *DESIGN OUT HAZARDS* because:

★ It determines what hazard contributing factors should not occur and then ensures they don't. Significant cost savings can be attributed if these problems are resolved early.

★ It pictorially represents the evaluation of each functional hazard so management and auditing groups, as well as technical experts, can easily understand the results.

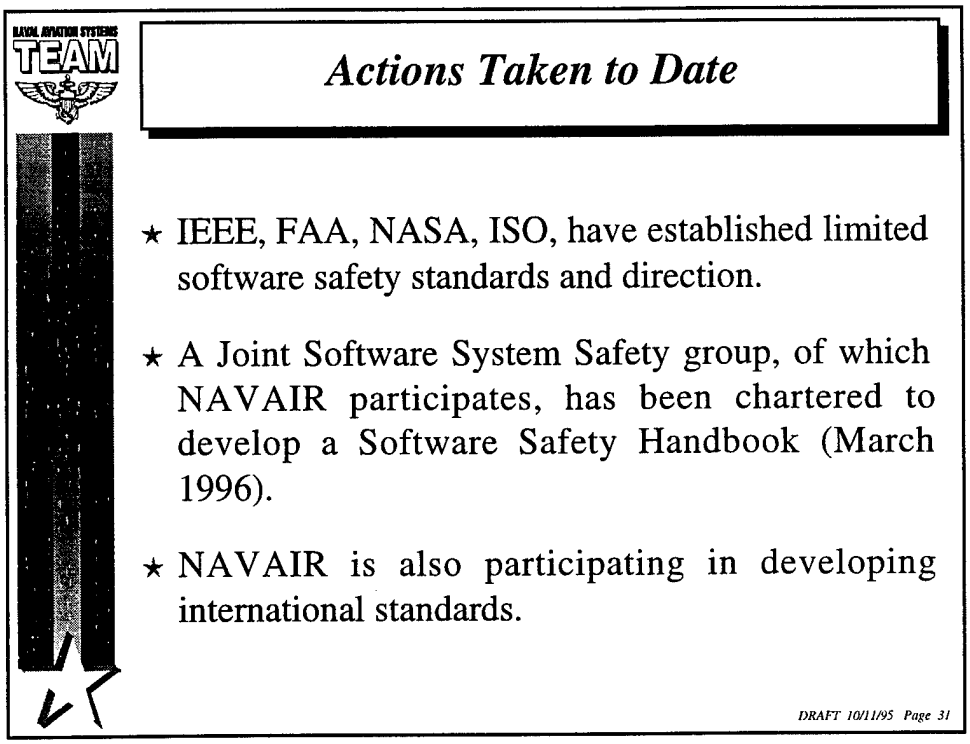
★ Provides clear, repeatable reliable methods for software safety analysis.



ALSO:

★ It may identify latent software faults or reactions to certain scenarios not usually found until testing, or even worse, until the prototype is out in the field.

The *BOTTOM LINE is that Cost Avoidance far outweighs the cost for the additional Software Safety Program activities needed to ensure the software/software environment is at the lowest possible safety risk.*

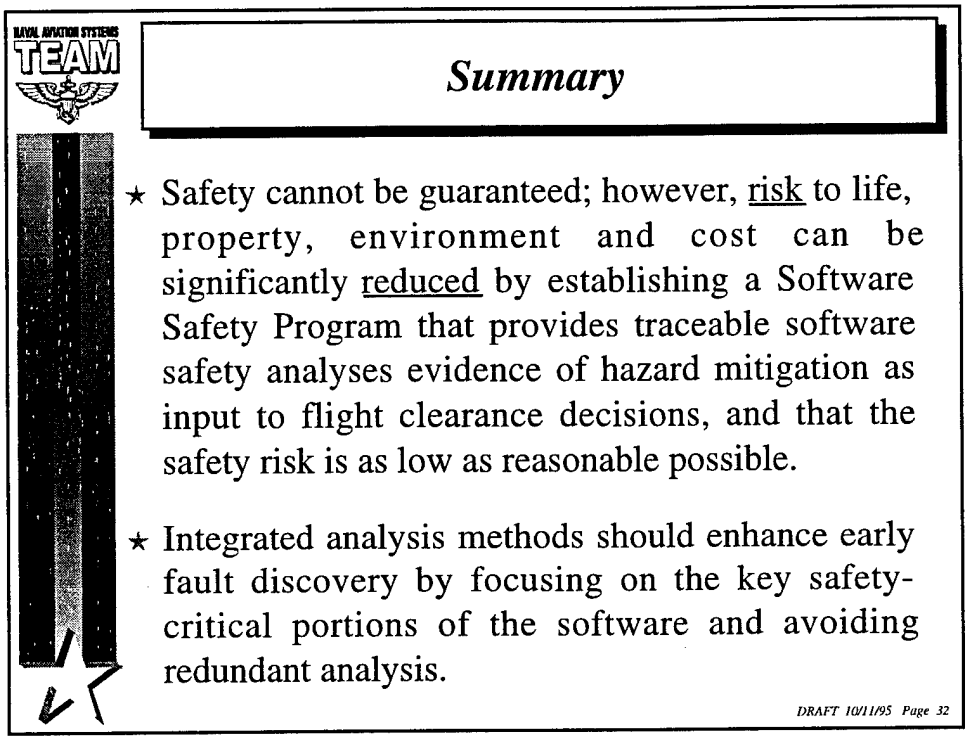


Some activities contributing to standardization are:

★ IEEE, FAA, NASA, ISO, have established limited software safety standards and direction.

★ A Joint Software System Safety group, of which NAVAIR participates, has been chartered to develop a Software Safety Handbook (March 1996).

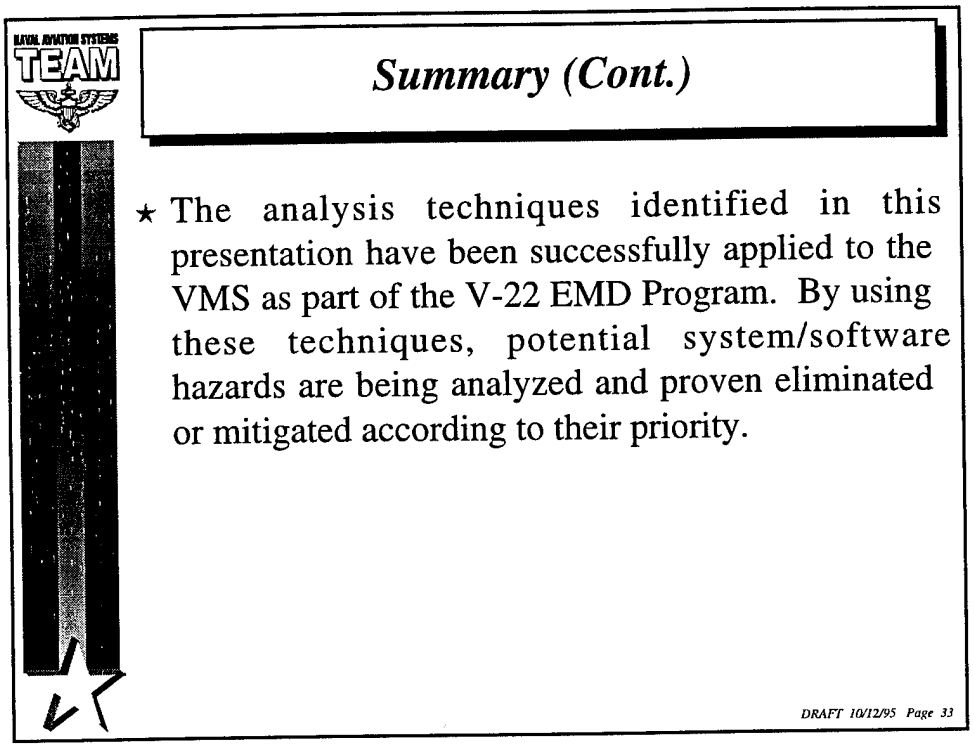
★ NAVAIR is also participating in developing international standards.



IN SUMMARY:

★ Safety cannot be guaranteed; however, risk to life, property, environment and cost can be significantly reduced by establishing a Software Safety Program that provides traceable software safety analyses evidence of hazard mitigation as input to flight clearance decisions, and that the safety risk is as low as reasonable possible.

★ Integrated analysis methods should enhance early fault discovery by focusing on the key safety-critical portions of the software and avoiding redundant analysis.



AND FINALLY:

★ The analysis techniques identified in this presentation have been successfully applied to the VMS as part of the V-22 EMD Program. By using these techniques, potential system/software hazards are being analyzed and proven eliminated or mitigated according to their priority.

PROVIDING EVIDENCE!!!