ASCA, Inc.

Use of the Dynamic Flowgraph Methodology (DFM) in Advanced PSA/PRA Applications: from Phased Missions to Software Intensive Systems

Sergio Guarro, Ph.D.

Open PSA Workshop

Washington, DC

3 October, 2007

ASCA, Inc.

Topics Covered

- **Introduction**
  - What is DFM and why we are interested in the Open-PSA initiative
- **Details and Examples of DFM Modeling**
  - DFM modeling environment
  - Inductive, deductive and test-generation uses of DFM
  - Application within CSRM (Context-based SW Risk Modeling) for Software Intensive Systems
  - Application Examples / Demo
- **Recent and Current Work, and Directions for the Future**
  - Integration of DFM with “classical PRA” modeling environments / tools
  - Participation in Open-PSA activities

ASCA, Inc.

Basic DFM Features / Capabilities

- DFM is a directed-graph, modeling methodology that uses multi-valued logic and discrete-event dynamic representation of system parameter and component states
- It is capable of handling – within the limits of the discrete state and time representations:
  - cause-effect relationships
  - time-dependent relationships
  - feedback and logic loops
- A DFM system model, once constructed, can be analyzed in either deductive (e.g., “fault-tree like”) of inductive (e.g., “FMEA or event-tree like”) mode
  - Deductive analysis produces the “prime implicants” for any “top event” that can be defined in terms of combinations of possible system parameter and/or component states (even across time boundaries)
  - Inductive analysis tracks the evolution of parameter, component and system states over discrete time and logic steps, starting from any user defined combination of states that represents a possible system state

ASCA, Inc.

Why Are We Interested in Open-PSA?

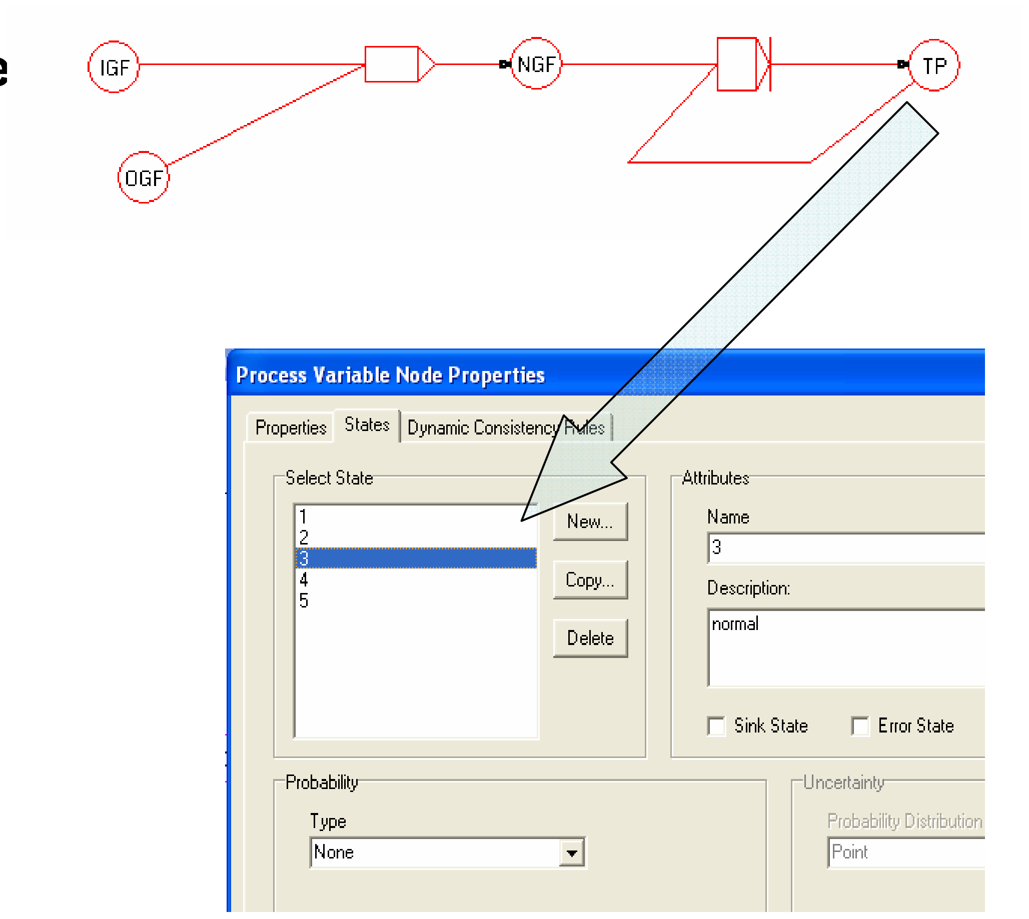
- DFM is not intended to be a substitute of any existing PRA tool (although DFM models built in “binary mode” can mimic both event-tree and fault-tree models
- DFM can be most useful as a PRA/PSA modeling supplement, for those special portions of a system or mission that call for the use of non-static, non-binary models
- For successful integration with an existing PRA/PSA framework DFM can benefit from the existence of a data interchange and API standard like the Open-PSA initiative is trying to develop and establish

ASCA, Inc.

DFM Constructs and Modeling Representations

- Nodes and discretized state vectors represent key process parameters and/or components

- Mapping between the discretized state-vectors is governed by multi-valued logic rules
  - transfer-boxes (decision tables)
  - transition-boxes (decision tables with built-in time transitions)



Select State: 1, 2, 3, 4, 5 (3 selected)

Attributes
Name: 3
Description: normal
Sink State [ ]   Error State [ ]

Probability
Type: None

Uncertainty
Probability Distribution: Point

S.Guarro

Basic Steps in a Typical DFM Analysis

- **Step 1: Model construction**
  - Construct DFM model of system of interest
    - Representing the system behavior and flow of causality
    - (Model is a network of nodes, transfer-boxes, transition-boxes and associated arc connections)
- **Step 2: System Analysis**
  - Use DFM inductive and deductive engines to:
    1. Verify specified system behavior (can be done on system “design model”)
    2. Identify system failure modes in terms of basic component failure modes (“Automated FMEA”)
    3. Identify prime implicants for system failure (“Top-Events” of interest)
    4. Define test sequences specifically suited to identify and isolate various classes of possible faults. (This feature is useful for generating input vectors for testing software based systems)

Basic Steps in a Typical DFM Analysis (cont.)

- **Step 3: Quantification of System Analysis**
  - DFM Model results usually identify subevents that contribute probability to the branch-points of a system / mission event tree
    - **DFM analysis is equivalent in concept and results to the fault-tree analyses carried out in traditional PRA to provide further definition and quantification to system sequences initially defined via event-tree models**
  - DFM “top events” are quantified in fashion similar to fault-tree “top events”
  - To quantify a DFM Top Event, the set of associated n prime implicants (PIs) is first converted into a set of m mutually exclusive implicants (MEIs)

$$\textbf{Top Event} = \mathbf{MEI_1} \vee \ldots \vee \mathbf{MEI_m}$$

  - The sum of probabilities for the MEIs yields the probability of the Top Event

$$\mathbf{P(Top\ Event) = P(MEI_1) + \ldots + P(MEI_m)}$$

  - We may venture to say that the above is the multi-value logic equivalent of the BDD quantification process for fault-trees

ASCA, Inc.

“The Good, the Bad and the Ugly” of DFM Modeling

- Construction of a “gold-standard” system model requires good understanding of system behavior
- Construction / validation of a large model requires TLC and patience
  - However the inductive – deductive step-by-step tracking capabilities of the tool do considerably facilitate the task
- A DFM model is a more compact, and therefore less immediately inspectable representation of system behavior than a fault-tree of comparable visual size
- The single system DFM model can be interrogated in many ways:
  - Deductively to analyze a large number of top events
  - Inductively to simulate the sequences from many different initial conditions

Use of DFM in “CSRM” Framework

- CSRM (Context-based Software Risk Model) is a framework to address and guide the integration of functional models of software-related risk into “classical” PRA / PSA frameworks
- CSRM is the modeling approach for software intensive space systems recommended and illustrated in the NASA PRA Procedures Guide
- CSRM can be implemented for simpler systems using only standard ET / FT PRA models
- For more complex systems, use of methods with more advanced and dynamic features (such as DFM or “colored Markov”) is recommended, at least for part of the modeling and analytical effort

ASCA, Inc.

Why “Context Based” Risk Models for Software Intensive Systems

- A majority of the software related space system missions losses that have occurred in recent year had as their root-cause or as a critical concurrent cause a software system design error
  - **Ariane 5, 1996** -- ***software design error***
  - **Delta III, 1998** -- ***LV control algorithm / software design error***
  - **Mars Polar Lander, 1999** -- ***system design flaw of which retro-rocket command SW became the “messenger”***
  - **DART Spacecraft, 2005** -- **Combination of spacecraft GN&C SW specification / design errors and test oversight**
  - **Mars Global Surveyor, 2006** -- **Combination of command uplink error and power management software design error**

Characteristics of Context-Based SW Risk Modeling (CSRM)

- Context based risk modeling is a systematic, scenario-based approach to understanding and controlling the potential vulnerabilities of a system and mission design
  - "Context based" means partitioning the potential failure domain of a system into "bins" that group together scenarios initiated by similar initiating conditions
  - This technique is a standard technique in the development of event-tree / event sequence diagram models for probabilistic risk assessment (PRA) analyses

- **Context-based modeling provides the framework and cementing logic for tools and methods specifically designed to address the risk question for complex and dynamic SW-intensive space systems applications:**
  1. **SW functional modeling, to capture and understand the system interactions by which SW may have a significant impact on mission risk scenarios**
  2. **Risk-informed SW testing and assurance, to adequately protect against such scenarios**

- **Modeling and quantification of SW risk can thus be focused on providing logic criteria and guidance for executing the SW test process in practically executable terms:**
  a) **How to logically partition the theoretical SW input-condition sampling space**
  b) **How much testing provides an adequate level of confidence in SW function execution success**

ASCA, Inc.

CSRM Basic Formulations

- A functional risk model defines SW-related risk scenarios (SRRS) as combinations of balance-of-system conditions and SW responses:

$$\boxed{SRRS_i \equiv SC_i \cap SWR_i} \quad , [1]$$

with a corresponding risk-metric conditional probability formulation:

$$\boxed{RM_i = P(SC_i) \times P(SWF \mid SC_i)} \quad , [2]$$

In this formulation:

- The system conditions SC_i are conditions for which it may or may not be possible to fully test the software, including both routine conditions and conditions at the boundary of the SW design and test envelope which may not be completely identified and defined in terms of system behavior, although they may not be ruled out as possibly being encountered during a mission.
  - For the latter, the probabilities $P(SC_i)$ may be expected to be low, whereas some of the conditional software failure probabilities $P(SWF \mid SC_i)$ can be expected to be quite high if the software has not been designed and/or has not been thoroughly tested for such a condition.

- In general, a functional / conditional risk model seeks to identify, to the extent possible, not only the conditions for which the SW has been designed to respond, but also the types of conditions for which it may have not been expressly designed or tested

ASCA, Inc.

SW Risk Quantification & Risk-informed Test Process

- **Analytical results of functionally oriented SW risk assessment models guide the risk-informed SW testing process**

  a. PRA logic model “cut-sets” and/or DFM “prime implicants” provide the list of risk contributor terms as per eqn. [1], chart 6

  b. Each system condition C_i of eqn. [1] identified in a cut-set or prime implicant defines the “test space” within which an appropriate SW module / function must be tested to assess the SW conditional risk contribution produced by the term SWR_i in eqn. [1] and quantified by the conditional probability $P(SWF \mid SC_i)$ in eqn. [2]

  c. For each condition quantitatively expressed by eqn. [2], a different level of SW module testing can be set, consistently with the estimated value of the probability $P(SC_i)$ of the system condition to which the SW is called to respond, and as necessary to demonstrate a target level of risk contribution associated with the occurrence of the system condition C_i .

  For example, if a system condition C_i corresponds to a system HW fault, its associated $P(SC_i)$ probability is typically low; thus only a limited amount of testing of the corresponding software module and function(s) is sufficient to demonstrate a value of the SW conditional probability of failure low enough to result in an acceptably low upper-bound for the i-th SW related risk contribution term $P(SC_i) \times P(SWF \mid SC_i)$ that appears in eqn. [2]

ASCA, Inc.

SW Risk Quantification & Risk-informed Test Process (cont.)

- **Results of systematically organized testing can be used to provide quantitative bounding estimates of SW related system risk contributions**
- **The PRA risk models can be quantified by making use of the basic formulations in eqns. [1] and [2] and combining HW PRA quantifications with bounding SW risk estimates obtained by means of risk-informed test executions**
  - Function oriented partitioning of the SW test space permits the use of quantification / probability-estimation methods most appropriate for the specific test conditions and associated level of risk to be demonstrated
    - **E.g., straight statistical estimation vs. "reliability growth" assessment methods based on fault identification and removal process**

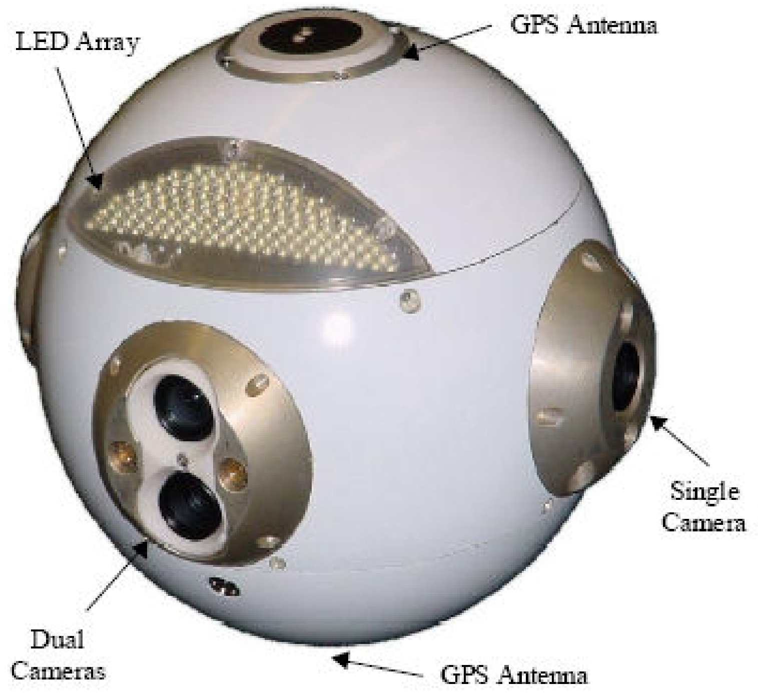
The modeling and quantification process discussed above and in the preceding charts is illustrated in the project application that follows, which utilizes a combination of "classical" PRA models and multi-valued logic Dynamic Flowgraph models (a simpler application example solely based on classical PRA modeling techniques is provided in the back-up charts

ASCA, Inc.

CSRM – DFM Application – Mini-AERCam GN&C System

This application is documented in a report recently produced by ASCA Inc for NASA JSC (POCs: Ken Chen & Henk Roelant)

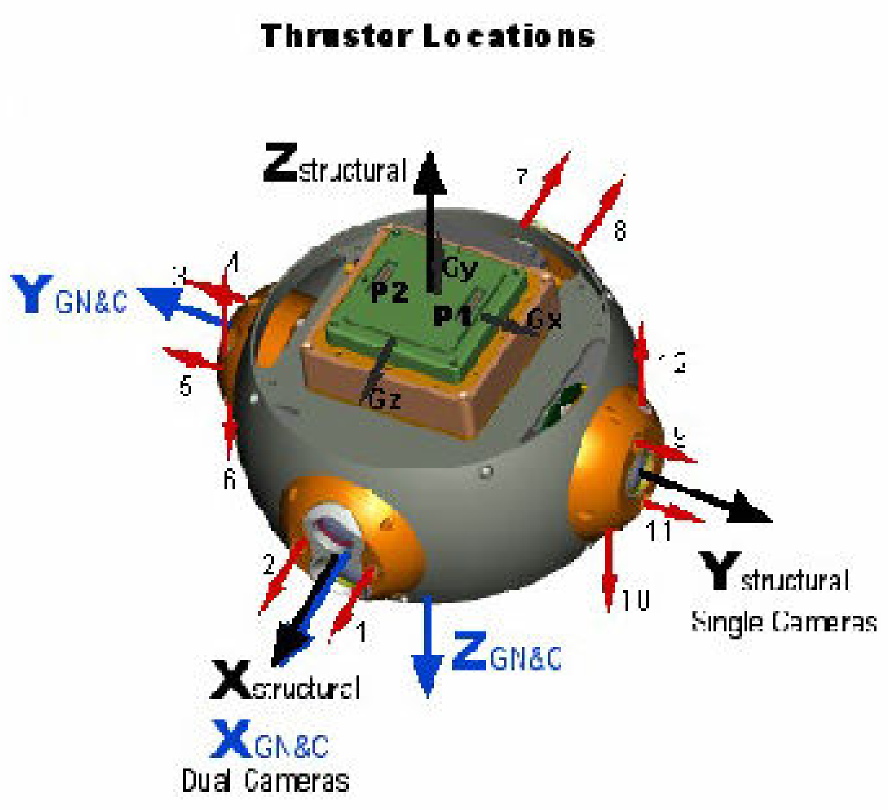
- This application considers the mission risk assessment for the Miniature Autonomous Extravehicular Robotic Camera (Mini-AERCam)
- The Mini-AERCam is a free-flying nanosatelite that provides flexible remote viewing capabilities to manned space missions. It was developed by NASA-JSC.
- The mission for the Mini-AERCam evaluated by this PRA is to support the operation of the International Space Station (ISS) by providing an orthogonal view of work being done by the ISS robotic arm.



ASCA, Inc.

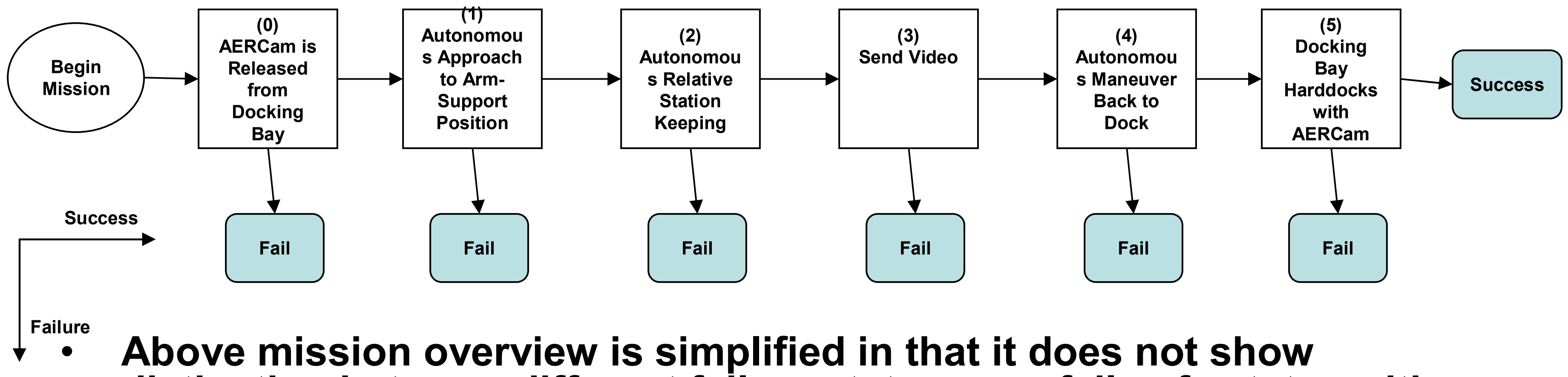
Primary Mini-AERCam Systems and Capabilities

- 12 independent fixed thrusters on central ring support movement with 6 degrees of freedom. Fueled by compressed Xenon gas from a shared tank. Each thruster cluster shares a fuel line.
- Controlled from a separate command station. 2-way communication with command station via wireless ethernet.
- Capable of manual maneuvers controlled by a translational hand controller (THC) and rotational hand controller (RHC).
- Capable of autonomous point to point movement and relative stationkeeping.
- Global Positioning System (GPS) and MEMS Gyros are used for navigation.
- Central Avionics hub provides storage, processing, and communication for all other onboard systems.
- Powered by a rechargeable Lithium Ion Battery.
- Has high-resolution still camera and two color video cameras with onboard image/video compression. Light Emitting Diode (LED) cluster provides illumination for cameras.



ASCA, Inc.

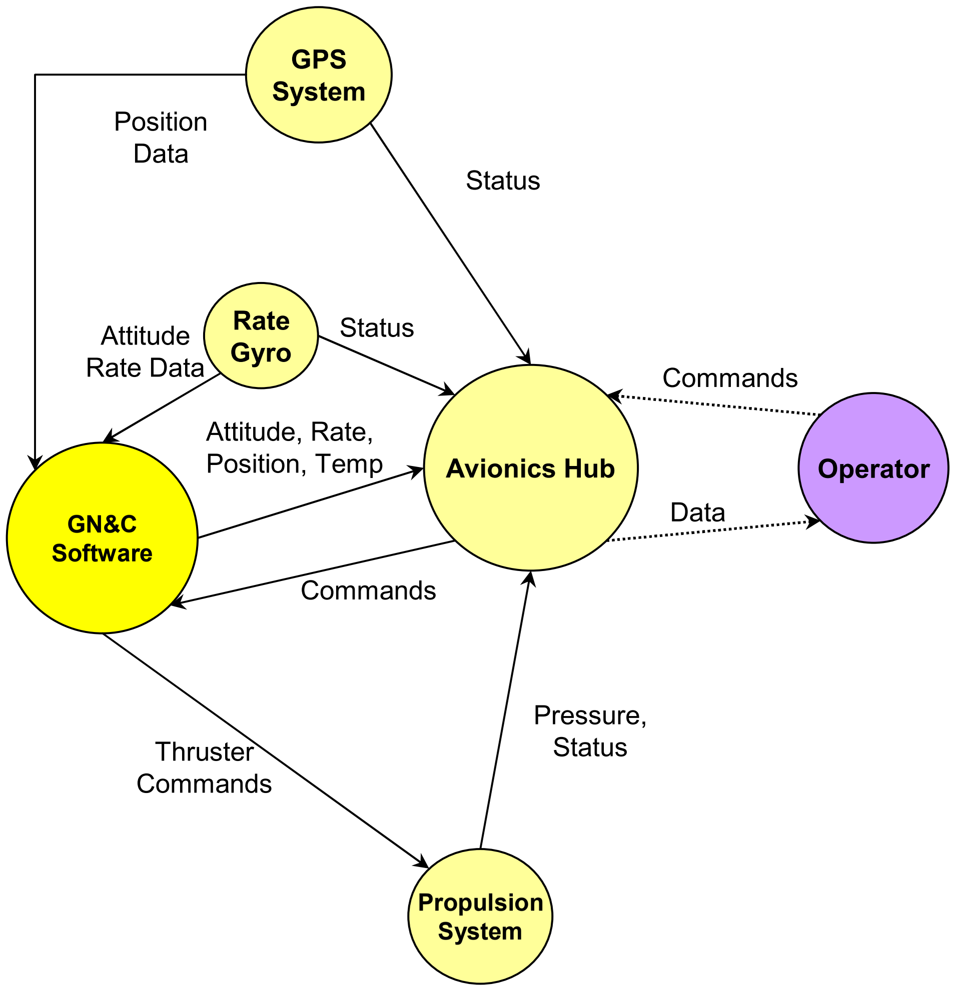
Simplified Mission Logic Overview


Begin Mission
(0) AERCam is Released from Docking Bay
(1) Autonomous Approach to Arm-Support Position
(2) Autonomous Relative Station Keeping
(3) Send Video
(4) Autonomous Maneuver Back to Dock
(5) Docking Bay Harddocks with AERCam
Success
Fail
Success
Failure

- **Above mission overview is simplified in that it does not show distinction between different failure states, e.g., fail-safe states with Mini-AERCam recovery, vs. Mini-AERCam lost, vs. Mini-AERCam collides with ISS structure**
- **Remainder of example focuses on analysis of GN&C software functions used during steps 1, 2, and 4 (approach, position hold, and return).**
  - Steps 1 and 4 are essentially identical, so evaluation of Steps 1 and 2 covers GN&C functions of interest

Mini-AERCam GN&C Functions and Schematics

- **The GN&C system has three main components:**
  - The GPS System: Determines the Mini-AERCam's position.
  - The Gyro System: Determines the Mini-AERCam's angular rate.
  - The GN&C Software System: Software that uses the Gyro, GPS data, and Operator commands to produce state data and thruster commands.
- **The GN&C system provides three functions:**
  - Navigation: Uses raw data from the GPS and Gyros to determine the position, attitude, velocity, and angular velocity of the Mini-AERCam relative to the ISS and Shuttle.
  - Guidance: Determines where the Mini-AERCam should go. In manual mode it processes user commands from the THC and RHC. In autonomous mode it automatically calculates its target position, attitude, and velocities.
  - Control: Produces the thruster commands necessary to implement the commands from the guidance system.
- **The GN&C also determines if incorrect thrust is occurring because of impact, propulsion leaks, or stuck thrusters. If so, the avionics hub is informed and signals the Mini-AERCam to enter Safe Mode.**

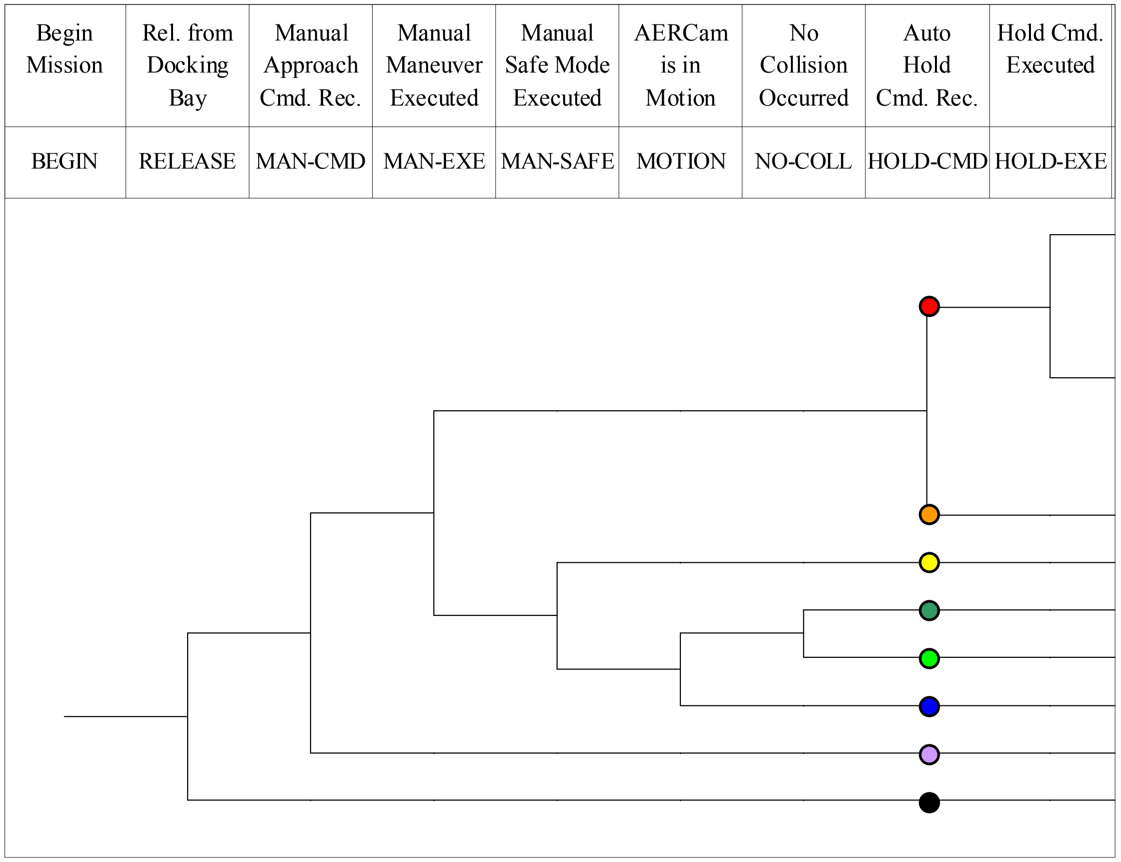


S.Guarro

ASCA, Inc.

GN&C Function Event Tree

- **A more detailed mission event tree was developed, focused primarily on the GN&C function.**
- **The event tree in this illustration identifies the key events and probabilities of interest with respect to Steps 1 and 2 shown in the previous slide.**

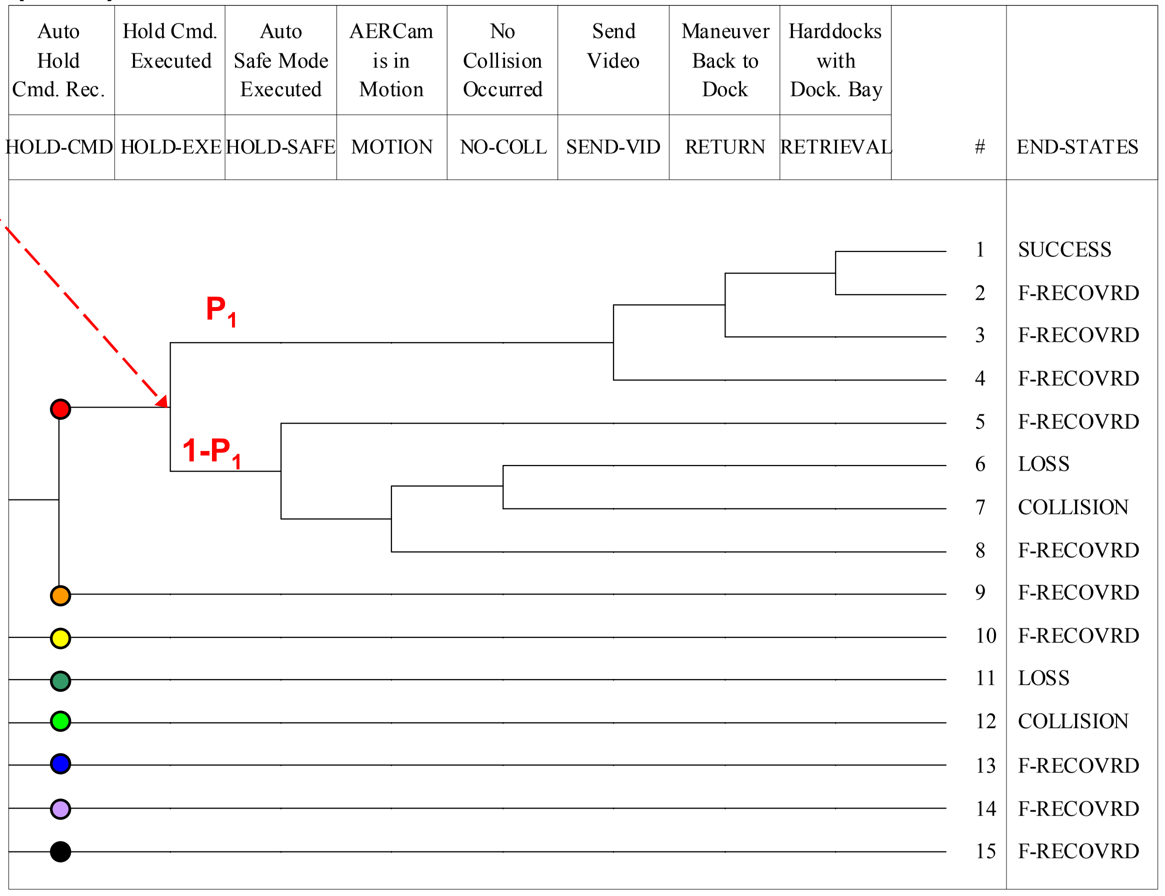


S.Guarro

GN&C Function Event Tree

- For each ET pivotal event a more detailed logic model may be developed to assess function and estimate associated probabilities
- the “Hold Command Executed” pivotal event involves relatively complex interactions of hardware and software GN&C functions and was modeled by means of a Dynamic Flowgraph Methodology (DFM) model

Develop and analyze a Dynamic Flowgraph Methodology (DFM) model of the Mini-AERCam system to:
- identify the prime implicants*,
- quantify these branch points



*** A prime implicant is a minimum combination of basic events that could cause the top event. It is the multi-valued logic equivalent of a minimal cut set.**

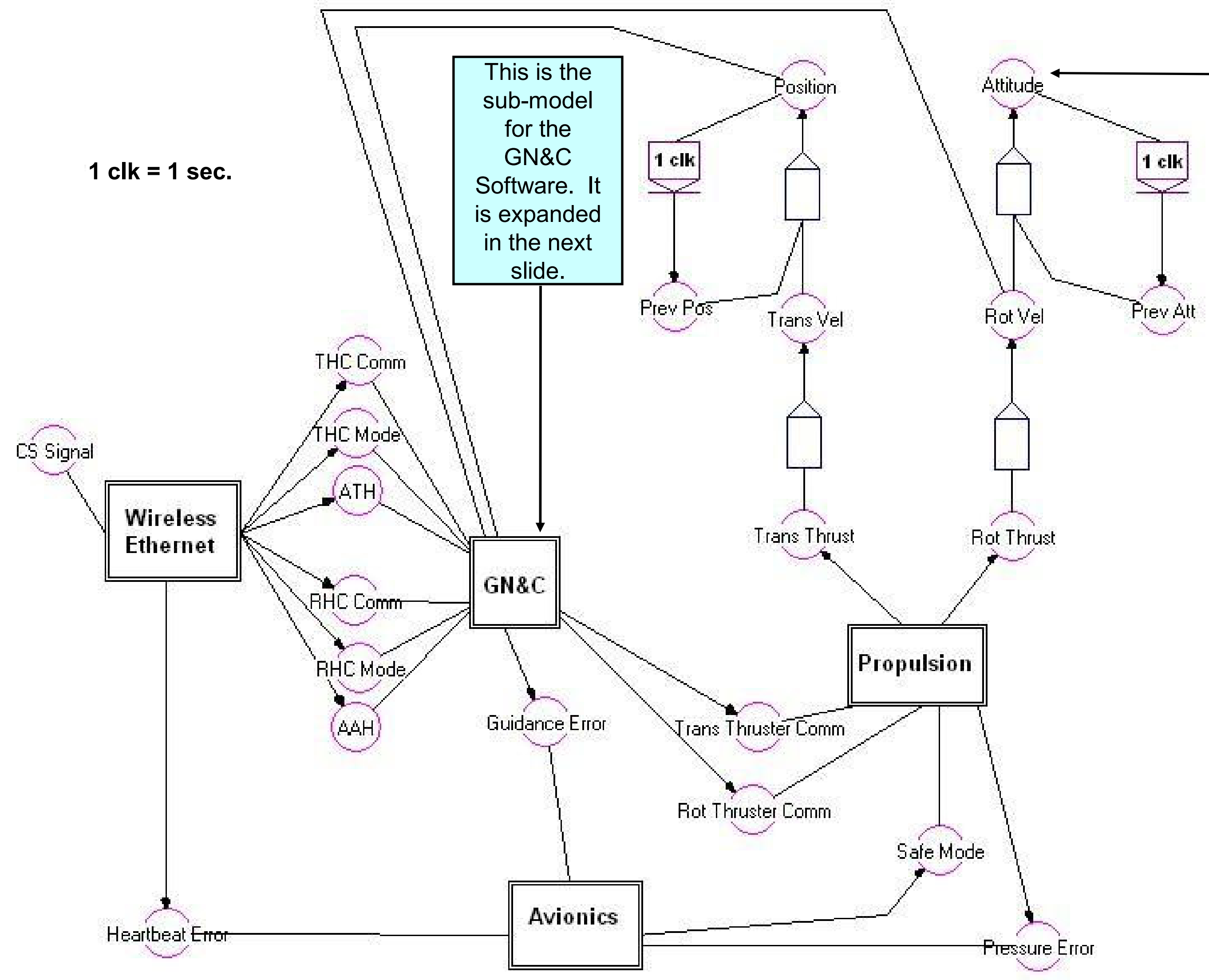
S.Guarro

ASCA, Inc.

Top Level DFM Model of the Mini-AERCam System

1 clk = 1 sec.

This is the sub-model for the GN&C Software. It is expanded in the next slide.



This node represents the actual attitude of the Mini-AERCam.

It is discretized into 3 states:

1. Correct
   (Error < 3˚)

2. Slightly Inaccurate
   (Error of 3˚ to 10˚)

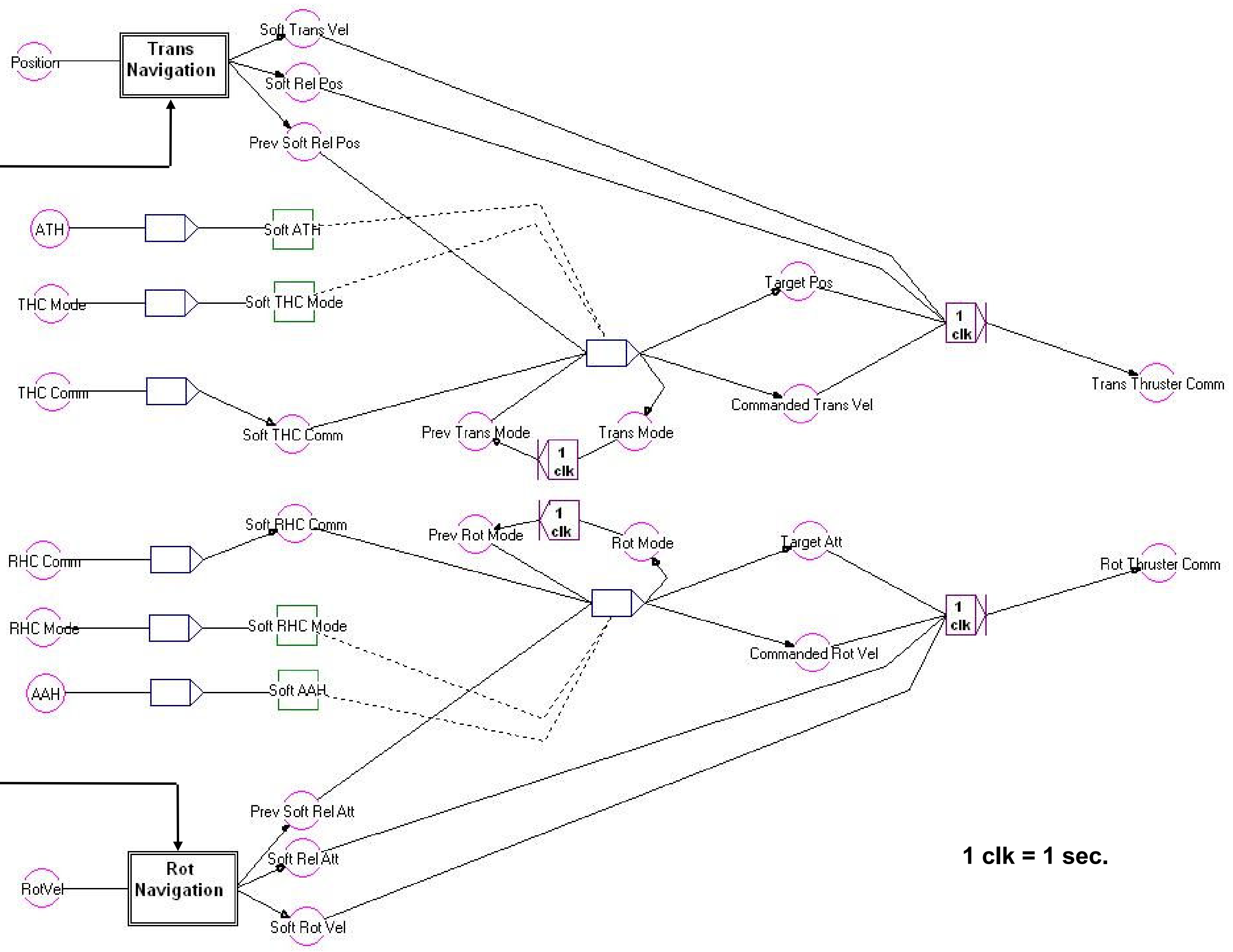
3. Inaccurate
   (Error > 10˚)

ASCA, Inc.

DFM Model of the GN&C Sub-Model

This sub-model includes the GPS hardware and the translational navigation software.

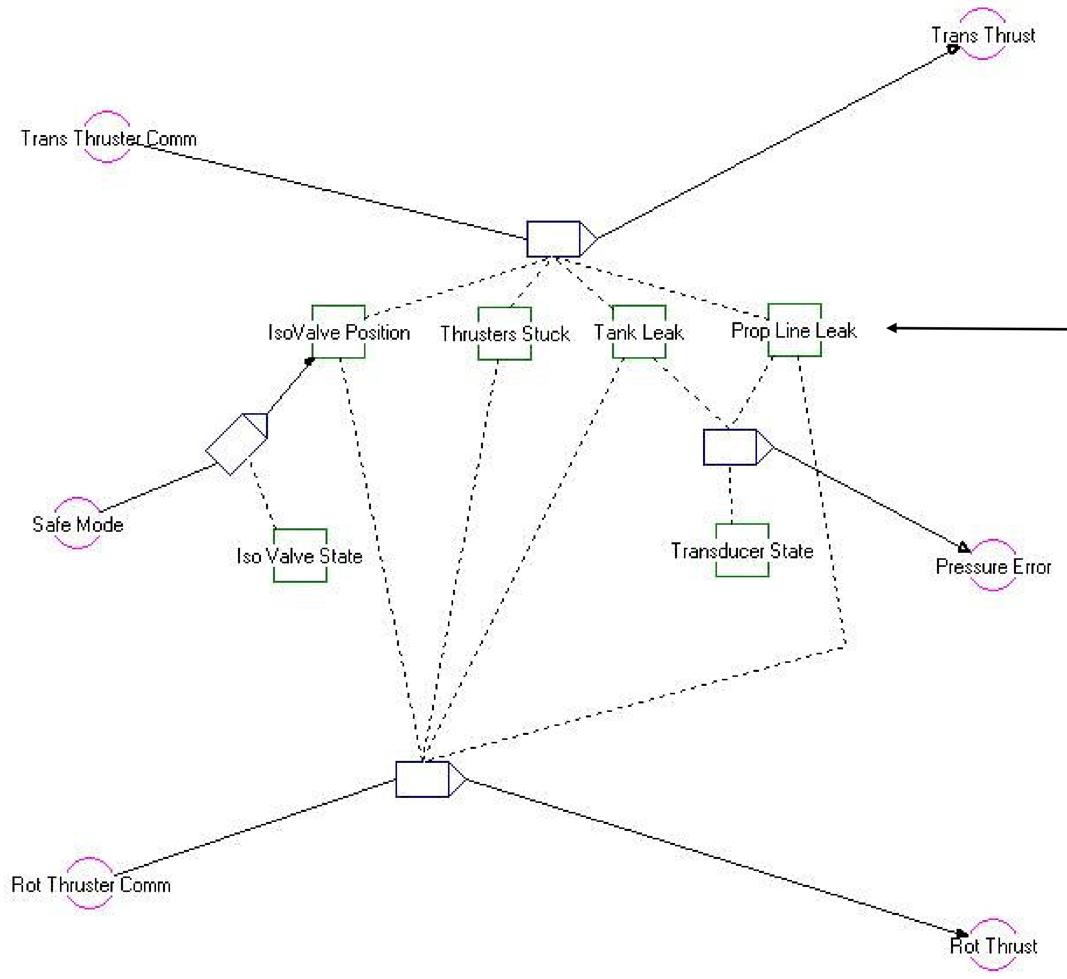
This sub-model includes the angular rate gyro hardware and the rotational navigation software.



1 clk = 1 sec.

ASCA, Inc.

DFM Model of the Propulsion Subsystem



This node represents a leak in the propulsion system fuel lines after the isovalve but before the thruster solenoids.

It is discretized into 4 states:

1. None
2. Small (1 – 40%)
   A small leak produces thrust and torque of less than 40% of the total thrust and torque the Mini-AERCam can produce to counteract it. A leak of this magnitude should not significantly affect the performance of the Mini-AERCam.
3. Large (41-80%)
   Produces thrust or torque within 80% of the Mini-AERCam’s. The Mini-AERCam can compensate and should be recoverable, but its performance is inadequate to perform its mission safely.
4. Critical (> 81%)
   The Mini-AERCam is expected to be uncontrollable.

ASCA, Inc.

Analysis of the Mini-AERCam DFM Model

- **Analysis of the Autonomous Hold Failure Top Event yields n prime implicants (PIs)**

  Top Event = $PI_1 \vee \ldots \vee PI_n$

- **Some prime implicants identify HW-only fault conditions. Some SW-only fault conditions. In general prime implicants identify combinations of hardware and software conditions that can be formally represented as per eqns. [1] and [2] on chart 6.**

  For example:

  – **Prime Implicant 1 is**

    …

    ***IsoValveCond = Stuck Closed at time -1*.**

    …

    This prime implicant identifies a condition whereby the Propellant tank iso-valve represented in the Propulsion Sub-model has stuck closed and no thrust is possible. This is a hardware-only fault.

  – **Prime Implicant 2 is**

    …

    ***TargetAtt = Inaccurate at time -1*.**

    …

    The *TargetAtt* node in the GN&C sub-model represents the accuracy of the target attitude determined by the rotational guidance software function. The PI identifies the possibility that a programmer introduced an error when coding the module, resulting in severely inaccurate output when the latter is used. This is a software-only error.

Analysis of the AERCam DFM Model (Cont.)

- **Prime Implicant 3 is**

  …

  PropLineLeak = Small Leak at time -2 and

  RotThrusterComm = Slightly Inaccurate at time -1

  …

This Prime Implicant corresponds to a combination of hardware and software conditions. **The hardware condition is a small leak in one of the propellant lines. The software condition is an algorithmic fault that causes drifting of the attitude control given a sub-nominal thrust caused by a line leak.**

– **If only one of the two conditions exists, the Mini-AERCam does not fail.**
  - The GN&C software works properly when no leak exists.
  - If a small leak occurs but there is no drift error in the attitude control, the GN&C is able to compensate for the leak by using the thrusters.
  - This PI example shows how this type of analysis identifies SW entry conditions for which the SW needs to be tested, which do not correspond to normal states of the system and may not be otherwise identified and tested for.

ASCA, Inc.

Risk-Informed Testing of Potential SW Risk Scenario and Quantification of DFM Prime Implicant

- **Prime Implicant 3 is one of the mutually exclusive implicants. It can be quantified by considering:**
  - The “entry condition” (i.e. small propellant line leak)
  - The conditional probability that the software causes an attitude shift under this triggering condition
- **From a HW failure rate database (e.g., NPRD), the entry condition can be determined to occur with a failure rate of 6.00E-06/hr. Given a mission duration of 5 hours, the associated probability is**
  $P(C_3) = 3.00E\text{-}05$.
- **The SW attitude control function can then be tested in the presence of the system (HW fault) entry condition to determine whether it performs correctly or not**
  - **Without the specific identification of the HW fault condition, random sampling of the SW input space may never cover the actual system condition !**
- **In the case discussed the risk quantification process was completed via a simulated “hardware in the loop” test process**
  - Sampling across the possible range of initial states (i.e., MiniAERCam spatial and rotational positions, compatible thruster command settings, etc.) in which the system could be at the onsetting of the leak condition.
- **With the aid of the CSRM – DFM analysis a normalized sampling set of 450 tests was sufficient to “demonstrate” a risk contribution in the order of 1.E-6 from this scenario, if no erroneous GN&C SW response was observed in the tests**
  - This was obtained via a straight Bayesian estimation, starting from a uniform, non informative prior

DFM Use in Phased-Mission Models

- **Phased-mission analysis can be challenging when executed with binary / static models**
  - Hard to carry “memory” of earlier phase failures into current phase, especially when the success criteria involving same group of components change from phase to phase
  - Some current classical PRA tools require the formulation of special post-processing rules to “trick” their solution engines into solving these situations correctly
  - <u>Simplest Example</u>: Two sequential mission phases in which the same two components (A and B) are used in the following system alignments for success:
    - Phase 1: 1 out of 2
    - Phase 2: 2 out of 2

DFM Use in Phased-Mission Models (cont.)

– Simplest Example: Two sequential mission phases in which the same two components (A and B) are used in the following system alignments for success:
  - Phase 1: 1 out of 2
  - Phase 2: 2 out of 2

Question:
- What are the min-cut-sets for Phase 1?
- for Phase 2?

(hold your answer till after the DFM demo)

ASCA, Inc.

DFM Software Tool Demo

ASCA, Inc.

Work Recently Completed or in Progress

- **NASA Applications**
  – Projects sponsored by NASA HQ/OSMA, NASA JSC and the Jet Propulsion Laboratory

- **NRC Research on “Risk Informing Digital I&C Regulation”**
  – Joint project with the Ohio State University: series of three NUREGs being published to illustrate use of DFM and “colored Markov” models in risk assessment of NPP Digital I&C systems

ASCA, Inc.

Direction of Future Work

- **Benchmarking / validation of new (32-bit Windows version) DFM tool / solution engine**
- **Integration of DFM models / results with “standard” PRA tools**
  - SAPHIRE interface being scoped / discussed
  - Others possible?
- **Participation in Open-PSA Group Activities**
  - Hoping to bring into discussion and making contributions to the topic of interface standards for integration of “classical” and “advanced” PRA / PSA modeling constructs
  - Also interested in development of cross-platform portable, standard test cases for PRA / PSA tools