Software Health Management with Bayesian Networks

Ole J. Mengshoel
Carnegie Mellon University
Moffett Field, CA 94035
Ole.Mengshoel@sv.cmu.edu

Johann Schumann
SGT, Inc. / NASA ARC
Moffett Field, CA 94035
Johann.M.Schumann@nasa.gov

Fourth IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)
August 2-4, 2011
Palo Alto, CA
<table>
<thead>
<tr>
<th>Time</th>
<th>Session 1: Reliable Software Management (Room MED-2)</th>
<th>Session 2: Autonomy and Automation (Room SAL-A)</th>
<th>Session 3: Cybersecurity and Networks (Room SAL-B)</th>
<th>Session 4: Small Spacecraft and Systems (Room STZ)</th>
<th>Room #</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:30-12:00</td>
<td>The Case for Software Health Management (<strong>031</strong>), Ashok Srivastava, NASA ARC, USA; Johann Schumann, SGT, Inc., USA</td>
<td>State-Based Scheduling via Active Resource Solving (<strong>002</strong>), Paul Morris, John Britz, NASA ARC, USA; Javier Barreiro, Michael Iatauro, SGT Inc., USA; Tristan Smith, MCT Inc., USA</td>
<td>(Invited) System Engineering Approach to Space-Cyber Situational Awareness, Joe Betser, Frank Belz, The Aerospace Corporation, USA; Roberta Ewart, USAF Space and Missile Systems Center, USA</td>
<td>PolySat’s Next Generation Avionics Design (<strong>29</strong>), Greg Manyak, John Bellardo, California Polytechnic State University, USA</td>
<td>TBD</td>
</tr>
</tbody>
</table>

Related Talks: Thursday August 4th
On January 28, 1968, a faulty electrical switch created a spark which ignited the pure oxygen environment; the fire quickly killed the Apollo 1 crew.

On September 2, 1998, Swissair 111 crashed into the Atlantic Ocean, killing all 229 people onboard. It was determined that wires short-circuited and led to a fire.

A battery failure occurred on the Mars Global Surveyor on November 2, 2006. A software error caused the battery to overheat due to over-exposure to sunlight.

In 1999, the Mars Polar Lander crashed into the surface of Mars, most likely due to a premature engine shutdown because of spurious lander leg signals.
Bayesian Methods in System Health Management

- **System health management challenges:**
  - Modeling of large, complex systems
  - Hybrid systems – discrete and continuous behavior
  - Hard reasoning problems, real time requirements

- **Probabilistic fault detection and diagnosis:**
  - Handling of broad range of faults using discrete and static Bayesian networks
  - Compilation of Bayesian networks to real-time arithmetic circuits

- **Applications:**
  - Strong performance on electrical power system data from ADAPT testbed
  - Software and sensor health management for guidance, navigation, and control
Bayesian Network Review
Bayesian Network Essentials

- Bayesian networks (BNs) are used for reasoning and learning under uncertainty [Pearl, 88]
- Probability theory and graph theory form the basis of BNs:
  - random variables are nodes
  - conditional dependencies are edges
  - instance of a graphical model
- A BN *compactly* represents a joint probability distribution
  - joint probability table size is exponential in the number of discrete random variables
- Construction of BNs:
  - statistical and data mining techniques
  - manual knowledge acquisition
  - auto-generation from structural and other knowledge

Nodes represent random variables, for example A

Nodes contain conditional distributions, for example $P(D \mid B, C)$
Bayesian Network Inference

- Bayesian network inference answers these queries:
  - *Marginal/MLV*: Given evidence at some nodes, infer posterior probability/most likely value (MLV) over one node.
  - *Most probable explanation (MPE)*: Given evidence, find explanation with greatest probability over remaining nodes.
  - *Maximum a posteriori probability (MAP)*: Given evidence, find explanation with greatest probability over some nodes.

- Computational hardness [Cooper, 1990; Shimony, 1994; Roth, 1996]
  - Care is needed, in modeling, machine learning, and inference.

- Inference algorithms:
  - *Approximate*: Stochastic local search [Kask & Dechter, 1999; Mengshoel, 1999; Mengshoel, 2008]; Variational inference; …
Bayesian Networks and Probability

Definitions:

- Random variables (nodes): \( V \)
- Evidence nodes: \( E \)
  - Includes sensors \( S \) and commands \( C \)
- Non-evidence nodes: \( X = V \setminus E \)
  - Includes health variables \( H \)
  - States of a health variable \( H \) are (for example): “healthy”, “persistently faulty”, or “intermittently faulty”
- Explanation: \( X = x \); in short \( x \)

Probability of explanation \( x \):

\[
\Pr(x \mid e) = \frac{\Pr(x, e)}{\Pr(e)} \propto \Pr(x, e) = \Pr(y)
\]

\[
\Pr(y) = \Pr(y_1, \ldots, y_n) = \prod_{i=1}^{n} \Pr(y_i \mid \text{pa}(y_i))
\]
Software Health Management using Bayesian Networks
Guidance, Navigation, and Control (GN&C)

• GN&C (Guidance, Navigation, and Control) is one of the most central software systems in a vehicle (including aircraft and spacecraft)
  • Guidance: “Where do I want to go and how do I get there?”
  • Navigation: “Where am I?”
  • Control: “Which actuators do I need to use to keep my attitude stable?”
Typical GN&C Architecture

- Software components run periodically, at different frequencies:
  - G: 2Hz
  - N: 10Hz-100Hz
  - C: 100Hz
- Software components:
  - Run in different processes
  - Use comm layer for inter-process communication

**Typical architecture:**
- PowerPC 750, RAM, Flash,
- IObus: MIL 1553 or CAN bus (automotive)
- OS: Real-time: VxWorks, RTLinux, OSEK compliant, …
GN&C Software Characteristics

Typical GN&C software characteristics:

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>N</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode logic</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>numerical</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>coordinates</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>state machine</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>control loop</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Kalman Filter</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Comm/OS</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Update</td>
<td>slow</td>
<td>med</td>
<td>fast</td>
</tr>
</tbody>
</table>

Possible software bugs:

- logic errors in Guidance state machine
- broken or very noisy sensors/actuators
- singularities in navigation (“crossing the date line”, F22 Raptors flying to Japan)
- communication failures between G,N,C software components
- OS bugs: timing, stack, memory, ...
- ...
Demonstration Scenario: On-Line
Sensing Software Status

Sensing status of software:

- Extract status directly from software
- Components interacting with software: Hardware sensors, OS, middleware, computer hardware, ...

<table>
<thead>
<tr>
<th>Software</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>flagged errors and exceptions</td>
</tr>
<tr>
<td>Memsize</td>
<td>used memory</td>
</tr>
<tr>
<td>Quality</td>
<td>signal quality</td>
</tr>
<tr>
<td>Reset</td>
<td>Filter reset (Navigation)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Intent</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FS_write</td>
<td>intent to write to FS</td>
</tr>
<tr>
<td>fork</td>
<td>intent to create new process(es)</td>
</tr>
<tr>
<td>malloc</td>
<td>intent to allocate memory</td>
</tr>
<tr>
<td>use_msg</td>
<td>intent to use message queues</td>
</tr>
<tr>
<td>use_sem</td>
<td>using semaphores</td>
</tr>
<tr>
<td>use_recursion</td>
<td>using recursion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operating System</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>CPU load</td>
</tr>
<tr>
<td>N_proc</td>
<td>number processes</td>
</tr>
<tr>
<td>M_free</td>
<td>available memory</td>
</tr>
<tr>
<td>D_free</td>
<td>percentage of free disk space</td>
</tr>
<tr>
<td>IPC</td>
<td>amount of available IPC</td>
</tr>
<tr>
<td>Semaphores</td>
<td>information about semaphores</td>
</tr>
<tr>
<td>realtime</td>
<td>missed deadlines</td>
</tr>
<tr>
<td>N_intr</td>
<td>number of interrupts</td>
</tr>
<tr>
<td>L_msgqueue</td>
<td>length of message queues</td>
</tr>
</tbody>
</table>

- Example:
  - Aircraft control works properly
  - However, other tasks might consume too many resource
  - This can lead to failure related to control task
- Need multitude of information sources
Demonstration Scenario: Bayesian Network

Bayesian Network Diagram:
- Health File_System
- Sensor File_System Error
- Delay
- Oscillation
- Status File_System
- Status Message_queue
- Sensor Queue_length
- Sensor Delta_queue
- Write File_System

Rest of Bayesian SWHM Network:
- Science Camera
- Transmitter
- SWHM System
- GN&C Guidance Navigation Control
- Message Queue
- RTOS Emulator (OSEK/Trampoline)
Bayesian network model of a system:

- It represents health of sensors and system components explicitly.
- It contains random variables for other parts of a system.

A probabilistic and computational approach to:

* **Diagnosis**: health state of system component nodes
* **Sensor validation**: health state of sensor nodes
* **Prognosis**: estimate of future health state of system
Experimental Results

Nominal scenario

Fault scenario
V&V Process

Bayesian Network

SWHM Arithmetic Circuit (Knowledge Base)

ISWHM Arithmetic Circuit Inference Engine

GN&C Guidance Navigation Control

RTOS Emulator (OSEK/Trampoline)

Domain knowledge
Reliability data

SSHM Requirements

SSHM Model

Parametric Model Analysis

Model Review

Model Coverage Testcase Generation

Compilation/translation

SSHM Implementation

Code Review
Static Analysis

Model Checking

Code Coverage Testing

Worst Case Exec. Time (data-driven)

System Integration

Flight Readiness Review
Electrical Power System Health Management using Bayesian Networks
Architecture using Bayesian Networks

Each health variable has at least two states (healthy and faulty), thus enabling the diagnoses of zero, one, two, or more faults.

See Mengshoel et al., 2008] and [Mengshoel et al., 2009] for BN auto-construction.
Electrical Power Systems at NASA

- Electrical power systems (EPSs) are critical in aerospace
- EPS loads include: avionics, propulsion, life support, and thermal management
  - increased EPS use in air- and spacecraft
- ADAPT EPS testbed at NASA Ames:
  - a capability for controlled insertion of faults, giving *repeatable failure scenarios*;
  - a *standard testbed* for evaluating diagnostic algorithms & software; and
  - a *stepping stone* for maturing diagnostic technologies.

See also http://ti.arc.nasa.gov/projects/adapt
The Hybrid Systems Challenge

Hybrid systems:
- **discrete**: Both healthy and faulty modes
- **continuous**: Both healthy and faulty behavior

Fault types in hybrid systems:
- *abrupt discrete faults*
- *abrupt continuous (parametric) faults*

  - **offset**
  - **stuck**

A sensor or component may see an arbitrarily small and faulty drop or increase in its value.

A sensor or component get stuck at any continuous value.
Fault Types - Part I

**Independent faults**

Abrupt
- Permanent
- Discrete
- Continuous (parametric)

Intermittent

Incipient

**Dependent faults**

Common cause

Cascading

Bayesian networks in general

See [Kurtoglu et al., 2009a] and [Kurtoglu et al., 2009b] for discussion of fault types

### Component Fault Description

<table>
<thead>
<tr>
<th>Component</th>
<th>Fault Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery</td>
<td>Degraded</td>
</tr>
<tr>
<td>Boolean Sensor</td>
<td>Stuck at Value</td>
</tr>
<tr>
<td>Circuit Breaker</td>
<td>Tripped</td>
</tr>
<tr>
<td></td>
<td>Failed Open</td>
</tr>
<tr>
<td></td>
<td>Stuck Closed</td>
</tr>
<tr>
<td>Inverter</td>
<td>Failed Off</td>
</tr>
<tr>
<td>Relay</td>
<td>Stuck Open</td>
</tr>
<tr>
<td></td>
<td>Stuck Closed</td>
</tr>
<tr>
<td>Sensor</td>
<td>Stuck at Value Offset</td>
</tr>
<tr>
<td>Pump (Load)</td>
<td>Flow Blocked</td>
</tr>
<tr>
<td></td>
<td>Failed Off</td>
</tr>
<tr>
<td>Fan (Load)</td>
<td>Over Speed</td>
</tr>
<tr>
<td></td>
<td>Under Speed</td>
</tr>
<tr>
<td></td>
<td>Failed Off</td>
</tr>
<tr>
<td>Light Bulb (Load)</td>
<td>Failed Off</td>
</tr>
</tbody>
</table>

Tier-1 and Tier-2 of DXC-09 diagnostics competition
ADAPT: Electrical Power System Testbed

Note: Tier 1 experiments are substantially easier than Tier 2 experiments:
- a subset of ADAPT was used
- relays were closed at all times
We consider different fault types: (1) abrupt persistent, (2) abrupt intermittent, ... Multiple faults can take place, and they may be simultaneous or sequential.

We emphasize the ProDiagnose algorithm.

See [Kurtoglu et al., 2008; Kurtoglu et al., 2009a; Kurtoglu et al., 2009b; Poll et al., 2010] for further discussions of metrics and benchmarking.

The ADAPT testbed is used for experimentation.
Bayesian Network for DXC-09 Tier 2

The Bayesian network model of ADAPT Tier 2.
Two types of scenarios:
- Tier 1 scenarios: nominal or contained one fault
- Tier 2 scenarios: nominal or contained single, double, or triple faults

The ADAPT EPS was used to generate fault and nominal scenarios:
- Faults were injected simultaneously or sequentially
- Fault types were additive parametric (abrupt changes in parameter values) and discrete (unexpected changes in system mode)
- Faults were permanent and included both component faults and sensor faults

<table>
<thead>
<tr>
<th>Metric</th>
<th>ADAPT DXC Tier 1</th>
<th></th>
<th>ADAPT DXC Tier 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ProADAPT</td>
<td>RODON</td>
<td>HyDE-S</td>
<td>ProADAPT</td>
</tr>
<tr>
<td>False positives (FP) rate</td>
<td>0.0333</td>
<td>0.0645</td>
<td>0.2000</td>
<td>0.0732</td>
</tr>
<tr>
<td>False negatives (FN) rate</td>
<td>0.0313</td>
<td>0.0968</td>
<td>0.0741</td>
<td>0.1392</td>
</tr>
<tr>
<td>Detection accuracy</td>
<td>0.9677</td>
<td>0.9194</td>
<td>0.8548</td>
<td>0.8833</td>
</tr>
<tr>
<td>Classification errors</td>
<td>2.0</td>
<td>10.0</td>
<td>26.0</td>
<td>76.0</td>
</tr>
<tr>
<td>Mean time to detect $T_d$ (ms)</td>
<td>1,392</td>
<td>218</td>
<td>130</td>
<td>5981</td>
</tr>
<tr>
<td>Mean time to isolate $T_i$ (ms)</td>
<td>4,084</td>
<td>7,205</td>
<td>653</td>
<td>12,486</td>
</tr>
<tr>
<td>Mean CPU time $T_c$ (ms)</td>
<td>1,601</td>
<td>11,766</td>
<td>513</td>
<td>3,416</td>
</tr>
<tr>
<td>Mean peak memory usage (kb)</td>
<td>1,680</td>
<td>26,679</td>
<td>5,795</td>
<td>6,539</td>
</tr>
</tbody>
</table>

| Score                              | 72.80            | 59.85        | 59.50            | 83.20            | 81.50          | 70.50          |
| Rank                               | 1                | 2            | 3                | 1                | 2              | 3              |

9 competitors in Tier 1. 6 competitors in Tier 2.
ADAPT scenarios with zero to three abrupt discrete and abrupt continuous faults were generated. Faults, if any, were inserted simultaneously or sequentially.

**ProADAPT1: DXC-09 version of ProADAPT - May 2009.**

<table>
<thead>
<tr>
<th></th>
<th>ProDiagnose (ProADAPT1)</th>
<th>FaultBuster</th>
<th>HyDE</th>
<th>RODON</th>
<th>Stanford</th>
<th>Wizards Of Oz</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positives</td>
<td>7.32%</td>
<td>81.43%</td>
<td>0.00%</td>
<td>54.17%</td>
<td>32.16%</td>
<td>51.06%</td>
</tr>
<tr>
<td>False Negatives</td>
<td>13.92%</td>
<td>24.00%</td>
<td>30.00%</td>
<td>9.72%</td>
<td>5.19%</td>
<td>9.59%</td>
</tr>
<tr>
<td>Classification Errors</td>
<td>76</td>
<td>130</td>
<td>121.57</td>
<td>84.01</td>
<td>110.55</td>
<td>159.25</td>
</tr>
<tr>
<td>Detection Accuracy</td>
<td>88.33%</td>
<td>42.50%</td>
<td>80.00%</td>
<td>72.50%</td>
<td>85.00%</td>
<td>74.17%</td>
</tr>
<tr>
<td>Mean Time to Detect</td>
<td>5973 ms</td>
<td>14099 ms</td>
<td>17610 ms</td>
<td>3490 ms</td>
<td>3946 ms</td>
<td>30742 ms</td>
</tr>
<tr>
<td>Mean Time to Isolate</td>
<td>11988 ms</td>
<td>37808 ms</td>
<td>21982 ms</td>
<td>36331 ms</td>
<td>14103 ms</td>
<td>47625 ms</td>
</tr>
<tr>
<td>Mean CPU Time</td>
<td>2922 ms</td>
<td>5798 ms</td>
<td>29612 ms</td>
<td>80261 ms</td>
<td>963 ms</td>
<td>23387 ms</td>
</tr>
<tr>
<td>Mean Peak RAM Usage</td>
<td>6539 KB</td>
<td>10261 KB</td>
<td>20515 KB</td>
<td>29878 KB</td>
<td>5912 KB</td>
<td>7498 KB</td>
</tr>
</tbody>
</table>

**ProADAPT2: September 2009 version of ProADAPT.**

<table>
<thead>
<tr>
<th></th>
<th>ProADAPT1</th>
<th>ProADAPT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positives</td>
<td>7.32%</td>
<td>0.00 %</td>
</tr>
<tr>
<td>False Negatives</td>
<td>13.92%</td>
<td>1.25 %</td>
</tr>
<tr>
<td>Classification Errors</td>
<td>76</td>
<td>20</td>
</tr>
<tr>
<td>Detection Accuracy</td>
<td>88.33%</td>
<td>99.17 %</td>
</tr>
<tr>
<td>Mean Time to Detect</td>
<td>5973 ms</td>
<td>2096 ms</td>
</tr>
<tr>
<td>Mean Time to Isolate</td>
<td>11988 ms</td>
<td>10961 ms</td>
</tr>
</tbody>
</table>
Experiments, Simulated ADAPT Data

<table>
<thead>
<tr>
<th>Inference Time (ms)</th>
<th>MPE</th>
<th>Marginals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VE</td>
<td>ACE</td>
</tr>
<tr>
<td>Minimum</td>
<td>17.25</td>
<td>0.1967</td>
</tr>
<tr>
<td>Maximum</td>
<td>38.45</td>
<td>2.779</td>
</tr>
<tr>
<td>Median</td>
<td>17.63</td>
<td>0.1995</td>
</tr>
<tr>
<td>Mean</td>
<td>17.79</td>
<td>0.2370</td>
</tr>
<tr>
<td>St. Dev.</td>
<td>1.513</td>
<td>0.2137</td>
</tr>
</tbody>
</table>

Comparison between Arithmetic Circuit Evaluation (ACE), Variable Elimination (VE) and Clique Tree Propagation (CTP)

Main conclusions:
- All three inference algorithms are quite efficient, thanks to auto-generation algorithm.
- ACE outperforms VE (for MPE) and CTP (for marginals), both in Mean and St. Dev.

ACE is the approach used in ProADAPT.
Bayesian Methods for Software Health Management

- System health management challenges include:
  - Modeling of large, complex systems including aircraft and spacecraft
  - Hybrid systems – discrete and continuous
  - Hard reasoning problems, real time requirements

- Bayesian network for SW health management:
  - Leverage existing work on system health management of hardware, for example ADAPT diagnostic system
  - Recognize the unique challenges and opportunities of SW
  - Suitable for different types of SW components
  - Monitoring on different layers (OS, middle-ware, process, …) - modularity
  - Integrated handling of HW and SW faults

Bayesian Reasoning for EPS Diagnostics: Operates in a state space of size $> 2^{500}$ in time < 1 ms.
Publications


BACKUP SLIDES
Bayesian Network Applications

System health monitoring (Diagnosis, monitoring, prognosis, ...)
  Electrical power systems
  Guidance, navigation, and control

Spam filtering
  Naïve Bayes (simple BN)

Error correction coding

Biological and medical research

Sensor and information fusion

Probabilistic risk assessment
  Fault trees

Natural language understanding

Intelligent tutoring systems

...
GN&C - Structure and Operation

Task: from attitude (0,0,0) obtain and keep attitude (0.1,0.05,0.02)

Use: the given set of control thrusters
Demonstration Architecture

Architecture for running the demonstration software with and without injected failures and gather information from HW and SW sensors for processing by SSHM reasoner.
System Health Management using Bayesian Networks: Electrical Power Systems (Part II)
Fault Types - Part II

Independent faults
- Abrupt
  - Persistent
    - Discrete
    - Continuous (parametric)
  - Intermittent
- Incipient

Dependent faults
- Common cause
- Cascading

Diagnostics Problem I (DP-I) and Diagnostics Problem II (DP-II) of DXC-10 diagnostic competition

Bayesian networks in general
Persistent and Intermittent Faults

Electrical power system ADAPT

Two configurations:
  - DP-I - Single-string ADAPT
  - DP-II - Redundant ADAPT

Example fault types:
  1. abrupt intermittent
  2. abrupt persistent
     a) offset
     b) stuck

See paper “Second International Diagnostic Competition – DXC-10” by Poll et al. for further information about fault types.
CUSUM - Continuous Faults (1)

See [Ricks & Mengshoel, 2009] for details.
$CS(t) = S(t) - \left( w_0 S(t) + w_1 S(t - 1) + w_2 S(t - 2) + w_3 S(t - 3) \right) + CS(t - 1)$

$CS(t) = S(t) - \left( 0.45 \times S(t) + 0.25 \times S(t - 1) + 0.24 \times S(t - 2) + 0.06 \times S(t - 3) \right) + CS(t - 1)$
Intermittent Faults

• Intermittent faults follow a square wave pattern

- When there is faulty behavior, amplitude can vary
- Duration of faulty condition can vary
- Duration of nominal condition (between two faulty conditions) can vary
- Noise is typically present during both nominal and faulty conditions
- There may be “blips” or transients – short bursts that are not considered intermittent faults
Transitions between faulty and nominal conditions must occur within the boundaries set by the lower and upper thresholds.
Intermittent Faults - Algorithm

• ProDiagnose's tracking algorithm, Count, is invoked under the following conditions:
  – An intermittent fault has not been established for a health node $H$.
  – A persistent fault that has a mapping to an intermittent fault has been seen by ProDiagnose for health node $H$.
  – Initially Count will be in the high end of the square wave.

• If persistent fault does not reach $T_{LF}$ or persists past $T_{UF}$:
  – Count resets tracking

• If persistent fault diagnosis disappears before reaching $T_{UF}$:
  – Considered a high to low transition in the square wave
  – Count continues to track fault.
  – If persistent fault reappears between $T_{LN}$ and $T_{UN}$:
    • Count continues to track fault.

• If sufficiently many cycles are tracked sequentially, then fault is considered to be intermittent, and the node $I$ is clamped.
When the Count algorithm determines that a faulty state is most likely intermittent, an intermittent node \( I \) in the Bayesian network structure for a sensor or component will be clamped to an intermittent state.

- In MPE, this may (internal diagnosis) or for sure (external diagnosis) change the state of health node \( H \) to intermittent faulty.
DXC-10 - Experimental Data

Diagnostic problem I (DP-I) represents a single-string EPS in an unmanned aircraft system (UAS):

- EPS component types: Battery, circuit breaker, inverter, load (fan, AC load, DC load), relay, sensor (position, current, temperature, voltage)
- Fault types:
  - Battery, breaker, inverter, relay: degraded, failed open, failed off, stuck open
  - Load faults: over speed, under speed, resistance offset, resistance drift, *intermittent resistance offset*, failed off
  - Sensor faults: offset, stuck, drift, *intermittent offset*

Diagnostic problem II (DP-II) represents a redundant EPS:

- EPS component types and fault types: Similar to DXC-09 Tier 2

Experiments reported here use DXC-10 training set only

Diagnostic algorithm provides decision support

We only present diagnostic and not recommendation metrics

See [Poll et al., 2010] for further discussions of DXC metrics and benchmarking.
Bayesian Network for DXC-10 DP-I

The Bayesian network model of ADAPT DP-I
## ProDiagnose Results: DP-I

<table>
<thead>
<tr>
<th>ProDiagnose Configuration</th>
<th>Classification errors</th>
<th>Intermittent Classification Errors</th>
<th>Detection time (ms)</th>
<th>Isolation time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ProDiagnoseI</strong></td>
<td>65</td>
<td>18</td>
<td>17969</td>
<td>85444</td>
</tr>
<tr>
<td><strong>ProDiagnoseII</strong></td>
<td>57</td>
<td>8</td>
<td>17969</td>
<td>88008</td>
</tr>
<tr>
<td><strong>ProDiagnoseIII</strong></td>
<td>55</td>
<td>6</td>
<td>17970</td>
<td>83190</td>
</tr>
<tr>
<td><strong>ProDiagnoseIV</strong></td>
<td>53</td>
<td>4</td>
<td>17969</td>
<td>72266</td>
</tr>
<tr>
<td><strong>ProDiagnoseV</strong></td>
<td>53</td>
<td>4</td>
<td>17970</td>
<td>72266</td>
</tr>
</tbody>
</table>

Five configuration of ProDiagnose:

- **ProDiagnoseI**: no intermittent fault detection
- **ProDiagnoseII**: minimize intermittent false positives
- **ProDiagnoseIII**: higher sequential cycle count before setting intermittent state
- **ProDiagnoseIV**: looser thresholds for $T_{LF}$, $T_{UF}$, $T_{LN}$ and $T_{UN}$
- **ProDiagnoseV**: aggressive intermittent detection (used in DXC-10)
ProDiagnose Results: DP-I and DP-II

1. **Mean Time To Isolate**: Time from the beginning of a fault injection to the start of the last persistent “high” isolation signal.
2. **Mean Time To Detect**: Time from the beginning of a fault injection to the moment of the first “high” detection signal.
3. **Mean CPU Time**: Average CPU load during an experiment, averaged over all experiments.
4. **Classification errors**: Hamming distance between true component mode vector and the diagnostic algorithm’s component mode vector.
5. **False Positives Rate (FPR)**: The ratio of experiments where a fault is announced by the DA while the system was actually non-faulty, or where a fault is announced too early.
6. **False Negatives Rate (FNR)**: The ratio of experiments where a fault is missed while the system was actually faulty.
7. **Mean Peak Memory Usage**: The maximum memory size at every step in an experiment, averaged over all experiments.
8. **Detection Accuracy**: The ratio of correctly classified experiments (scenarios) to the total number of experiments.

<table>
<thead>
<tr>
<th>Metric</th>
<th>DP-I</th>
<th>DP-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Time To Isolate</td>
<td>72.3 sec</td>
<td>40.8 sec</td>
</tr>
<tr>
<td>Mean Time To Detect</td>
<td>18.0 sec</td>
<td>5.6 sec</td>
</tr>
<tr>
<td>Mean CPU Time</td>
<td>7 sec</td>
<td>13 sec</td>
</tr>
<tr>
<td>Classification Errors</td>
<td>53</td>
<td>23</td>
</tr>
<tr>
<td>False Positives Rate</td>
<td>0%</td>
<td>3%</td>
</tr>
<tr>
<td>False Negatives Rate</td>
<td>8.8%</td>
<td>11.5%</td>
</tr>
<tr>
<td>Mean Peak Memory Usage</td>
<td>3.3 MB</td>
<td>7.1 MB</td>
</tr>
<tr>
<td>Detection Accuracy</td>
<td>92.3%</td>
<td>87.9%</td>
</tr>
</tbody>
</table>

DP-II is using ADAPT in a redundant UAS setting.

Evaluation of the Arithmetic Circuit takes on the order of 1 ms [Mengshoel et al., 2008; Mengshoel et al., 2010]. The longer isolation and detection times above are due to our desire to have low FPR and FNR.