Using Expert Systems to Manage Diverse Networks & Systems

With a Focus on Operations

Greg Stanley


Contact the author at
http://www.gregstanleyandassociates.com/contactinfo/contactinfo.htm
Using Expert Systems to Manage Diverse Networks & Systems

I. Overview

II. Representation of networks & applications

III. Architectures

IV. Case studies
I. Overview

• Managing diverse networks
• Major operational goals
• Major components
• Alarm filtering & correlation examples
Diverse networks & systems

• Numerous device types & manufacturers
  Circuit switching/packet switching hardware

• Data vs. real-time for voice & video
  (it's not all ATM yet...)

• Different protocols (TCP/IP, CMIP, ...)
  Connection-oriented vs. connectionless

• LAN/WAN differences

• Wireless vs. terrestrial

• Changing topologies
  Portable computers, wireless, low-earth-orbit satellites)

• Complex devices
  Sub-objects with one IP address

• Subsystem interfaces & proxies
  Element management systems
Diverse enterprise network systems

Network, plus software processes and overall applications need to be managed

• Software processes & overall application
  • New Client/Server applications
  • Applications, including legacy
  • Services
  • Resources
    e.g., disk

Rapidly changing technology increases the need for flexible systems & rapid development
OSI Network management

Operations areas considered here

• Fault management
• Performance management

Other areas

• Security management
• Configuration management
• Accounting management
How does a real-time, object-oriented expert system help?

• Flexibility

• Speed of development - overall development environment

• Incremental development environment for rapid development & feedback, partial solutions

• Representation power: modelling the systems for use in diagnostics, analysis, prediction, ...

• Portability between platforms

• Systems integration capabilities
Some operations issues addressed by real-time, object-oriented expert systems

- Early detection of problems (proactive)
  *Predictions from performance or patterns*
- Alarm/message/event filtering
  *Suppression of repetitive alarms*
- Alarm correlation
  *Grouping of related alarms*
- Diagnosis
  *Pinpointing the causes of alarms*
- Procedure automation
  *Testing for diagnostic & filtering purposes*
  *Resolving problems*
  *Enforcing standard procedures*
  *Semi-automatic - guiding operator*
- Online information
  *Help, topology, hierarchy, relations*
- "What-if" simulations for analysis or training
Alarm/message filtering examples

• Alarm X occurs, then clears by itself within timeout. Suppress it (do not present to operator).

(Also log suppressed alarms for analysis)

• Alarm X occurs. Further testing reveals this alarm to be false or to have cleared itself. Suppress it.

• Alarm X is repeated n times. Present first alarm only, update a repetition counter

• Alarm X is not a real problem until it occurs n times within timeout. Present one alarm only, after n alarms, update repetition counter
Alarm correlation examples

• Alarm X and Alarm Y occur within timeout. Suppress these, present new message Z to operator.

• Alarms X1, X2, ..., X6, sent from different agents, are all complaining about "target" device Y. Acknowledge X1...X6, and send an alarm about Y.

• Alarms X1, X2, ..., X8 were all sent by "sender" device Y. Send an alarm indicating suspicious behavior of Y.
Model-based alarm correlation & diagnosis

Models are typically based on connectivity, part-of hierarchy, cause-effect failure models, individual device models such as state diagrams

• Multiple failures have occurred on the same LAN segment. Poll the remaining devices - if all fail, then warn the operator that that segment as a whole has failed (e.g., cable break), and acknowledge the individual source alarms

• Multiple devices X1, X2, ... are sending messages complaining that they cannot communicate with device Y. Send a message that device Y has failed, and acknowledge all the messages for X1, X2, ...

• High-level services requiring particular interface cards X1, X2, ... are all failing. X1, X2, ..., are all plugged into a common backplane or have some other common failure mode. Diagnose and alarm on the common mode failure, and acknowledge X1, X2, ... .
Procedure-based alarm correlation, diagnosis & resolution

• Alarm X occurs. Wait 60 seconds. Check for symptoms again by polling, or log in to a computer execute some UNIX commands (using remote shell). If the problem is still there, send an alarm, otherwise suppress it (except for an optional log entry).
II. Representation of networks as a basis for applications

• Knowledge management view

• "Build yourself a graphical language" to more closely match your tool to your domain

• Representation in OPA
Knowledge management view

- Emphasizes representation of knowledge for applications
- Not just data!
  - System hardware & software models
  - System topology
  - Failure & fault propagation models
  - Operating rules & procedures
- Example: alarm correlation & diagnostics need process topology and device models, also usable in operator training and in planning.
- Object-oriented, with graphical representation
Major characteristics of a KBES ("Knowledge-Based Expert System")

• KBES represents both qualitative and quantitative models

• Object orientation is the key part of modern expert systems

• KBES represent information explicitly, rather than embedded in code

analogy: simultaneous mathematical equations vs. set of assignment statements and iterative procedure in FORTRAN, or schematics rather than as a set of statements generating the schematic

• Emphasis on building "declarative" descriptions, independent of subsequent use, and easily inspectable by wide class of users

- Goal to simplify representation & re-use of knowledge for multiple purposes

• Some KBES's (G2) have strong graphics orientation as part of its declarative knowledge

• Static vs. real-time KBES

• Development environment
• KBES provide powerful new high-level tools for modelling and re-use

• High-level descriptions:
  - Equipment class implies behavior
  - Schematic drawings: connections imply fault propagation, data flow, reachability, reliability
  - "Part-of" relation implies fault propagation model
  - "Is-a-kind-of" specialization simplifies descriptions

  *all modems share some common properties*

  *reachability analysis ignores differences between most devices, and may include software processes*

  - Generic statements utilize these high-level constructs to generate specific diagnosis or simulation, using common attributes

• Model declarations are independent of ultimate usage

• Qualitative models (e.g., cause-effect)
• Portion of a class hierarchy

- LOGICAL-UNIT, an object-definition
  - BUS-NODE, an object-definition
  - COMM-TWO-PORT, an object-definition
    - CLUSTER-CONTROL, an object-definition
      - SERVER, an object-definition
        - COMPUTER, an object-definition
      - SMALL-CLUSTER-CONTROL, an object-definition
    - MODEM, an object-definition
      - PROTOCOL-CONVERTER, an object-definition
        - BRIDGE, an object-definition
        - REPEATER, an object-definition
        - GATEWAY, an object-definition
        - ROUTER, an object-definition
        - TRANS-LAN, an object-definition
    - TOKEN-RING-REPEATER, an object-definition
      - LAN-TRANSCEIVER, an object-definition
      - CLUSTER-CONTROL-EXT, an object-definition
      - IBM-CHANNEL-CONTROLLER, an object-definition
      - Q-BUS-NODE, an object-definition
    - MEDIUM-COMPUTER, an object-definition
      - SMALL-COMPUTER, an object-definition
      - BIG-COMPUTER, an object-definition
      - WORKSTATION, an object-definition
    - TERMINAL-SERVER, an object-definition
Portions of class hierarchy (indented form)

- TELECOM-DEVICE
  - ELECTRONIC-DEVICE
    - LOGICAL-UNIT
      - BUS-NODE -- 1 instance
        - TOKEN-RING-REPEATER -- 1 instance
        - LAN-TRANSCEIVER -- 1 instance
        - ETHERNET-TRANSCEIVER -- 48 instances
      - CLUSTER-CONTROL-EXT -- 1 instance
      - IBM-CHANNEL-CONTROLLER -- 4 instances
      - Q-BUS-NODE -- 3 instances
        - Q-BUS-RS-232-NODE -- 1 instance
      - HUB -- 1 instance
      - COMM-TWO-PORT
        - MODEM
          - REMOTE-LOOPBACK-MODEM -- 7 instances
            - REMOTE-LOOPBACK-MODEM-RS-232 -- 11 instances
          - MANUAL-LOOPBACK-MODEM -- 1 instance
            - MANUAL-LOOPBACK-MODEM-RS-232 -- 5 instances
          - IN-HOUSE-MODEM -- 1 instance
            - IN-HOUSE-MODEM-RS-232 -- 1 instance
          - MODEM-NO-LOOPBACK -- 1 instance
            - MODEM-NO-LOOPBACK-RS-232 -- 1 instance
          - PROTOCOL-CONVERTER -- 4 instances
          - BRIDGE -- 1 instance
          - REPEATER -- 3 instances
          - GATEWAY -- 1 instance
          - ROUTER -- 2 instances
          - TRANS-LAN -- 3 instances
        - CLUSTER-CONTROL
          - SMALL-CLUSTER-CONTROL
            - IBM-3274-CLUSTER-CONTROLLER -- 2 instances
        - SERVER
          - TERMINAL-SERVER
            - RS-232-TERMINAL-SERVER -- 4 instances
        - COMPUTER
          - MEDIUM-COMPUTER -- 5 instances
          - SMALL-COMPUTER -- 11 instances
          - BIG-COMPUTER -- 2 instances
          - WORKSTATION -- 27 instances
          - GMS-NODE -- 124 instances
        - COMPUTER-PERIPHERAL-DEVICE
          - TERMINAL -- 4 instances
            - RS-232-TERMINAL -- 8 instances
          - PRINTER -- 4 instances
Sample class definitions

△ MODEM

△ MANUAL-LOOPBACK-MODEM

△ MANUAL-LOOPBACK-MODEM-RS-232

△ MODEM-NO-LOOPBACK

△ MODEM-NO-LOOPBACK-RS-232

△ REMOTE-LOOPBACK-MODEM

△ REMOTE-LOOPBACK-MODEM-RS-232
<table>
<thead>
<tr>
<th><strong>REMOTE-LOOPBACK-MODEM, an object-definition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class name</strong></td>
</tr>
<tr>
<td><strong>Superior class</strong></td>
</tr>
<tr>
<td><strong>Attributes specific to class</strong></td>
</tr>
<tr>
<td><strong>Capabilities and restrictions</strong></td>
</tr>
<tr>
<td><strong>Class restrictions</strong></td>
</tr>
<tr>
<td><strong>Change</strong></td>
</tr>
<tr>
<td><strong>Menu option</strong></td>
</tr>
<tr>
<td><strong>Inherited attributes</strong></td>
</tr>
<tr>
<td><strong>Default settings</strong></td>
</tr>
<tr>
<td><strong>Attribute displays</strong></td>
</tr>
<tr>
<td><strong>Stubs</strong></td>
</tr>
</tbody>
</table>
Using the editor to change the stubs for a class

<table>
<thead>
<tr>
<th>Cancel</th>
<th>a wire wire-port-in located at a wire wire-port-in located at a wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>top</td>
</tr>
<tr>
<td></td>
<td>bottom</td>
</tr>
<tr>
<td></td>
<td>right</td>
</tr>
<tr>
<td></td>
<td>left</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>unacknowledged; test-state is no-test; time-last-proven-good is 0; user-entered-failure-status is ok; distance-from-g2 is 0; address is &quot;&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default settings</td>
<td>none</td>
</tr>
<tr>
<td>Attribute displays</td>
<td>inherited</td>
</tr>
<tr>
<td></td>
<td>a wire wire-port-in located at left 10 with style diagonal; a phone-connection wire-port-out located at a phone-connection wire-port-out located at</td>
</tr>
</tbody>
</table>
Example: Icon editor

- Region: modem-region
- Color: transparent
- Width: 45
- Height: 25
- x1, x2, x3
- Acknowledgment
- Purple
- modem-region
- alarm-region

Layer-color:
- Transparent
- Foreground:
  - Black
  - Gray
  - Pink
  - Brown
  - Tan
  - Sienna
  - Khaki
  - Green
  - Lime green
  - Medium aquamarine
  - Cadet blue
  - Slate blue
  - Medium orchid
  - Plum
  - Magenta
- Pale green
- Forest green
- Green
- Aquamarine
- Light blue
- Turquoise
- Cyan
- Sky blue
- Medium blue
- Blue
- Dark slate blue
- Thistle
- Purple
- Violet
- Maroon
- Violet red
Example relation used in diagnosis

<table>
<thead>
<tr>
<th>A-POSSIBLE-CAUSE-FOR, a relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes</td>
</tr>
<tr>
<td>User restrictions</td>
</tr>
<tr>
<td>First class</td>
</tr>
<tr>
<td>Second class</td>
</tr>
<tr>
<td>Relation name</td>
</tr>
<tr>
<td>Inverse of relation</td>
</tr>
<tr>
<td>Type of relation</td>
</tr>
<tr>
<td>Relation is symmetric?</td>
</tr>
</tbody>
</table>

- an event may be a-possible-cause-for more than one message;
- a message may be possibly-caused-by more than one event.
Example generic rule using connectivity

For any telecom-device D connected to any hub H
if the status of H is failed
then conclude that the status of D is failed
and conclude that the alarm-priority of D = the alarm-priority of H

(actual syntax)
Example generic rule used in alarm filtering

For any electrical-device $D$
whenever any message $MSG$ becomes an-event-for $D$
and when the count of each message $MSG2$ that is an-event-for $D > 4$
then conclude that ....
and start multiple-message-filter($D$)

*(actual syntax)*
Some benefits of KBES representation

• Reduces gaps between system analysis, specification, design, implementation, run-time use, maintenance.
  - Explicit models carried through all phases
  - Inspectable by all classes of users, not just programmers

• Common representation for multiple applications, with one consistent model for development & maintenance

• Generic library: default behavior specified for given class of object, connections - no additional special lists to fill out unless object deviates from the defaults
Some features of G2 - the graphically-oriented, real-time Knowledge-Based Expert System (KBES)

- Objects with attributes
- Class hierarchy for objects, with inheritance of properties and behavior - allowing "differential modelling"
- Associative knowledge, relating objects in the form of connections and relations
- Structural knowledge (e.g., "part-of" relation)
- Representation and manipulation of objects and connections graphically
- Generic rules and associated inference engine
- Concurrent procedures
- Analytic knowledge, such as functions, formulas, differential equation simulation
- Real-time task scheduler, supporting concurrency, priorities, time stamping, validity intervals, timed actions, event-driven activity, reasoning within a fixed deadline, history-keeping, data interfaces
• Interactive development environment and run-time environment

• Graphics

• External interfaces for systems integration
An option: "Build yourself a graphical language"

- Match tool to domain - reduce semantic gap between tool and problem
- Build library of classes & methods (procedures), rules, etc.
- Build "configurer" GUI based on cloning objects from a palette, connecting them, filling out tables of attributes
- Fairly common in many domains
Common graphical elements

• Containment hierarchy / "part-of" for physical areas, common-modes, physical equipment, hierarchy

• Objects in a class hierarchy with specialization & inheritance.

  Workstation is a-kind-of computer

  Abstract classes such as "hardware"

• Objects include attributes and methods (procedures), e.g., test methods

• Almost everything, whether physical or abstract, is an object

• Graphical connections represent physical connectivity, logical connectivity, or relationships such as cause/effect, hierarchy
RTES-based Petri net example

The language

- Petri net represents actions & state transitions
- Procedures executed at each node
- "Token" passed among nodes, split when parallel operations are launched
- Explicit concurrency control
  
  e.g., "Rendezvous" to re-unite concurrent operations
- Used in control & other applications, to execute sequential, procedural operations

The RTES/Object-oriented implementation

- Objects represent nodes, rendezvous, token
- Methods (procedures) called at each node, using underlying implementation language
- Connections (objects) for transitions
- Rules or procedures watch for state transitions
RTES-based state diagram example

The language

- Diagram represents states & state transitions
- Procedures executed at each node
- "Token" passed among nodes

The RTES/Object-oriented implementation

- Objects represent nodes, token
- Connections (objects) for transitions
- Rules or procedures watch for state transitions

Implementation simpler than Petri net, similar
Other common graphical approaches

• Logic networks (AND/OR gates, etc.)
  Input symptoms, output causes
  Roughly equivalent to specific rules

• Fault trees, decision trees, AND/OR trees, hierarchical fault models, with goal-seeking
  Similar objects, different program control

• Cause/effect diagrams

• Procedures to analyze schematic/map
Representation in OPA

- Telecom devices and software processes
  includes the "managed objects"
- Class hierarchy
- Workspace ("part-of", "containment") hierarchy
- Containers ("sites", "networks")
  alarm and acknowledgement status is propagated up the containment hierarchy
- Alarms/messages/events
- Relations
- Connections - topology information
- Test and operator actions representation
- Common framework shared between message-handling, OPAC graphics language, and schematics
Example palettes for telecom-devices

NETWORK EQ.  
Select an object to clone it.
Bridges, etc.  Bus nodes
Protocol Converter  Generic Node
               Lan transceiver
Bridge  Ethernet transceiver
Repeater
       Token ring repeater
Trans Lan
       Hub
Gateway
Router
       Cluster Control Extension O-Bus IBM Channel Controller
Remote Links
Phone Link
Point-to-point
Phone Wire Link

Hide
Example attribute table: telecom-device

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>On/off</td>
<td>1</td>
</tr>
<tr>
<td>In service</td>
<td>1</td>
</tr>
<tr>
<td>What if in service</td>
<td>1</td>
</tr>
<tr>
<td>Highest message priority</td>
<td>99999</td>
</tr>
<tr>
<td>Acknowledgement status</td>
<td>acknowledged</td>
</tr>
<tr>
<td>Test state</td>
<td>no-test</td>
</tr>
<tr>
<td>Time last proven good</td>
<td>4618</td>
</tr>
<tr>
<td>User entered failure status</td>
<td>ok</td>
</tr>
<tr>
<td>Distance from g2</td>
<td>3</td>
</tr>
<tr>
<td>Polling method</td>
<td>ping</td>
</tr>
<tr>
<td>Test method</td>
<td>no-method</td>
</tr>
<tr>
<td>Address</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Agent</td>
<td>snmp</td>
</tr>
</tbody>
</table>
## Container configuration palette

### Containers

Select an object to clone it.

<table>
<thead>
<tr>
<th>Locations</th>
<th>Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site</td>
<td>(Generic) Internet</td>
</tr>
<tr>
<td>City</td>
<td>(Generic) network</td>
</tr>
<tr>
<td>Room</td>
<td>(Generic) LAN</td>
</tr>
<tr>
<td>Floor</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Geograph.</td>
<td>Ethernet segment</td>
</tr>
<tr>
<td>USA States:</td>
<td>Token-ring LAN</td>
</tr>
<tr>
<td></td>
<td>Token-ring segment</td>
</tr>
<tr>
<td></td>
<td>X.25 Network</td>
</tr>
</tbody>
</table>

---

Hide
Workspace (map) hierarchy I
Workspace (map) hierarchy II
### A site

<table>
<thead>
<tr>
<th><strong>NJ, a northeast-states-region</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Notes</strong></td>
</tr>
<tr>
<td><strong>User restrictions</strong></td>
</tr>
<tr>
<td><strong>Names</strong></td>
</tr>
<tr>
<td><strong>Highest message priority</strong></td>
</tr>
<tr>
<td><strong>Acknowledgement status</strong></td>
</tr>
</tbody>
</table>
Processing for incoming events

• Decode messages as needed, including identification of target, sender, category
• Eliminate obvious repetitions by simple message filtering
• Create "raw" warning messages
• Apply model-based diagnosis, heuristics, procedural reasoning when possible
  • Acquire additional information & run tests
  • Select candidate "most likely" failures based on model or other information
  • Draw conclusions about root causes and sympathy events, prove nodes "good" or "bad"
• Cluster remaining alarms into reasonable groups when possible
• Automatically fix problems where possible
• Notify the operator with summarized alarms and other alarms, guide through repairs
• Pass information to trouble ticket system
• Recognizing recurring problems & notify system administrator
## Sample filtered message

<table>
<thead>
<tr>
<th>Notes</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>User restrictions</td>
<td>none</td>
</tr>
<tr>
<td>Names</td>
<td>none</td>
</tr>
</tbody>
</table>

**Additional text**

A REMOTE-LOOPBACK-MODEM connected to TERM-SERV-B1 failed remote loopback. Caused 3 sympathy events.

**Sender**

"G2-DIAGNOSTIC-SYSTEM-1"

**Target**

"A REMOTE-LOOPBACK-MODEM connected to TERM-SERV-B1"

**Message id**

-99999

**Creation time**

20 Nov 93 4:02:51 p.m.

**Revisit time**

22 Aug 98 6:31:14 a.m.

**Revisit method**

no-method

**Deletion time**

20 Nov 93 5:03:15 p.m.

**Priority**

2

**Acknowledgement status**

unacknowledged

**Acknowledgee**

none-yet

**User comment**

""

**Sympathy events for:** BLOB, A REMOTE-LOOPBACK-MODEM on the subworkspace of GOTHAM-CITY, A REMOTE-LOOPBACK-MODEM connected to BLOB.""
Filtered messages

The main message sent would be the ones on the filtered-message handler. The above message shows up in summary form on the message handler as:

Filtered warnings

<table>
<thead>
<tr>
<th>A REMOTE-LOOPBACK-MODEM connected to TERM-SERV-B1 failed remote loopback. Caused 3 sympathy events.</th>
</tr>
</thead>
<tbody>
<tr>
<td>At least one failure among: LINE-B1. Caused 1 sympathy event.</td>
</tr>
</tbody>
</table>
## G2-manager-process

<table>
<thead>
<tr>
<th>INMS-G2-MANAGER, a g2-manager-process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes</td>
</tr>
<tr>
<td>User restrictions</td>
</tr>
<tr>
<td>Names</td>
</tr>
<tr>
<td>Model update interval</td>
</tr>
<tr>
<td>Routine pinging interval</td>
</tr>
<tr>
<td>Ping timeout interval</td>
</tr>
<tr>
<td>Polling interface</td>
</tr>
<tr>
<td>Suspect list</td>
</tr>
<tr>
<td>Proven good list</td>
</tr>
<tr>
<td>Failed list</td>
</tr>
<tr>
<td>Degraded list</td>
</tr>
<tr>
<td>Hypothetical model reachable list</td>
</tr>
<tr>
<td>Hypothetical model unreachable list</td>
</tr>
<tr>
<td>Model reachable list</td>
</tr>
<tr>
<td>Model unreachable list</td>
</tr>
<tr>
<td>Outstanding ping list</td>
</tr>
<tr>
<td>Failed ping list</td>
</tr>
</tbody>
</table>
FEP-3 failed a ping
FEP-2 failed a ping
FEP-1 failed a ping
BACKPLANE-1 failed a ping
BACKPLANE-1 failed a ping. Caused 4 sympathy events.
BLOB failed a ping

A REMOTE-LOOPBACK-MODEM connected to TERM-SERV-B1 failed remote loopback. Caused 3 sympathy events.

At least one failure among: LINE-B1. Caused 1 sympathy event.

CRT-B3 failed a ping
MOD-B4

State Control

♦ Out of service test
✓ Out of service
✓ Reportedly not working
✓ Unit is OK
Example filtering scenario: raw messages

Instead of sending all these failure messages to the operator, the following filtered message would be sent, as shown on the "filtered messages" handler:
Filtered version of the previous 22 messages, sent to operator

Filtered warnings

M4-TX failed a ping. Caused 21 sympathy events.

The details of this message show the original information that went into this summarized message. The additional-text explanation is assembled automatically.
Ma-TX failed a ping. Caused 21 sympathy events.

**sender** INMS-GZ-MANAGER

**target** M4-TX

**message id** 130

**repetitions** 0

**creation time** 14 Mar 94 3:13:30 p.m.

**revisit time** 19 Aug 94 9:14:25 a.m.

**revisit method** no-method

**deletion time** 26 Mar 94 3:05:03 a.m.

**priority** 1

**acknowledgement status** unacknowledged

**acknowledger** none-yet

**additional text**

Ma-TX failed a ping. Adjacent node DK-MM is good.

Sympathy events for: TXAAF, TXBDF, TXXBF, TXBCF, TXAUF, TXAYF, TXAWF, TXAKF, TXAYF, TXBHF, TXAZF, TXBF, TXAUF, TXBU, TXBU, TXCU, TXABF, TXACF, TXADF, TXBFF, TXBEF, TXBDF.

Original messages now automatically acknowledged:

Ma-TX failed a ping
TXAAF failed a ping
TXBDF failed a ping
TXXBF failed a ping
TXBCF failed a ping
TXAUF failed a ping
TXAYF failed a ping
TXAWF failed a ping
TXAKF failed a ping
TXAYF failed a ping
TXBHF failed a ping
TXAZF failed a ping
TXBF failed a ping
TXAUF failed a ping
TXBU failed a ping
TXBU failed a ping
TXCU failed a ping
TXABF failed a ping
TXACF failed a ping
TXADF failed a ping
TXBFF failed a ping
TXBEF failed a ping
TXBDF failed a ping

**user comment** **
Decisions palette

Decisions
(default = manual)

User-entered expression, with example
"$stack == 1"

Decisions
(Special Cases)
(You can write your own decision-procedure)

opac-match-substring
"pattern"

opac-exact-string-match
"pattern"

consume-decision-from-stack

opac-if-token-error-free
UNIX actions palette

Read, Write file

Spawn process

Kill process

Delete file

File exists test

Special cases of general procedures

find-pid-from-ps pid
Misc. actions palette

Stack operations
- opac-put-text-on-stack-proc
- Pop general stack and delete
- opac-pop-general-stack
- opac-put-connected-objects-on-stack
- opac-snmp-get1-proc
  "1.3.6.1.2.1.1.1.0"
  4

Local parameters
- Text local-name
  "" (initial value)
- Integer local-name
  0 (initial value)
- Local object-name
  local-name
  ""

Debugging tools
- opac-show-token-info-

Generic blocks requiring user-developed G2 procedure.
- user-proc-to-set
  local-name
  send-message-user-de
  0
  ""
- opac-send-smh-msg
  ""
- put-something-on-stack

Other
- (Connection post)
The name of this start block is the subtask, which by name, multiple times, concurrently, from multiple
started by G2 program, by another diagram, or by

Showed this workspace on the
window of the caller.

Shows help workspace
on the window of the caller.

The "pause" capability is attached as a "stop-signal".
This pauses, generating a dialog for the user to do,
but times out as shown at 20 seconds.

Hides most recently-displayed
workspace as default.

This block calls a subtask, which has its own graph
procedure. Many of these blocks can call the same.
The displayed attribute points to the task name.

This "general procedure" block executes a G2 plug-in
by name (the displayed attribute). This one calls
"opac-do-nothing", just sending a message.

This "decision proc" calls a procedure to decide
path to take. Here, the user is prompted to make
choice, but any procedure could be called.

If chosen, the same timer demo is called again.
The endless loop was started for "target" P6S04. The menu above was generated automatically by the decision block, which had the following table. Note the use of variables, indicated with the $.

<table>
<thead>
<tr>
<th>Control proc</th>
<th>opac-default-control-proc-for-decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision proc</td>
<td>opac-manual-decision-proc</td>
</tr>
<tr>
<td>Timeout</td>
<td>60</td>
</tr>
<tr>
<td>Default decision</td>
<td>1</td>
</tr>
<tr>
<td>Header text</td>
<td>&quot;In task $task, for target= $target, select choice:&quot;</td>
</tr>
<tr>
<td>Choice 1 description</td>
<td>&quot;Ready to begin demo cycle&quot;</td>
</tr>
<tr>
<td>Choice 2 description</td>
<td>&quot;Not ready to start demo cycle&quot;</td>
</tr>
</tbody>
</table>
Block pause capability

Pause in task OPA-DEMO-ACTION-SEQUENCE, for target=P6S04. Select choice:

Continue with next step, A OPAC-HIDE-WORKSPACE on the subworkspace of SAMPLE-ACTION-DIAGRAMS

Abort this task

Done

The menu above was generated automatically by the block-pause-capability, which had the following table. Note the use of variables, indicated with the $.

<table>
<thead>
<tr>
<th>an opac-block-pause-capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control proc</td>
</tr>
<tr>
<td>Header text</td>
</tr>
<tr>
<td>Continue text</td>
</tr>
<tr>
<td>Timeout</td>
</tr>
</tbody>
</table>
OPAC-DEMO-PARSING

default

opac-put-text-on-stack-proc
"$target FAILED due to POWER LOSS"

opac-show-stack-top

opac-match-substring

"OK"

opac-match-substring

"POWER LOSS"

exact-string-match

"A different string"

"Try again. (In developer mode, enter a different text string)"

Note the use of $ in strings for substitution of commonly-
e.g., target object. You can also use $ for local variables the $ inside of the pattern text for the match-substring ar
match. See the demo on local parameters, coming next.
The local parameters are "superior" and "normal-val". "Normal-val" is initialized to the value "OK". They on within this procedure call

```
set-parm-to-superior-of-target
superior
```

```
opac-put-text-on-stack-proc
"$target FAILED due to POWER LOSS.
Check superior ($superior) for problems."
```

```
opac-show-token-info-proc
opac-show-stack-top
```

```
opac-match-substring
"$normal-val"
```

```
opac-match-substring
"POWER LOSS"
```

```
opac-
"A diff
```
Alarm Thresholding Example

In this ex: If a 'Noisy' msg has been received 3 or more times against this target in the last 5 minutes, then a new message 'Serious noise problem' will be posted.

For Demonstration:

Clear History

<table>
<thead>
<tr>
<th>Database</th>
<th>smh-history</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results destination</td>
<td>stack</td>
</tr>
<tr>
<td>Query class</td>
<td>smh-history</td>
</tr>
<tr>
<td>Pattern</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Results as count only</td>
<td>yes</td>
</tr>
<tr>
<td>Match time end</td>
<td>current-time</td>
</tr>
<tr>
<td>Match time interval minutes</td>
<td>5</td>
</tr>
<tr>
<td>Target</td>
<td>&quot;threshold-testobj-1 &quot;</td>
</tr>
<tr>
<td>Sender</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Message category</td>
<td>&quot;Noisy&quot;</td>
</tr>
</tbody>
</table>
Alarm Filtering / Diagnosis Examples

Network traps / messages can be diagnosed & filtered according to almost any criteria, and using a variety of OPAC tools. This shows a few examples.

1. **Simulate Trap from Netwk**
   - ALMS800-ACTION
   - put-message-in-local-parm
   - mag
   - default
   - FILTERING-TESTOBJ-1

2. **Check for a different alarm.**
   - Target "$target"
   - Message category "ALM64"
   - Match time interval minutes 2
   - "$stack == 1"
   - send-user-defined-hyp-msg 2
     * "OPA. Confirmed: $mag"
   - Filter based on results from Unix command.
   - Filter if no other alarm against the same object.

3. **Further (external) tests to see if alarm still exists.**
   - "ps -ef | wc -l"
   - Execute Unix cmda and return results.

4. **Alarm may clear itself within timeout.**
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>smh-history</td>
</tr>
<tr>
<td>Results destination</td>
<td>stack</td>
</tr>
<tr>
<td>Query class</td>
<td>smh-history</td>
</tr>
<tr>
<td>Pattern</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Results as count only</td>
<td>yes</td>
</tr>
<tr>
<td>Match time end</td>
<td>current-time</td>
</tr>
<tr>
<td>Match time interval minutes</td>
<td>2</td>
</tr>
<tr>
<td>Target</td>
<td>&quot;$target&quot;</td>
</tr>
<tr>
<td>Sender</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Message category</td>
<td>&quot;ALM*&quot;</td>
</tr>
</tbody>
</table>
In this example: If 'Common Fail' msg has been received against all 3 targets within the last 2 minutes, then a correlation message will be posted against the Supervisor object.

1. START here.
2. `$stack > 0`?
   - If true, send 'Common Fail'
   - If false, continue.
3. `$stack = 0`?
   - If true, send 'Common Fail'
   - If false, continue.
4. `$stack = 0`?
   - If true, send 'Common Fail'
   - If false, send user-defined message.
5. Historical query specification.
6. Decision block allows a user-entered expression.

Message category 'Common'
Match time interval minutes 2

Do nothing.

Mag or Trap allows user-entered priority and text.
Correlate related alarms against connected objects. Connections may represent cause-effect relationships, hierarchy info, etc. SELECT model-based-corr FROM A CONNECTED NODE'S MENU TO START THE PROCEDURE.

**Model-Based Correlation Example**

- Correlate related alarms against connected objects. Connections may represent cause-effect relationships, hierarchy info, etc. SELECT model-based-corr FROM A CONNECTED NODE'S MENU TO START THE PROCEDURE.

- **default**

- **opaque-put-connected-objects-on-stack**
  - **"$target"** Objects connected to your selection.

- **opaque-put-connected-objects-on-stack**
  - **"$stack"** Objects connected to connected objects.

- **How many msgs containing a reference to 'Backplane' for these connected objects?**
  - **"$stack"** Message category "Backplane"
    - Match time interval minutes 3

- **$stack = 5**

- **Send-user-defined-hyp-mag**
  - 2 Connected nodes with a common backplane are getting multiple alarms. [corr-common-backplane]
III. OPA Architecture

• Overall Architecture/System Integration
• Major components
Overall Architecture
OPA Major building blocks

- User & Developer Interfaces
  - Schematic (Maps)
  - OPAC
    - Graphical action language
  - Message Handling & MIB (Storage)
- Data Interfaces (GSI...)

Message Management - a message "MIB"

- Messages are objects with attributes such as priority, acknowledgement status, time stamp, timeouts for procedure execution such as escalation, etc.

- Message handlers store messages

- Individual views or messages handlers can be set up per Telewindows user

- Messages are organized and related to "target", "sender", "ID" (category), window, etc., for analysis or browsing.

- Unified framework with OPAC, e.g., same "target", "sender", "ID" (OPAC uses "notify" to designate message-handler

- Messages determine the priority and acknowledgement status of objects in the schematic (map)

- Programmatic access, as well as access by users
User interactions with message handlers

• Acknowledgement

• Deletion

• Optional modification (e.g., comments)

• Navigation to find sender, target, etc.

• Navigation from schematic objects to browse messages at any level (object, or larger unit with a subworkspace hierarchy)
Systems Integration

- GSI C-based library to build custom bridges
  - GSI runs as separate process, across network
  - Asynchronous communications
  - Remote procedure calls
  - Polling or event-driven
- SQL-type interfaces to databases (Oracle, Sybase, ...)
- OpenView (SNMP/DM) interface
- File I/O and process spawning
OpenView bridge

- Interfaces G2/OPA to OpenView and general network via SNMP

- Runs as separate process, on same CPU as OpenView (G2 generally runs on a different machine)

- Written in C, using binaries from GSI library and OpenView library

- Works with OpenView DM platform, or the SNMP platform (which is a subset of DM)

- Supports standard SNMP get, get-next, set, send-trap, receive-trap

- With DM platform, can register for events using HP Event Management Services

- XMP calls (which support CMIP protocol) available later

- "Blocking" and "Non-blocking" modes
Using the OpenView bridge: mechanics

• G2/OPA can initiate interactions, or receive unsolicited traps

• G2/OPA can poll, or do management by exception

• G2/OPA can communicate directly with other managers, agents, or software (e.g., Bridgeway's Eventix) by getting and sending traps

• G2/OPA can change colors on OpenView map by sending standard "status" trap

• Operators using OpenView Windows can send traps directly to G2 (via "snmptrap" utility, after configuration of executable icons or menu entries)

• G2/OPA communications to or from bridge are G2 remote procedure calls
Using the OpenView bridge: strategies

• Might configure G2 as an "intelligent operator", or as an "intermediary", intercepting all alarms on the way to OpenView

• Might use polling or management by exception

  Polling can cause slow response, poor scaling

  Pure management by exception works better when messages have guaranteed delivery, but SNMP datagrams don't guarantee delivery

• Large distributed system may require some filtering, parsing & tokenization of alarm messages close to the sources

  May want "proxy" or other agents

  Future OPA versions may directly help generate intelligent agents when needed
File I/O and process spawning

• Typical need to launch UNIX processes, receive results, read and write files

  example - log in via remsh, do a "ps -ef | grep xxxxx" to find if a particular process is running, and interpret the results, possibly kill a process and start a process

  example - again via remsh, check if a file exists. If not, start some process. When the file exists, read its first line and take action based on that first line.

• OPAC language blocks directly execute spawns, file I/O
IV. Case studies using real-time expert systems

- AT&T EasyLink (Commercial electronic mail service): Using OPA for alarm filtering & diagnostics, procedure automation
- Intelsat: network monitoring, satellite telemetry monitoring
- Stanford Telecom ATM applications: DoD SSCN & SPANet, ANMA ATM manager
- Texaco Trading & Transportation
- SWIFT (Belgium) - monitoring bank wire transfers
- CRT Banca (Italy), others: Remote bank security monitoring
- Telefonica (Spain)