Using Expert Systems to Manage Diverse Networks & Systems

With a Focus on Operations

Greg Stanley

Presented as the keynote presentation at Enterprise Management Summit '94, in Conference Proceedings, Enterprise Management Summit '94,
Nov. 14-18, 1994, San Francisco, CA, USA, pp. 7-82.

Contact the author at
http://www.gregstanleyandassociates.com/contactinfo/contactinfo.htm

Using Expert Systems to Manage Diverse Networks & Systems

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

<u>Operations areas considered here</u>

- Fault management
- Performance management

<u>Other areas</u>

- 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

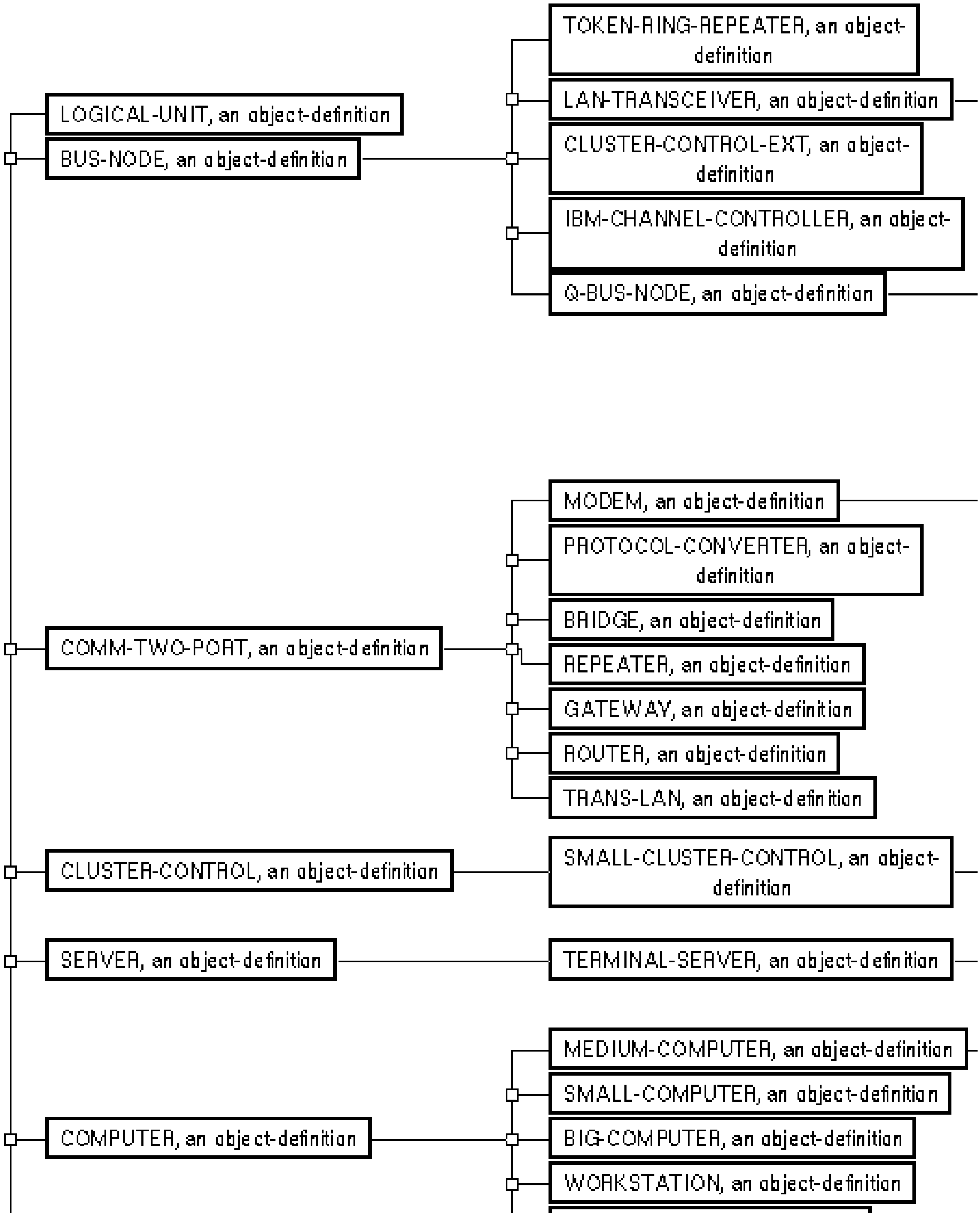
    reachability analysis ignores differences between most devices, and may include software processes

  - 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



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 -- 3 instances
| | | | | | | 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

REMOTE-LOOPBACK-MODEM, an object-definition	
Class name	remote-loopback-modem
Superior class	modem
Attributes specific to class	test-method is remote-loopback
Capabilities and restrictions	none
Class restrictions	none
Change	none
Menu option	a final menu choice
Inherited attributes	on-off is 1; polling-method is no-method; agent is no-agent; ext-name is ""; in-service is 1; what-if-in-service is 1; state is ""; highest-message-priority is 99999; acknowledgement-status is 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 ""
Default settings	none
Attribute displays	inherited
Stubs	a wire wire-port-in located at left 10 with style diagonal; a phone-connection wire-port-out located at

Using the editor to change the stubs for a class

Cancel

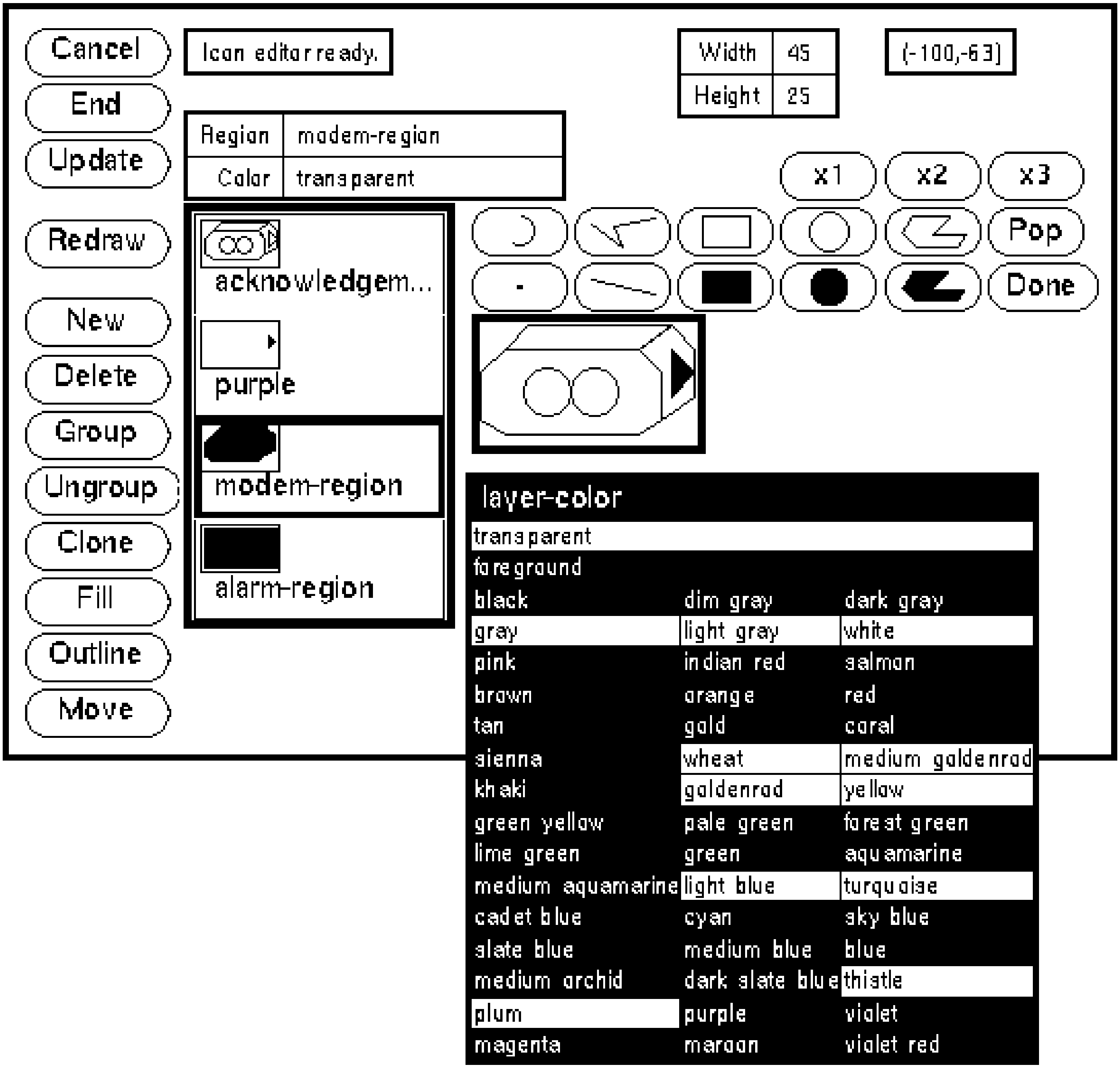
Undo

a wire wire-port-in located at

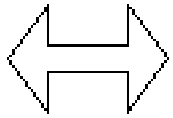
top
bottom
right
left

	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 ""
Default settings	none
Attribute displays	inherited
Stubs	a wire wire-port-in located at left 10 with style diagonal; a phone-connection wire-port-out located at

Example: Icon editor



Example relation used in diagnosis



A-POSSIBLE-CAUSE-FOR
possibly-caused-by

A-POSSIBLE-CAUSE-FOR, a relation	
Notes	OK
User restrictions	none
First class	event
Second class	message
Relation name	a-possible-cause-for
Inverse of relation	possibly-caused-by
Type of relation	many-to-many
Relation is symmetric?	no
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., "Rendevous" 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

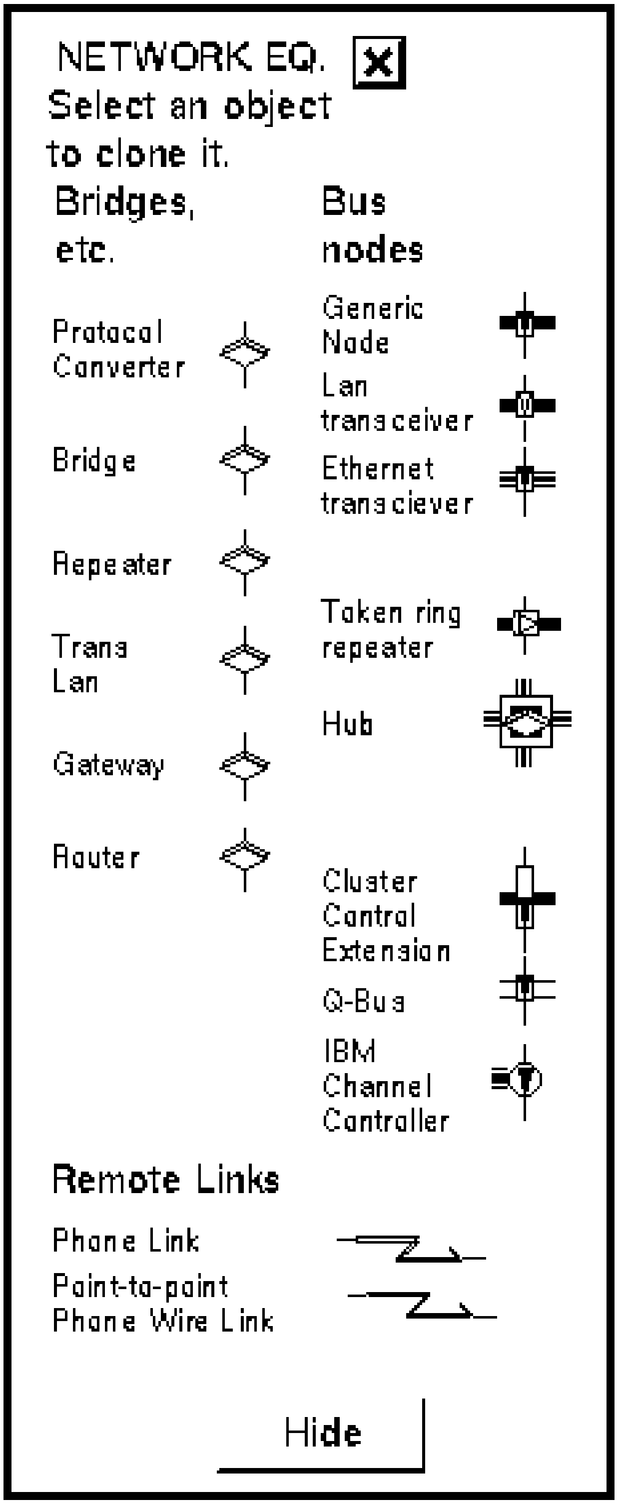
  includes the "managed objects"

- 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

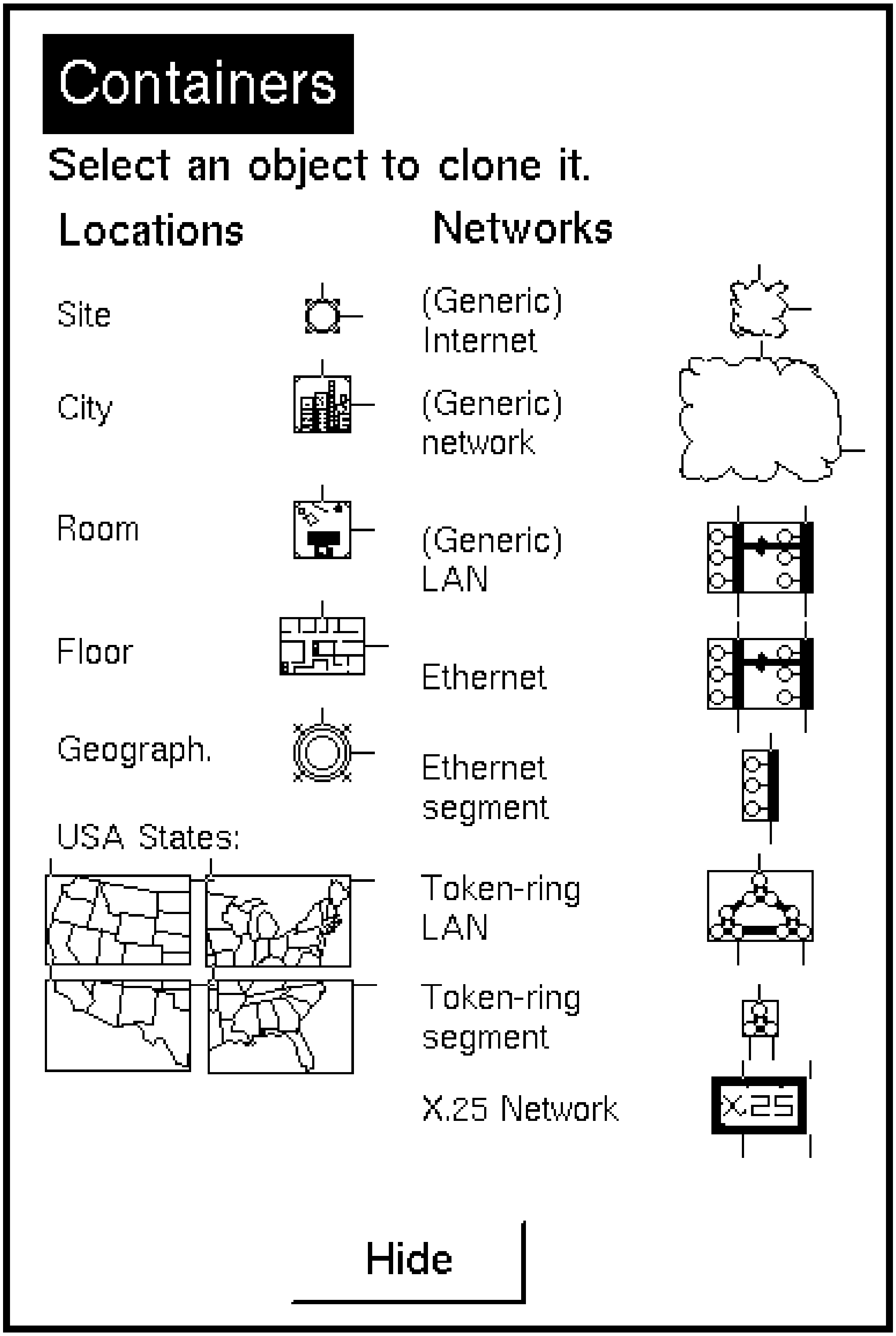
Example palettes for telecom-devices



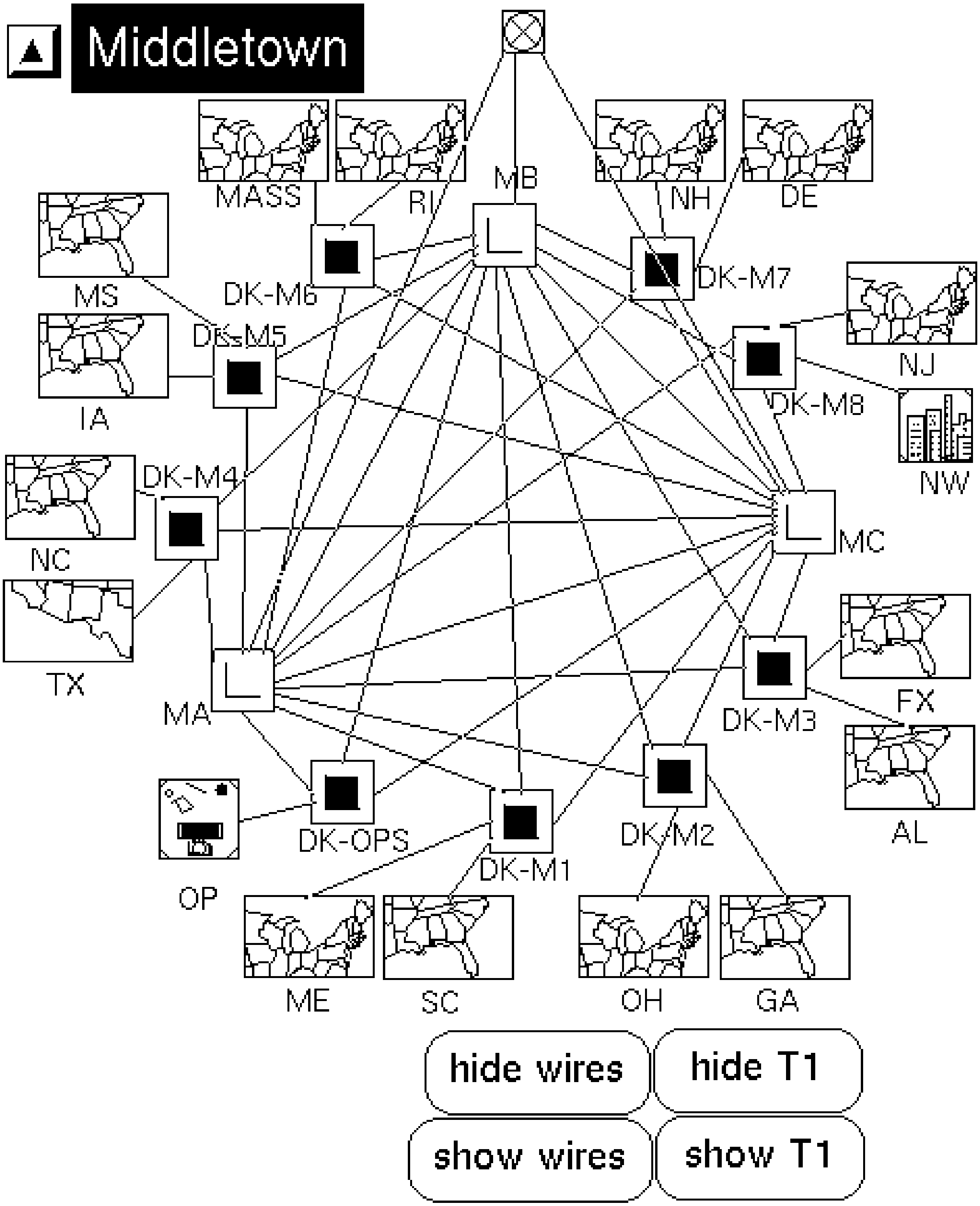
Example attribute table: telecom-device

NJAU, a gms-node	
On off	1
In service	1
What if in service	1
Highest message priority	99999
Acknowledgement status	acknowledged
Test state	no-test
Time last proven good	4618
User entered failure status	ok
Distance from g2	3
Polling method	ping
Test method	no-method
Address	""
Agent	snmp

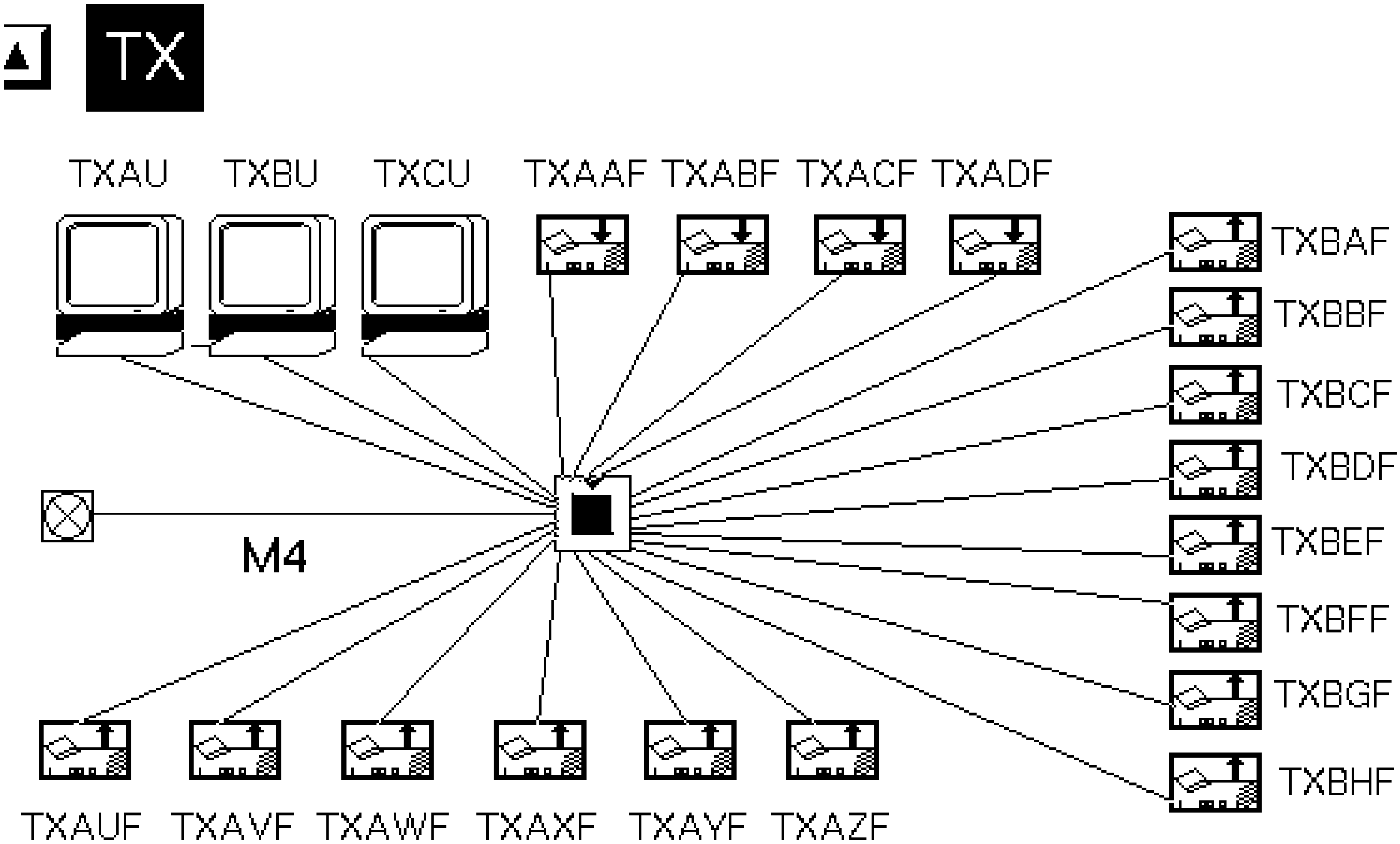
Container configuration palette



Workspace (map) hierarchy I



Workspace (map) hierarchy II



A site

NJ, a northeast-states-region	
Notes	OK
User restrictions	none
Names	NJ
Highest message priority	1
Acknowledgement status	unacknowledged

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

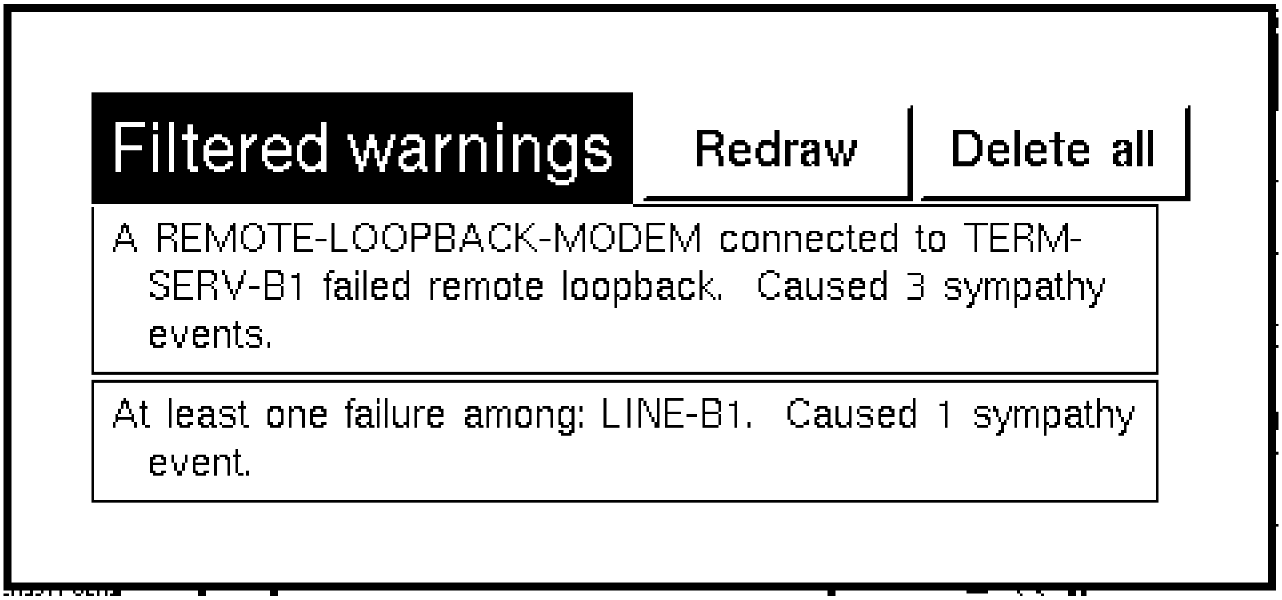
- Recognizing recurring problems & notify system administrator

Sample filtered message

a smh-small-message	
Notes	OK
User restrictions	none
Names	none
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
Acknowledger	none-yet
Additional text	"A REMOTE-LOOPBACK-MODEM connected to TERM-SERV-B1 failed remote loopback. Sympathy events for: BLOB, A REMOTE-LOOPBACK-MODEM on the subworkspace of GOTHAM-CITY, A REMOTE-LOOPBACK-MODEM connected to BLOB."
User comment	""

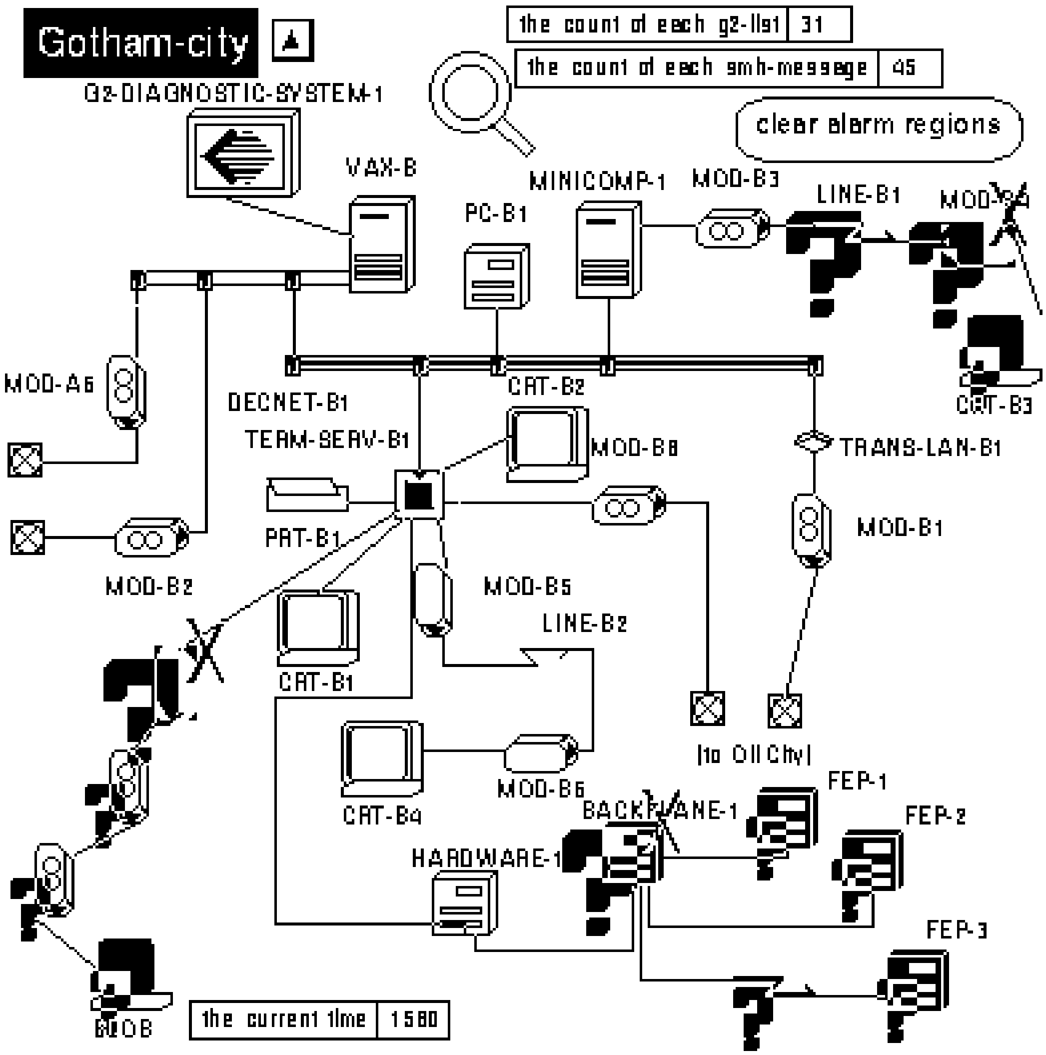
Filtered messages

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:



G2-manager-process

INMS-G2-MANAGER, a g2-manager-process	
Notes	OK
User restrictions	none
Names	INMS-G2-MANAGER
Model update interval	40
Routine pinging interval	30
Ping timeout interval	20
Polling interface	simulated
Suspect list	a net-item-list-without-duplicates
Proven good list	a net-item-list-without-duplicates
Failed list	a net-item-list-without-duplicates
Degraded list	a net-item-list-without-duplicates
Hypothetical model reachable list	a net-item-list-without-duplicates
Hypothetical model unreachable list	a net-item-list-without-duplicates
Model reachable list	a net-item-list-without-duplicates
Model unreachable list	a net-item-list-without-duplicates
Outstanding ping list	a net-item-list-without-duplicates
Failed ping list	a net-item-list-without-duplicates


Gotham-city
the count of each g2-list 31
the count of each smh-message 45
G2-DIAGNOSTIC-SYSTEM-1
clear alarm regions
VAX-B
PC-B1
MINICOMP-1
MOD-B3
LINE-B1
CRT-B3
MOD-A6
DECNET-B1
CRT-B2
TERM-SERV-B1
MOD-B8
TRANS-LAN-B1
PRT-B1
MOD-B1
MOD-B2
MOD-B5
LINE-B2
CRT-B1
CRT-B4
MOD-B6
BACKPLANE-1
FEP-1
FEP-2
HARDWARE-1
FEP-3
the current time 1580

GOTHAM-CITY

Browse results Redraw Delete

FEP-3 failed a ping

FEP-2 failed a ping

FEP-1 failed a ping

BACKPLANE-1 failed a ping

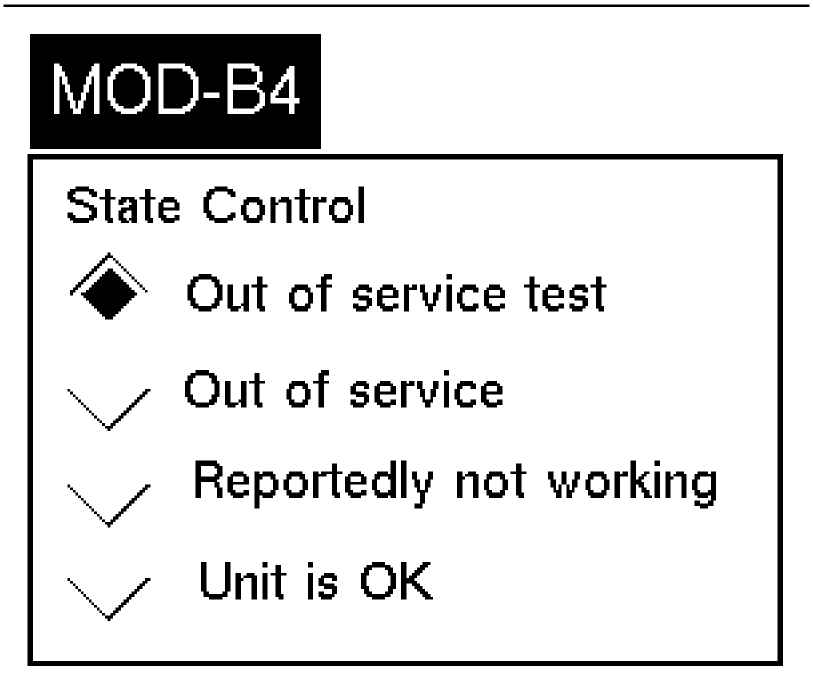
BACKPLANE-1 failed a ping. Caused 4 sympathy events.

BLOB failed a ping

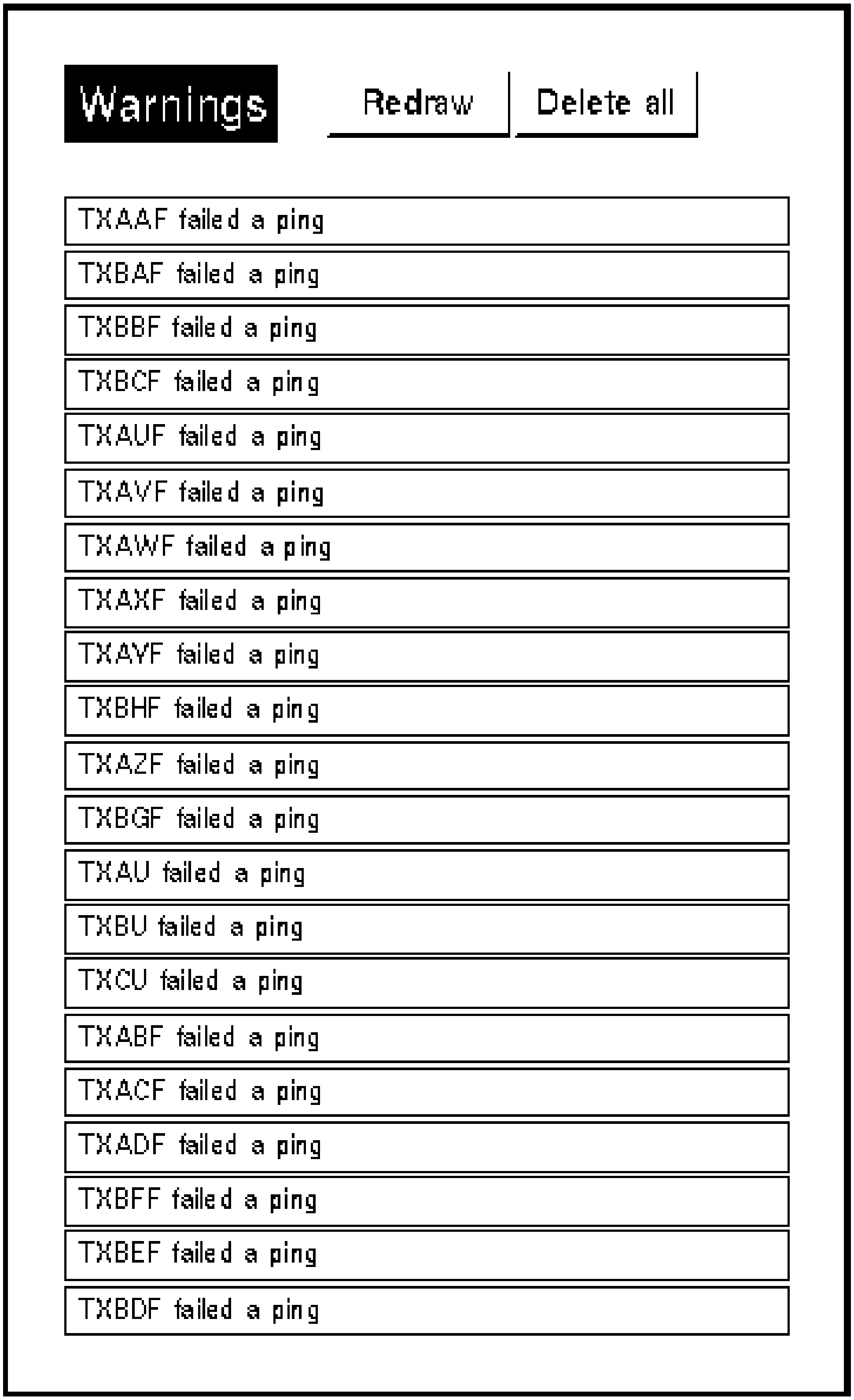
A REMOTE-LOOPBACK-MODEM connected to TERM-SERV-B1 failed remote loopback. Caused 3 sympathy events.

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

CRT-B3 failed a ping

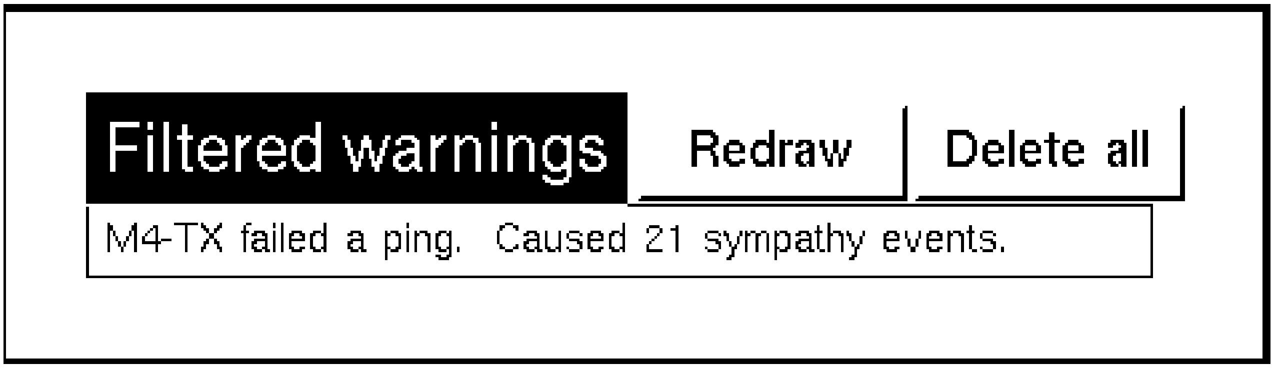

MOD-B4
State Control
Out of service test
Out of service
Reportedly not working
Unit is OK

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:

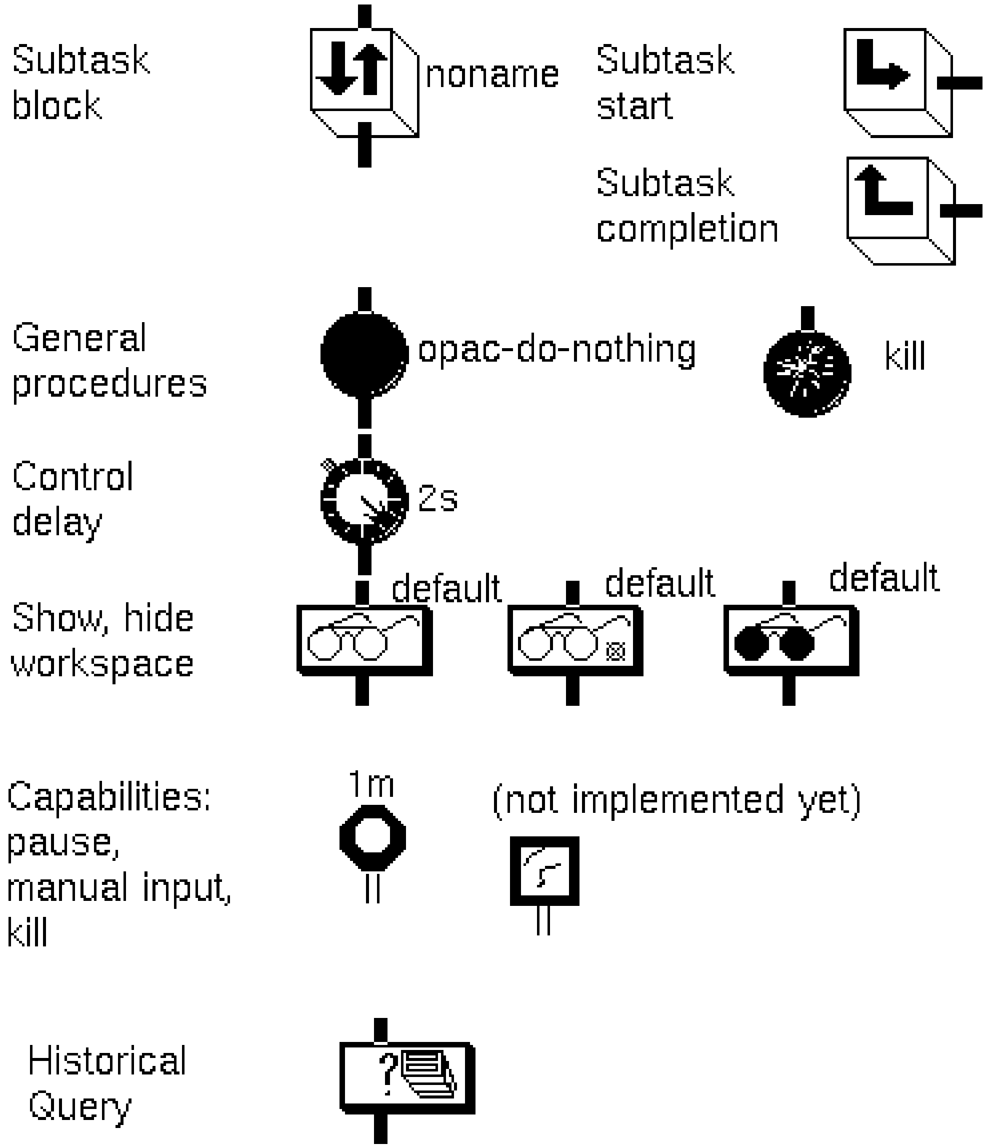
Filtered version of the previous 22 messages, sent to operator



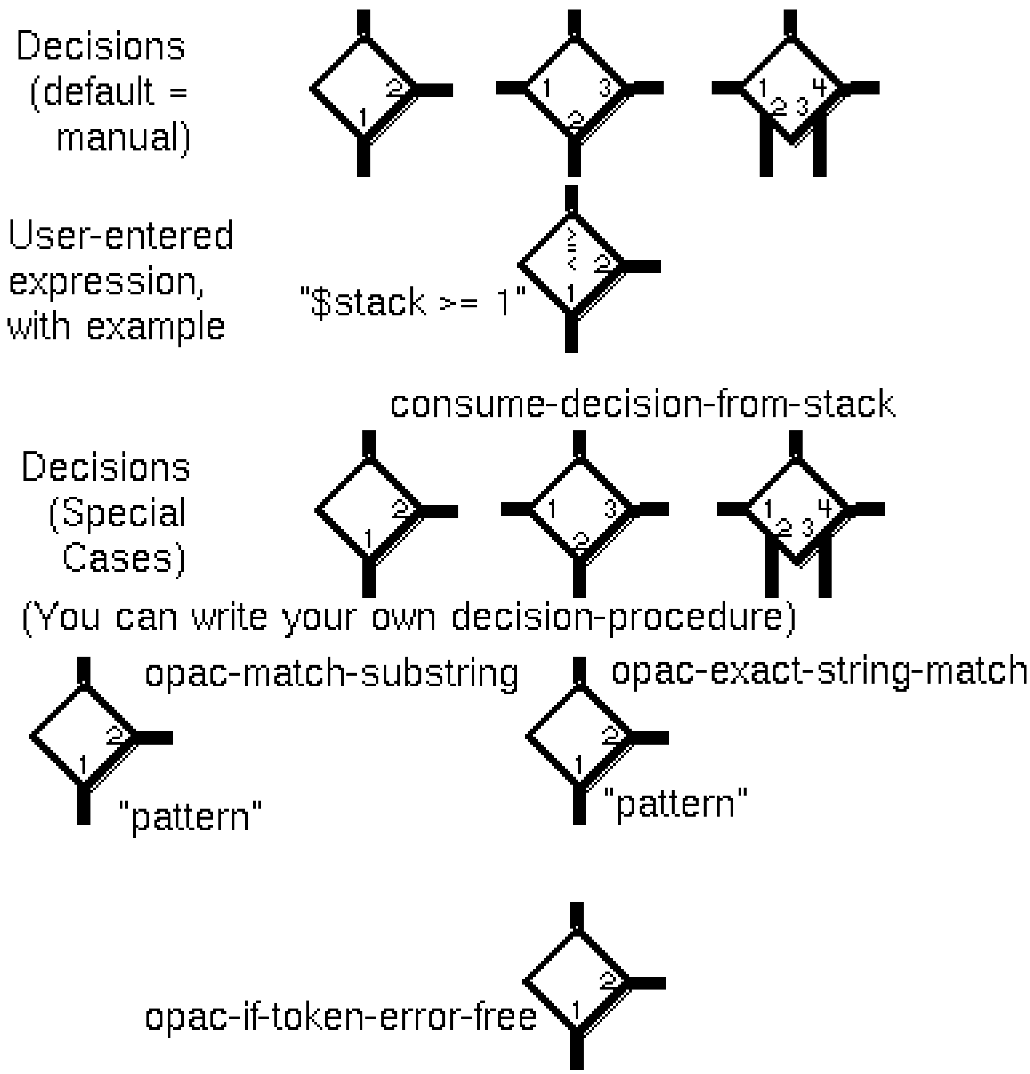
The details of this message show the original information that went into this summarized message. The additional-text explanation is assembled automatically.

a smh-small-message	
M4-TX failed a ping. Caused 21 sympathy events.	
Sender	"INMS-G2-MANAGER"
Target	"M4-TX"
Message id	150
Repetitions	0
Creation time	14 Mar 94 3:13:30 p.m.
Revisit time	19 Aug 98 9:14:26 a.m.
Revisit method	no-method
Deletion time	26 Mar 94 5:05:05 a.m.
Priority	1
Acknowledgement status	unacknowledged
Acknowledger	none-yet
Additional text	"M4-TX failed a ping. Adjacent node DK-M4 is good. Sympathy events for: TXAAF, TXBAF, TXBBF, TXBCF, TXAUF, TXAVF, TXAWF, TXAXF, TXAYF, TXBHF, TXAZF, TXBGF, TXAU, TXBU, TXCU, TXABF, TXACF, TXADF, TXBFF, TXBEF, TXBDF. Original messages now automatically acknowledged: M4-TX failed a ping TXAAF failed a ping TXBAF failed a ping TXBBF failed a ping TXBCF failed a ping TXAUF failed a ping TXAVF failed a ping TXAWF failed a ping TXAXF failed a ping TXAYF failed a ping TXBHF failed a ping TXAZF failed a ping TXBGF failed a ping TXAU 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	""

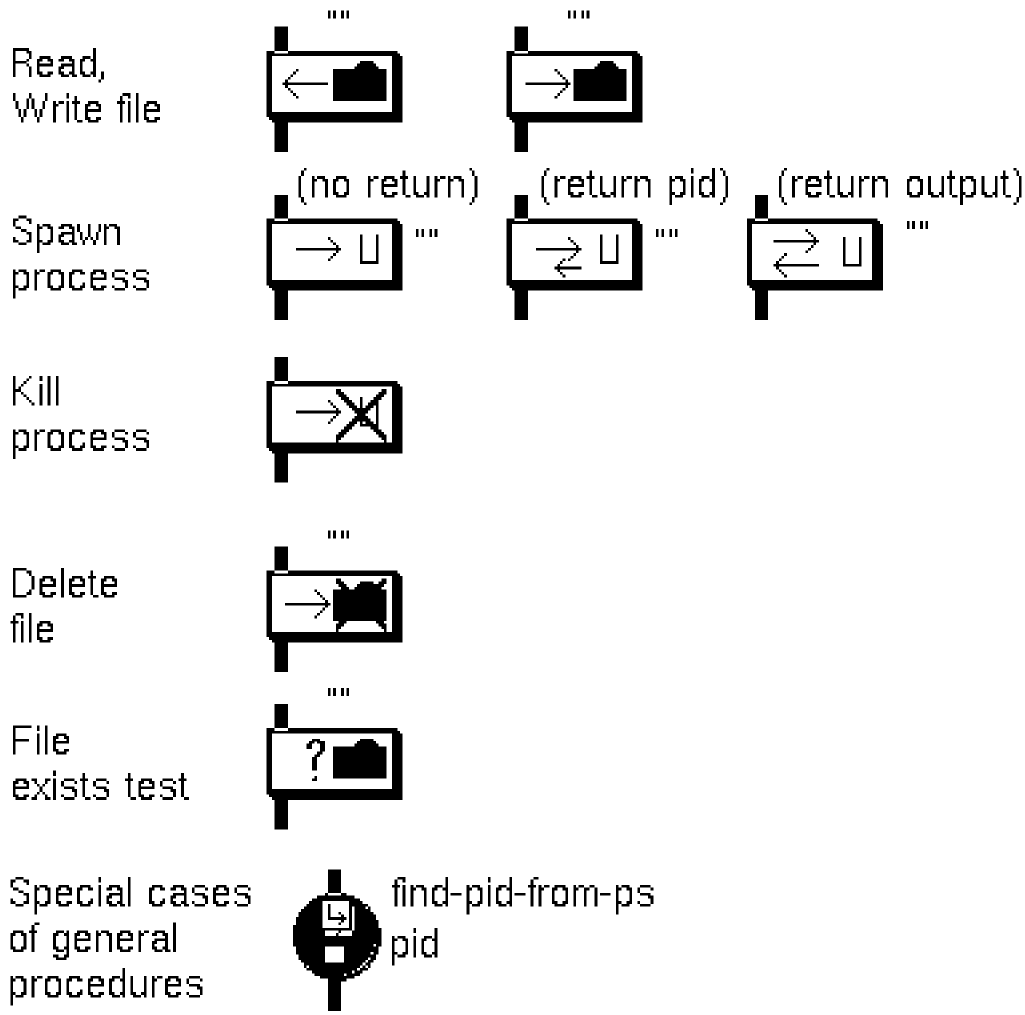
General actions palette



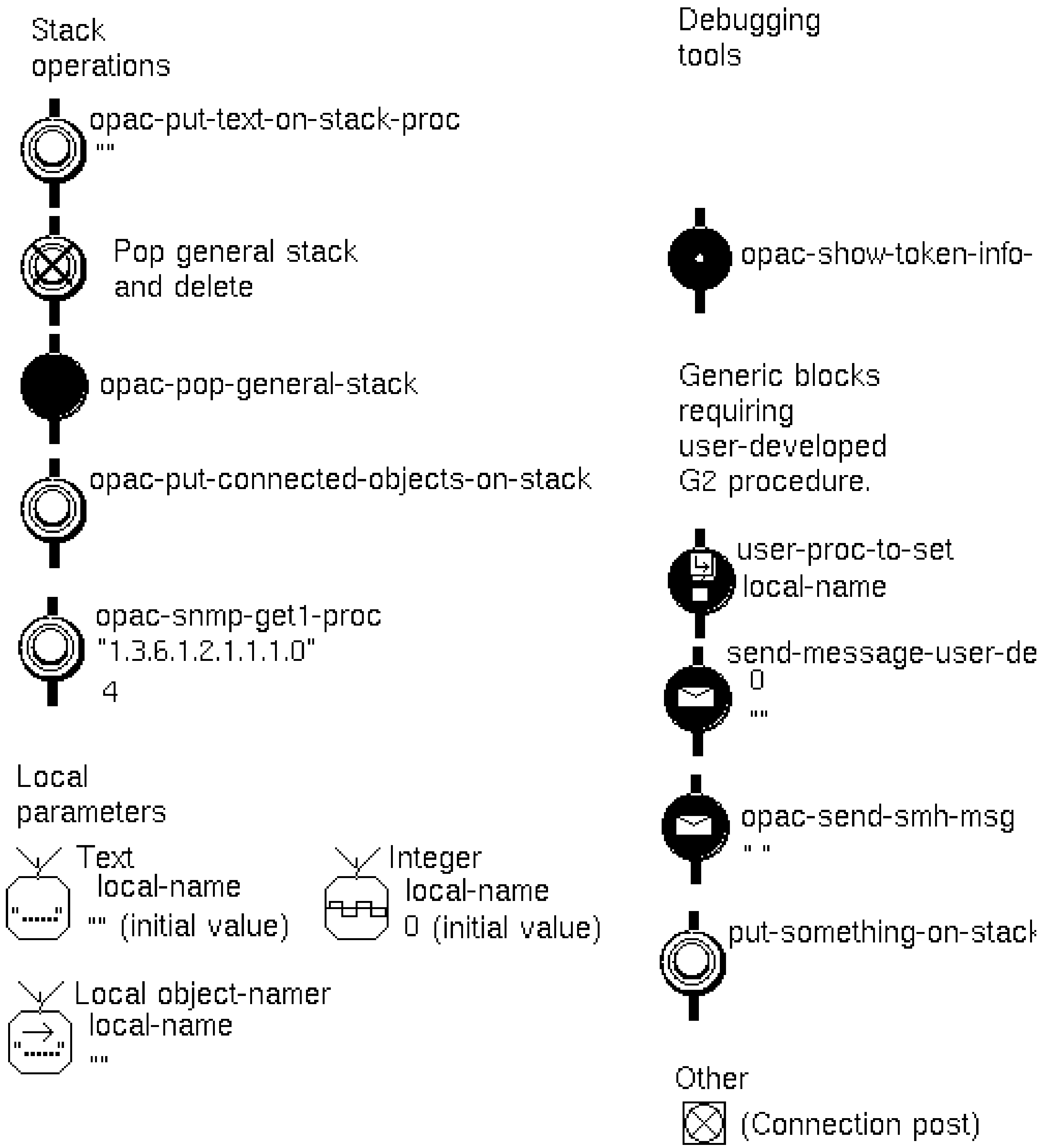
Decisions palette

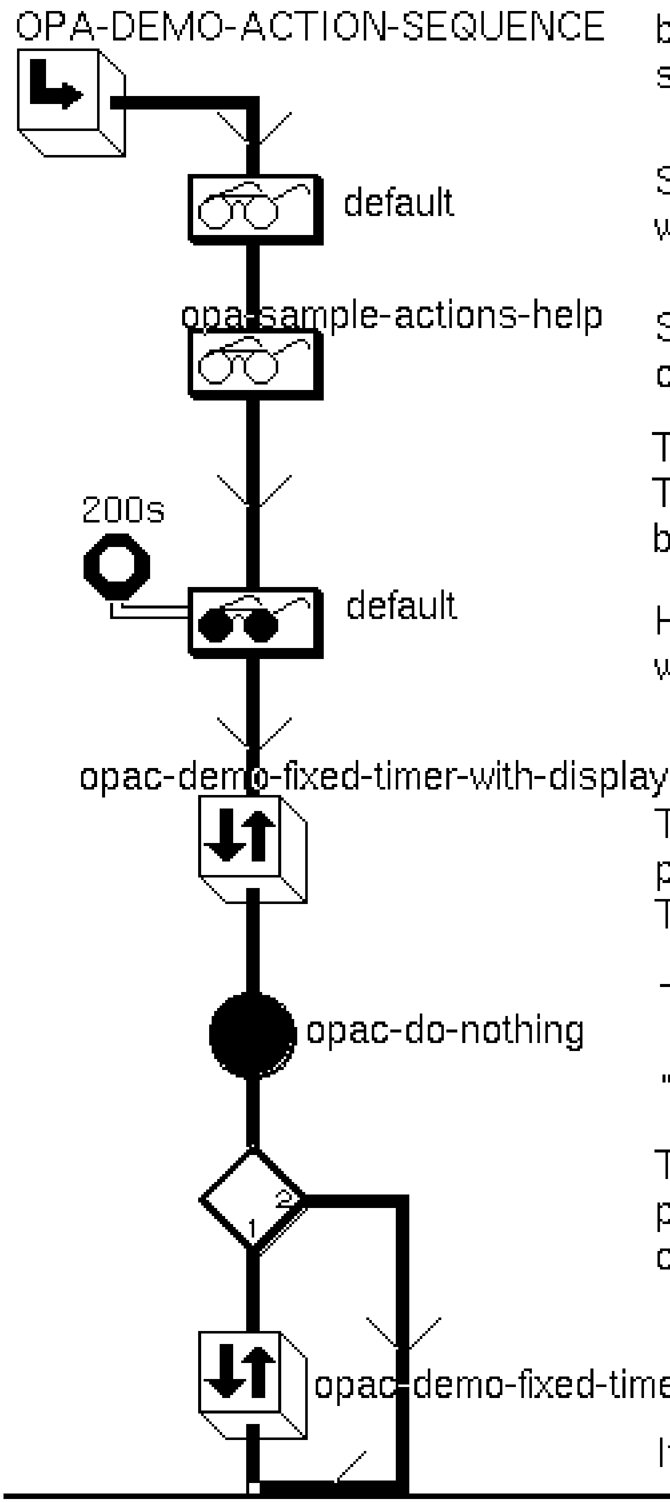


UNIX actions palette



Misc. actions palette





The name of this start block is the subtask, whick by name, multiple times, concurrently, from multipl 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-sig This pauses, generating a dialog for the user to c but times out as shown at 20 seconds.

Hides most recently-displayed workspace as default.

OPAC-DEMO-FIXED-TIMER-WITH-DISPL

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

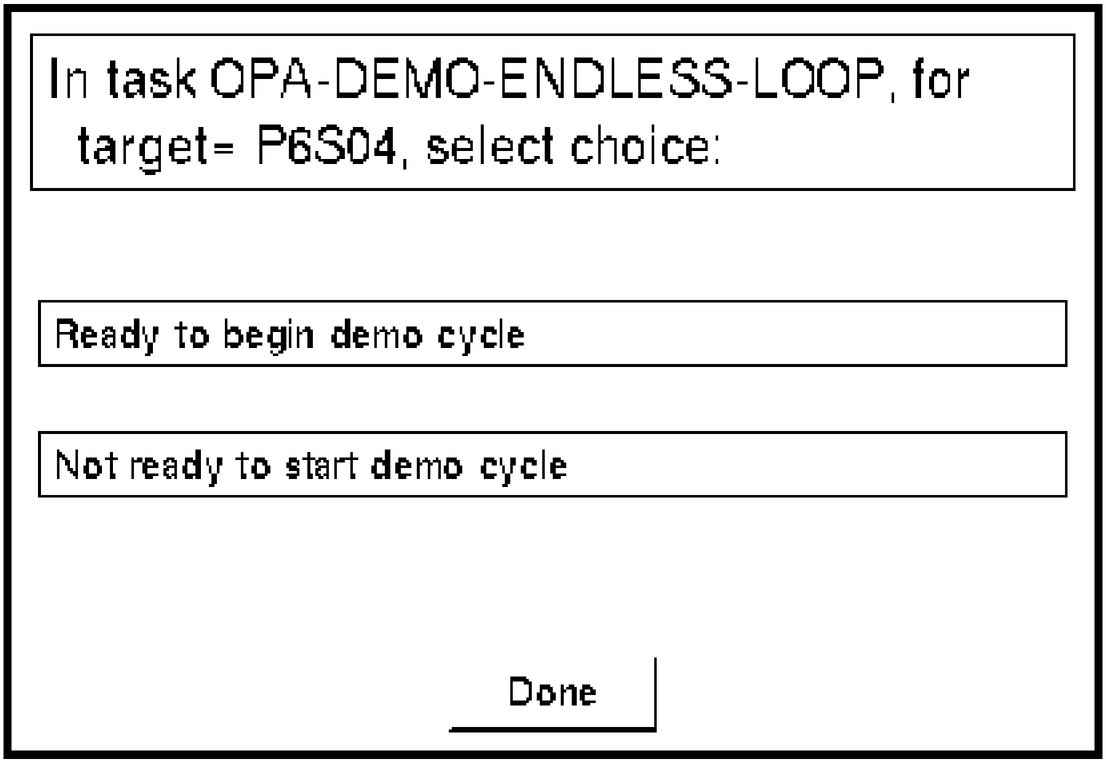
This "general procedure" block executes a G2 pı by name (the displayed attribute). This one calls "opac-do-nothing", just sending a message.

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.

opac-demo-fixed-timer-with-display

If chosen, the same timer demo is called again.

Decision block with manual input



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 $.

an opac-2-way-decision	
Control proc	opac-default-control-proc-for-decisions
Decision proc	opac-manual-decision-proc
Timeout	60
Default decision	1
Header text	"In task $task, for target= $target, select choice:"
Choice 1 description	"Ready to begin demo cycle"
Choice 2 description	"Not ready to start demo cycle"

Block pause capability

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

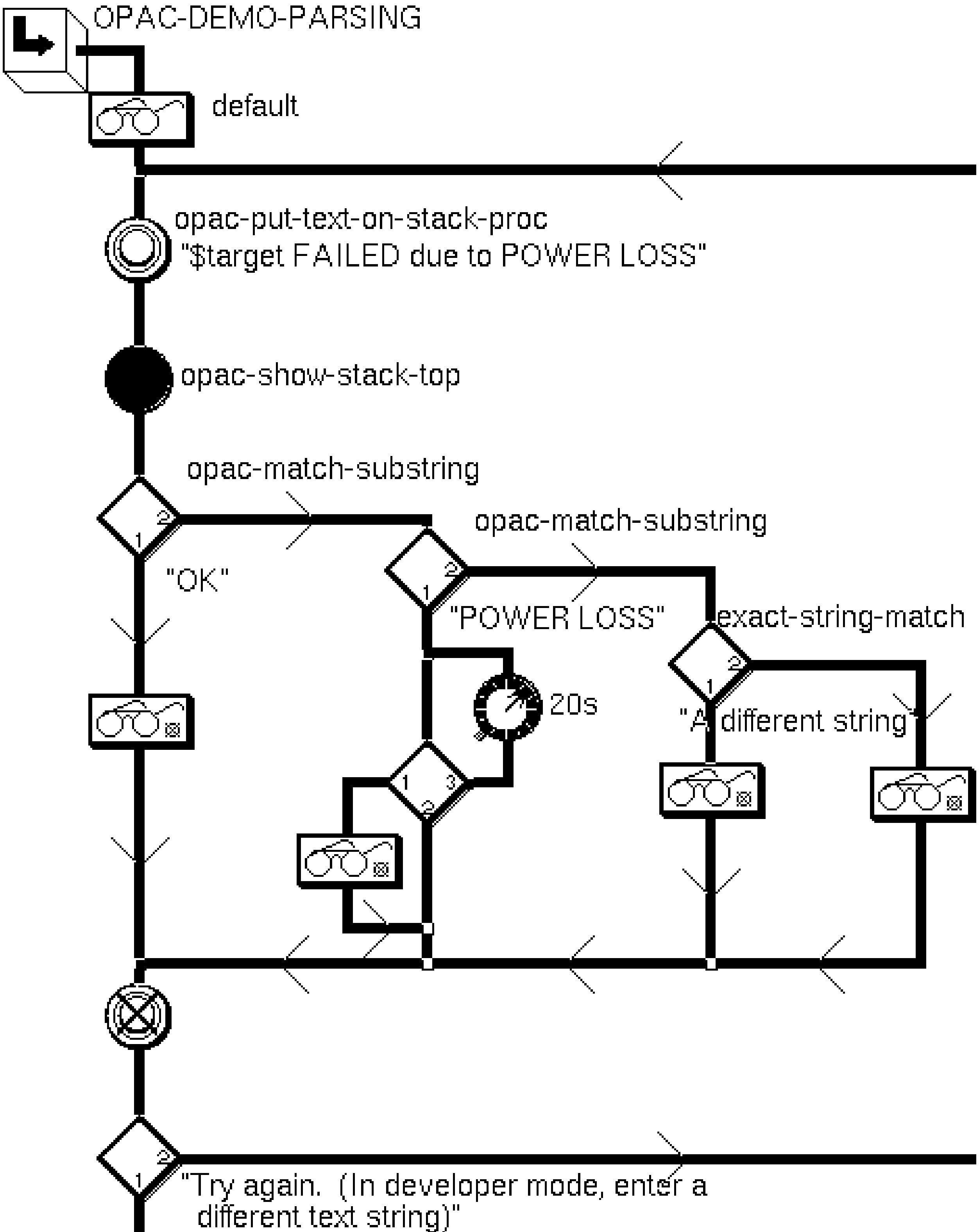
Continue with next step, A OPAC-HIDE-WORKSPACE on the subworkspace of SAMPLE-ACTION-DIAGRAMS

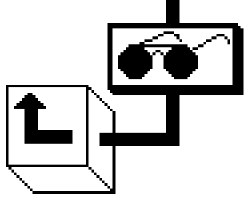
Abort this task

Done

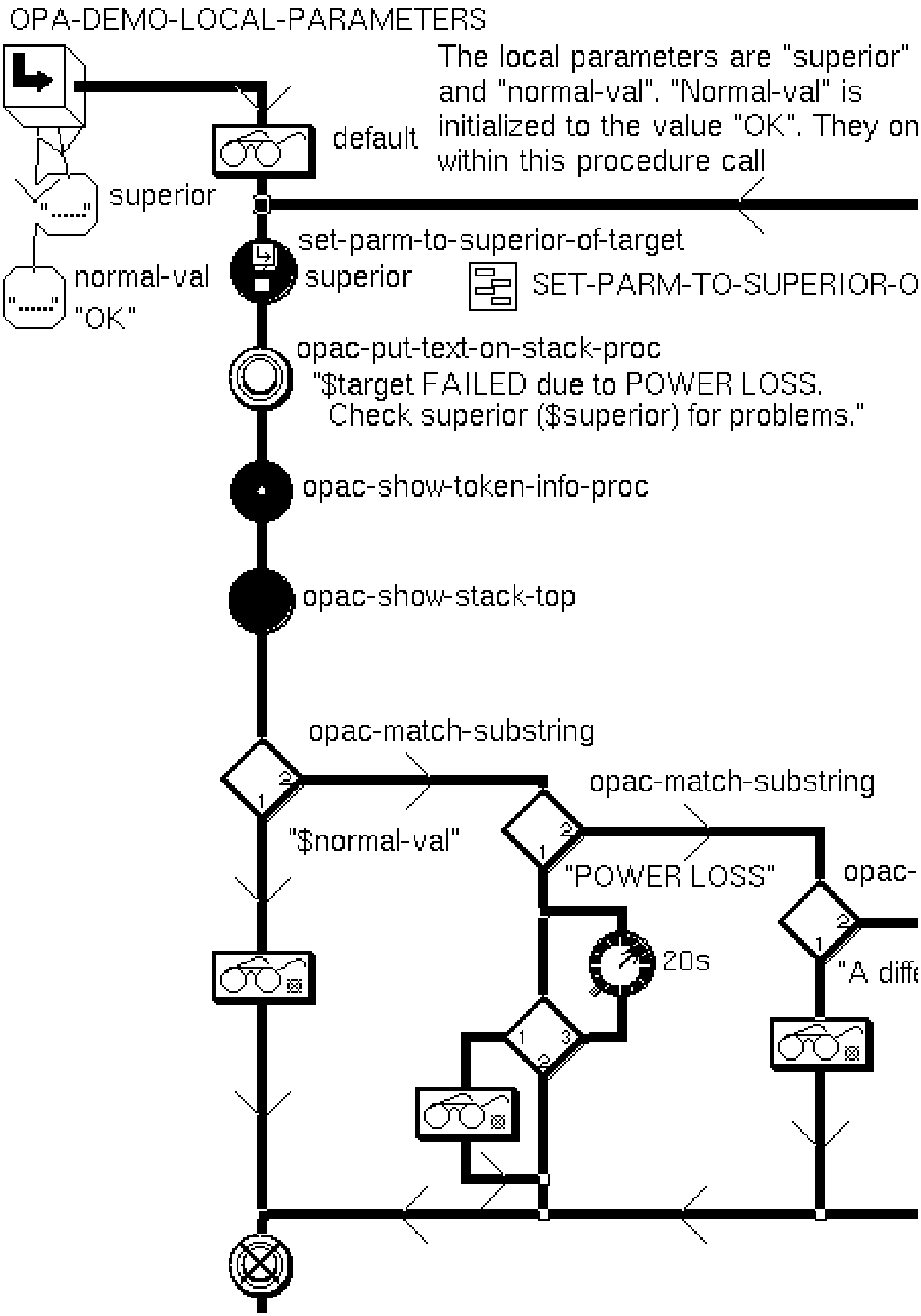
The menu above was generated automatically by the block-pause-capability, which had the following table. Note the use of variables, indicated with the $.

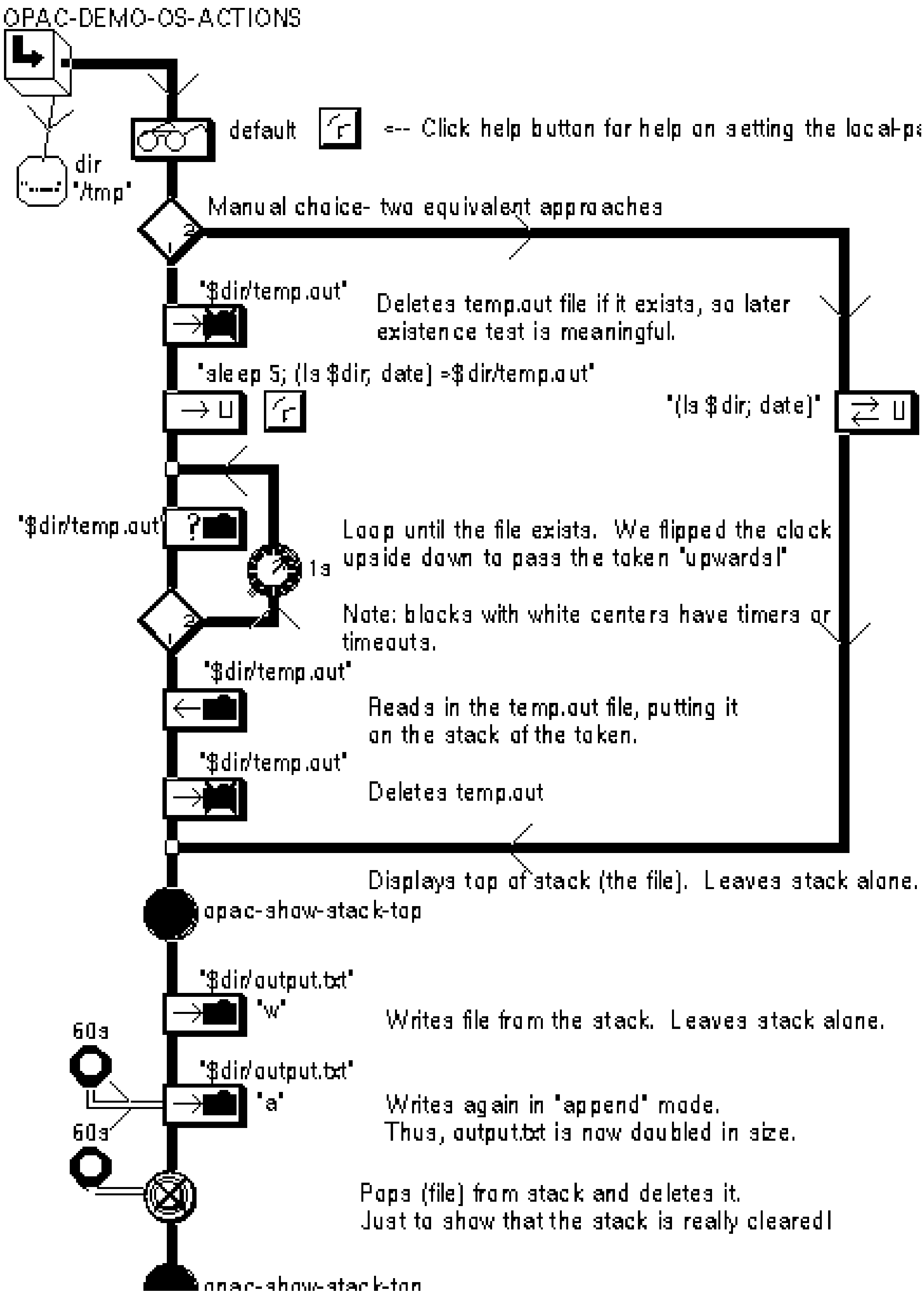
an opac-block-pause-capability	
Control proc	opac-block-pause-proc
Header text	"Pause in task $task, for target=$target. Select choice:"
Continue text	"Continue with next step, $block"
Timeout	200s



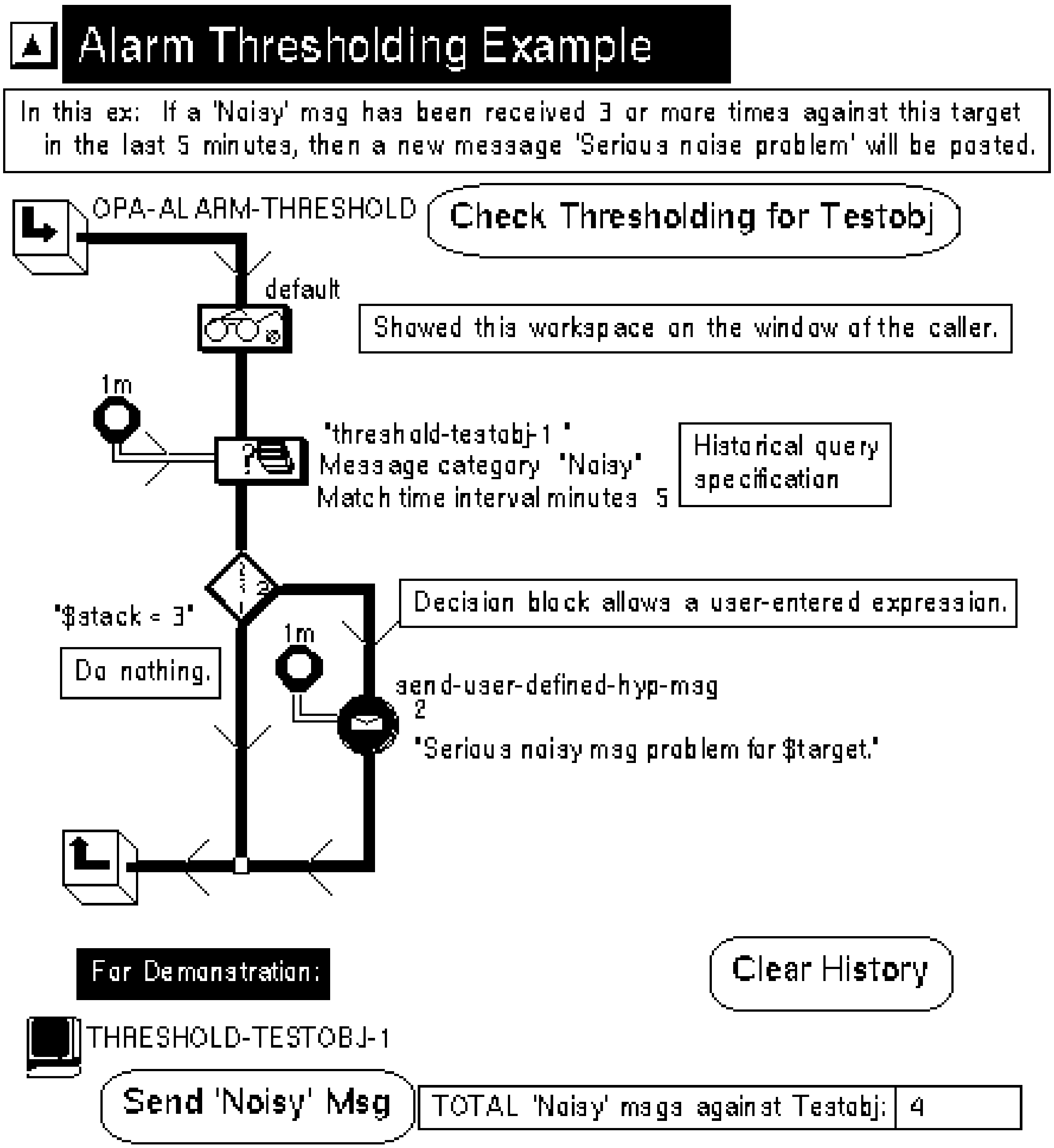


Note the use of $ in strings for substitution of commonly-
e.g., target object. You can also use $ for local variables
the $ inside of the pattern text for the match-substring ar
match. See the demo on local parameters, coming next.


OPA-DEMO-LOCAL-PARAMETERS
The local parameters are "superior" and "normal-val". "Normal-val" is initialized to the value "OK". They on
within this procedure call
default
superior
"......"
normal-val
"OK"
set-parm-to-superior-of-target
superior
SET-PARM-TO-SUPERIOR-O
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-
20s
"A diffe



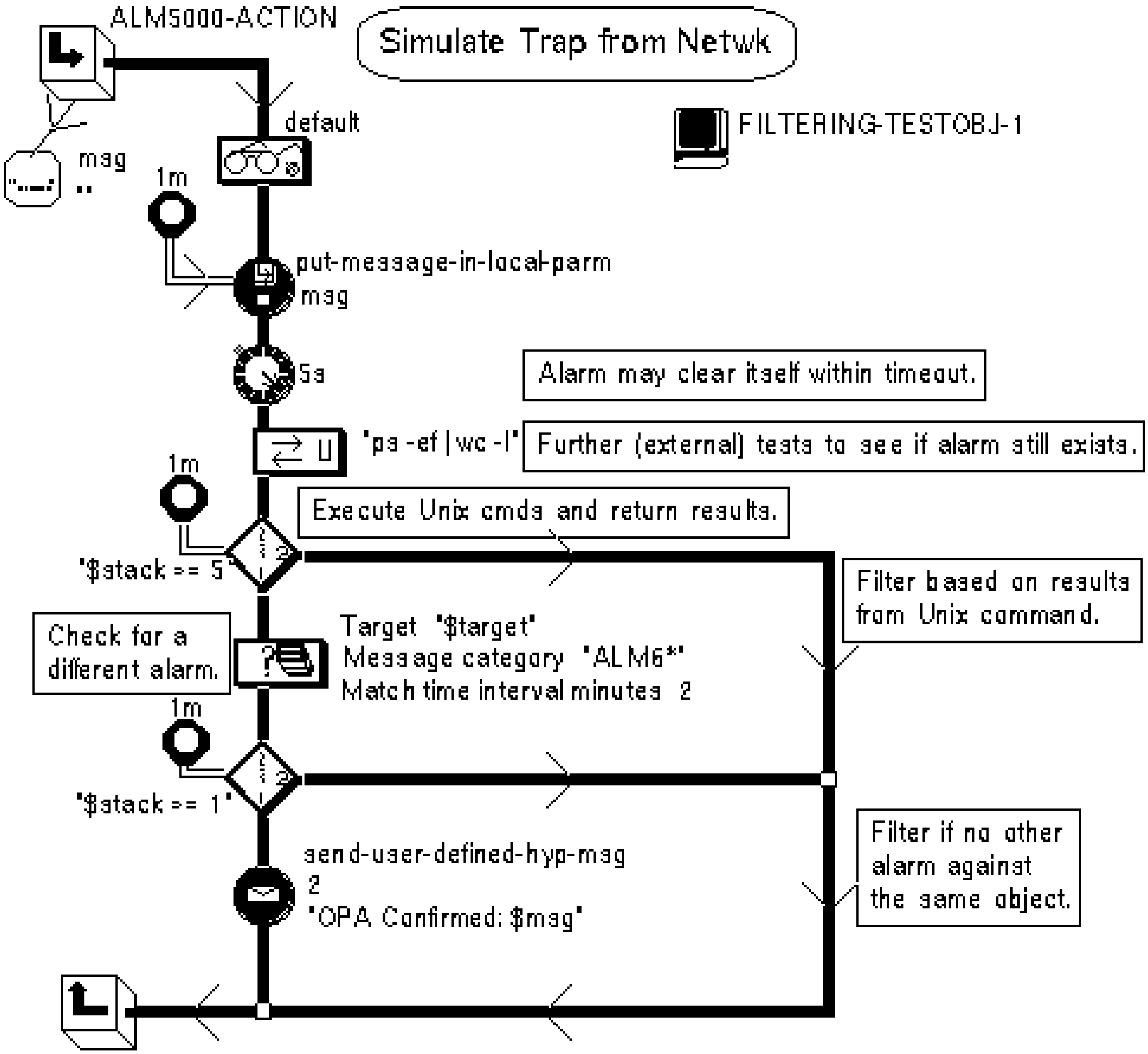
OPAC-DEMO-OS-ACTIONS
default
<-- Click help button for help on setting the local-pa
dir
"/tmp"
Manual choice- two equivalent approaches
"$dir/temp.out"
Deletes temp.out file if it exists, so later
existence test is meaningful.
"sleep 5; (ls $dir; date) >$dir/temp.out"
"(ls $dir; date)"
"$dir/temp.out"
Loop until the file exists. We flipped the clock
upside down to pass the token "upwards!"
1s
Note: blocks with white centers have timers or
timeouts.
"$dir/temp.out"
Reads in the temp.out file, putting it
on the stack of the token.
"$dir/temp.out"
Deletes temp.out
Displays top of stack (the file). Leaves stack alone.
opac-show-stack-top
"$dir/output.txt"
"w"
Writes file from the stack. Leaves stack alone.
60s
"$dir/output.txt"
"a"
Writes again in "append" mode.
Thus, output.txt is now doubled in size.
60s
Pops (file) from stack and deletes it.
Just to show that the stack is really cleared!
opac-show-stack-top


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.
OPA-ALARM-THRESHOLD
Check Thresholding for Testobj
default
Showed this workspace on the window of the caller.
1m
"threshold-testobj-1 "
Message category "Noisy"
Match time interval minutes 5
Historical query specification
"$stack = 3"
Decision block allows a user-entered expression.
Do nothing.
1m
send-user-defined-hyp-msg
2
"Serious noisy msg problem for $target."
For Demonstration:
Clear History
THRESHOLD-TESTOBJ-1
Send 'Noisy' Msg
TOTAL 'Noisy' msgs against Testobj:
4

an opac-historical-message-query	
Database	smh-history
Results destination	stack
Query class	smh-history
Pattern	""
Results as count only	yes
Match time end	current-time
Match time interval minutes	5
Target	"threshold-testobj-1 "
Sender	""
Message category	"Noisy"

Alarm Filtering / Diagnosis Examples

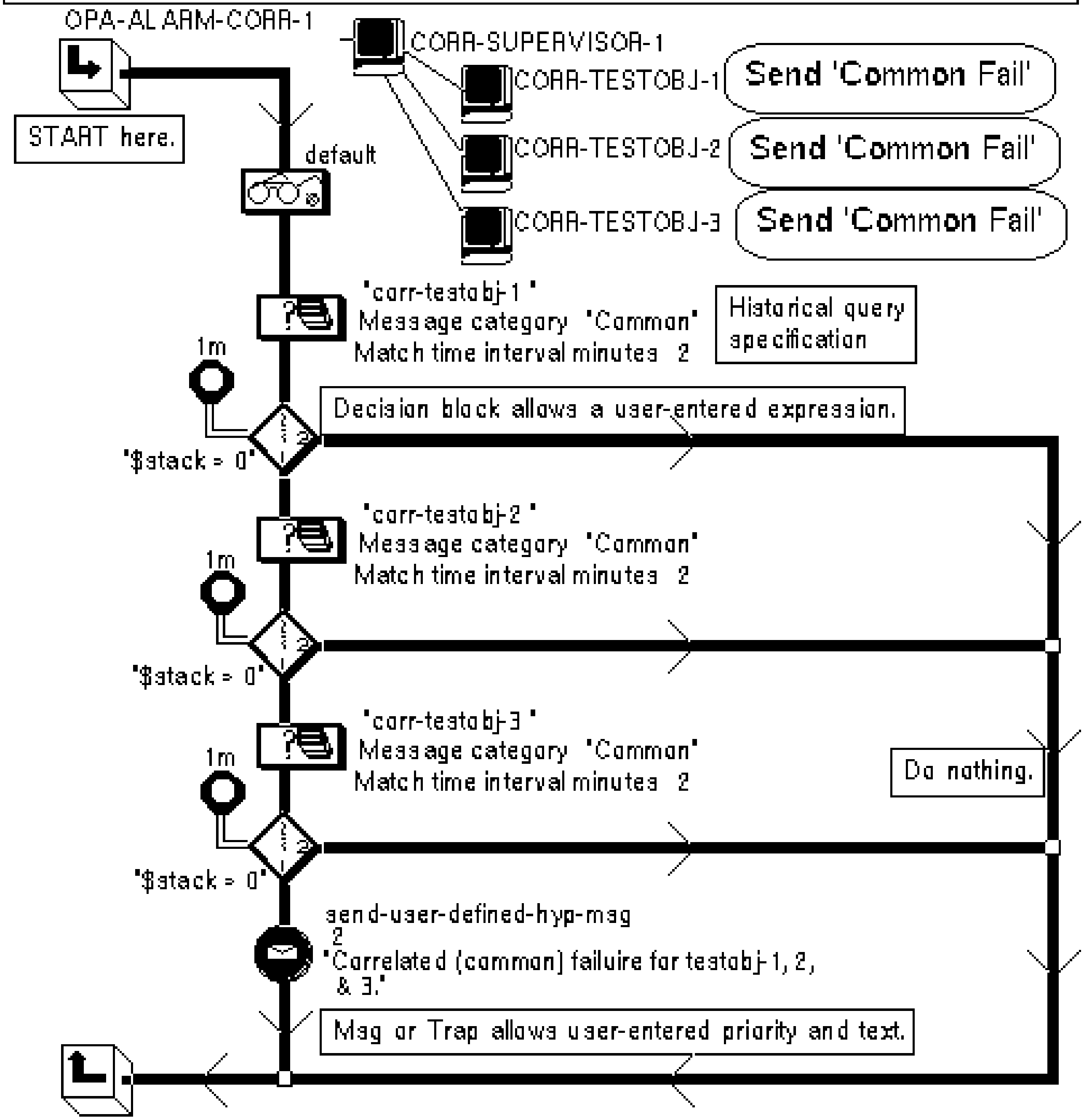
Network traps / messages can be diagnosed & filtered according to almost any criteria, and using a variety of OPAC tools. This shows a few examples.



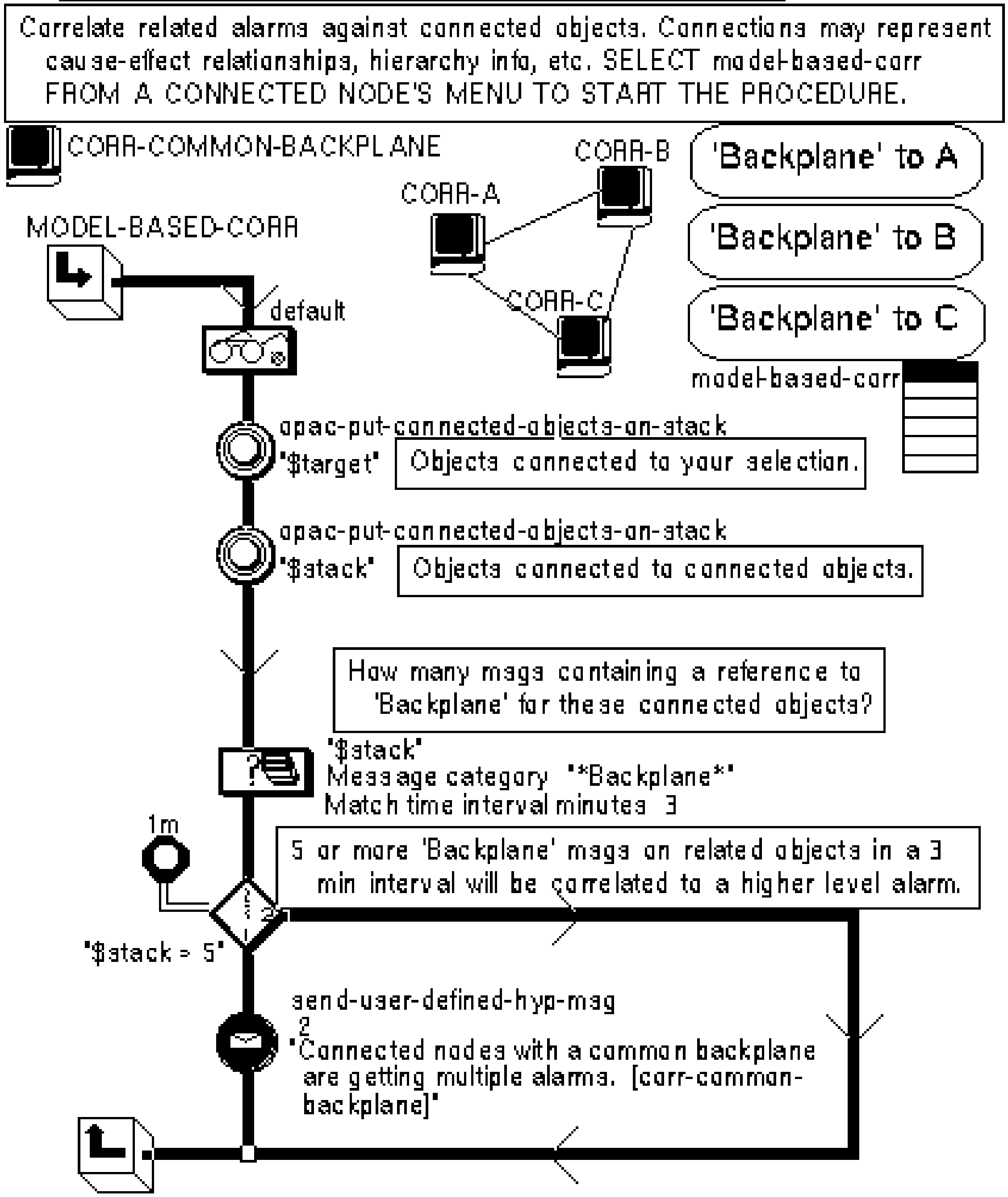
an opac-historical-message-query	
Database	smh-history
Results destination	stack
Query class	smh-history
Pattern	""
Results as count only	yes
Match time end	current-time
Match time interval minutes	2
Target	"$target"
Sender	""
Message category	"ALM6*"

Alarm Correlation Example

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.



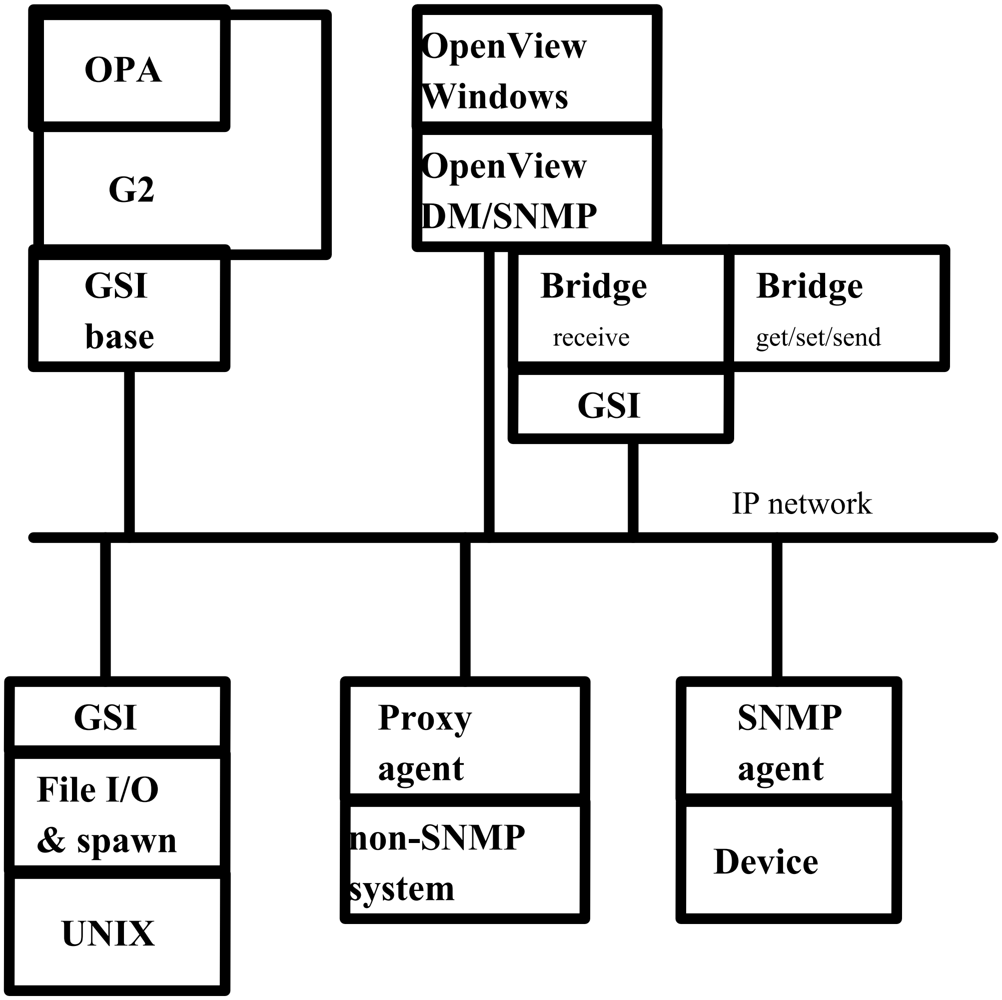
Model-Based Correlation Example



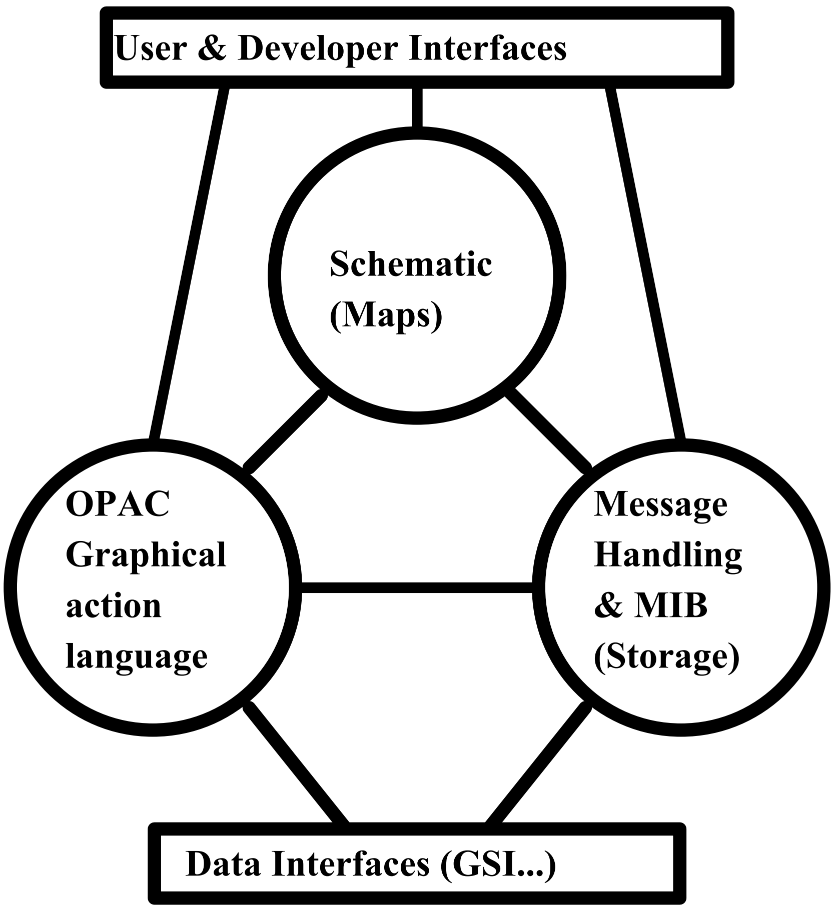
III. OPA Architecture

- Overall Architecture/System Integration
- Major components

Overall Architecture



OPA Major building blocks



Message Management - a message "MIB"

- Messages are objects with attributes such as priority, acknowledgement status, time stamp, timeouts for proecedure 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)