# Integrity Reasoner White Paper

**Dr. Ramesh Vaidhyanathan and Dr. Gregory Stanley**
**Gensym Corporation, 1776 Yorktown, Suite#700, Houston, TX, U.S.A.**

## 1. What is Integrity Reasoner?

Operators of large systems are constantly bombarded by asynchronous events in the form of messages and alarms, often resulting from a small number of root cause problems. Operators cannot adequately analyze all these events in real-time. Once events are detected, or arrive from an external system, operators need an intelligent system to:

- Filter out redundant events so attention can be focused on truly new information.
- Recognize and correlate related events and present a summary to the operator, even if the root cause is not determined.
- Diagnose the root causes of problems by identifying suspected root-cause failures based on incoming symptoms, and selecting additional tests to run in order to investigate those root-causes.
- Run automated or guided manual tests as soon as there is evidence of a problem.
- Proactively execute automated or guided manual corrective actions to address symptoms or fix problems.
- Communicate results to users and to other computer applications.
- Perform Impact analysis / What-if simulation of the failures to predict their effects in the system, for example, projecting failure impact upon service levels.

Integrity Reasoner incorporates Gensym's SymCure technology. SymCure is a development and deployment environment for building intelligent systems to automate real-time fault and availability management of large-scale operations — addresses these goals. SymCure performs on-line event correlation and interactive diagnosis, to address the full life cycle of problem identification based on symptoms, root-cause analysis, diagnostic testing, fault isolation, and recovery.

SymCure provides a powerful model-based framework consisting of generic, class-based fault propagation models, tied to an object-oriented domain representation and scalable algorithms. SymCure also provides the tools and a graphical user interface (GUI) for:

- The development, debugging and analysis of the fault propagation models.
- Runtime operator interaction.

In addition, SymCure automatically accounts for configuration changes in equipment, topology, or operating modes in managed operations, requiring minimal on-site manual customization for practical deployment and thus eliminating the need for expensive reconfigurations of the managing applications.

This enables application developers to implement fault and availability management capabilities quickly and reliably, and build re-usable applications.
SymCure can be used to automate fault and availability management of operations in domains as diverse as communications networks, enterprise-wide soft-ware applications, and manufacturing process plants. SymCure is part of the Integrity product family, which includes:

- CORE Services - the Integrity foundation classes, a message management system, tools for building a domain map representing the managed system, and a set of utilities.
- OPAC - a graphic programming language.
- Bridges - a group of products for interfacing external systems to Integrity.
- Industry and Vendor Libraries - classes, objects, and tools specific to certain domains or vendors.

## 2. What is a Fault Propagation Model?

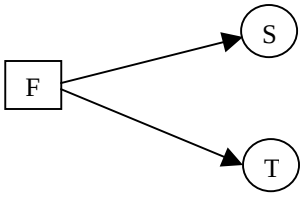
A fault propagation model in SymCure describes the propagation of failures via potential failure paths in the system by modeling how fault events will cause other symptom events and test-result events. It is a causal directed graph (digraph) model, where:

- Nodes represent fault, symptom, and test events, which can take the fuzzy truth-value greater than or equal to 0.0, or “false,” and less than or equal to 1. 0, or “true.” A value in between 0.0 and 0.5 is “fuzzy false.” A value in between 0.5 and 1.0 is “fuzzy true.” A value of 0.5 is “unknown.”

- Symptom and Test events can use AND/OR logic for the propagation of failures via these events.

- Arcs, called causal links, connecting the nodes, represent the “cause and effect” relationships or dependencies among these events.

As the following figure illustrates, a causal link from a fault event node F to symptom event node S means that fault event F (true) causes symptom event S (true). Thus, if fault event F occurs, then symptom event S will occur. In addition, fault event F (true) causes the test event T (true). Thus, the result of the test event T can prove (true) or disprove (false) the fact that fault event F is the root-cause failure depending on its outcome (true/false).



One can think of these models as a “simulation” for predicting impacts – if fault F occurs (“true”), then one would expect to see downstream symptoms S and T to occur (to become “true”). In a real model with multiple possible faults, one can also think of the use of the model for diagnosis: If a symptom occurs, any “upstream” faults are possible causes of that symptom. In real models, there may also be many intermediate symptoms as well. The advantage of having a single model is that it can be used both for impact analysis (simulation/what-if), and for diagnosis (find possible causes for a given symptom).

In SymCure, once event interfaces are in place, the main focus of applications development becomes the building of this model. This model can be inspected and discussed by a much larger group of people than possible with other types of tools.

# 3. Concepts and Terms

### *Faults*

A fault is an underlying independent root cause of observed symptoms that indicate some sort of failure or problem. For example, a fault could be a loose connection in a communications network or a disk full in a computer.

In SymCure fuzzy truth values determine the status of the fault event, where:

- A fuzzy true value between 0.5 and 1.0 indicates belief that the fault has occurred.
- A fuzzy false value between 0.0 and 0.5 indicates belief that the fault did not occur.
- A fault with a value of 0.5 (unknown) becomes a “Suspect” if it is a possible cause of observed symptoms with fuzzy true value.

Faults can have associated corrective or mitigation actions that can be started against a target domain object, when a fault is suspected as a root cause of the observed symptoms and when a fault is inferred to have occurred. These actions can be executed automatically or presented to an operator for manual execution. For example, a suspect mitigation method could be defined for a critical fault to pro-actively execute a set of mitigation actions as soon as it is suspected as a root-cause, without waiting for SymCure to diagnose and resolve that fault. Similarly, a mitigation method could be defined for a fault to guide the operator through a series of actions to be executed in order to mitigate the fault identified as a root-cause.

In addition a recovery-method can be defined for a fault to be executed automatically when a root-cause fault gets fixed. This method can be used to execute a set of actions when a fault that was previously identified as a root-cause is mitigated and recovered. For example, a recovery method could be defined for a fault to delete and clean-up all the root-cause fault messages created and presented to the operator earlier about that fault.

### *Symptoms*

A symptom is an effect of an underlying fault in the managed system. The effects of faults are generally passed from object to object via symptoms. For example, a loose connection fault may cause a symptom of corrupted messages. The corrupted messages themselves are then passed from node to node in a communications network.

For a symptom event:

- A fuzzy true value between 0.5 and 1.0 indicates belief that the symptom has occurred.
- A fuzzy false value between 0.0 and 0.5 indicates belief that the symptom did not occur.

Symptoms can represent either the events that are actually monitored in the system or events that can only be inferred from other events. A monitored symptom arrives asynchronously and unsolicited from external systems. The measurements/data that are easily available using current instrumentation are modeled as symptoms by specifying appropriate thresholds for the measured values. These symptoms might be polled periodically by a performance monitoring system and sent to SymCure if the specified threshold is crossed.

An “inferred-only” symptom is used for convenience in modeling to represent those state changes, such as failed sensors, and effects that cannot be directly monitored but are important for fault propagation. Examples could include corrupted-messages explained above, which are not directly monitored, but propagate from object to object as effects of the fault.

The symptoms can be defined to use OR or AND logic for event propagation. An Or-symptom indicates that the symptom will be observed if any of its input events are true. An And-symptom indicates that the symptom will be observed only if all of its input events are true.

### *Tests*

A test is an observable effect of a fault in the managed system, like a monitored symptom. Test results arrive asynchronously and can be unsolicited, just like a symptom. But unlike symptoms, a test can be requested at any time.

For a test event:

- A fuzzy true value between 0.5 and 1.0 indicates belief that the test result is affirmative.
- A fuzzy false value between 0.0 and 0.5 indicates belief that the test result is negative.

The notion of test is powerful and general and can imbed arbitrarily complex

analysis and actions, as long as it returns a single fuzzy truth-value. The events representing observations on request that require a series of actions and analysis of the measured data should be defined as tests. Test events have an associated set of actions, which can be started externally
against a specific domain object, for example, pinging a computer to make sure it is reachable. The test actions can be executed automatically or presented to an operator for execution. Upon request, after some indeterminate amount of time, the result of the test is returned to SymCure.

The tests can be defined to use OR or AND logic for event propagation. An Or-test indicates that the test result will be true if any of its input events are true. An And-test indicates the test result will be true only if all of its input events are true.

### *Ambiguity Group*

An ambiguity group is a set of faults that are suspected as root-causes to be resolved by SymCure. Since SymCure does not have a single fault assumption, one or more faults of an ambiguity group could be actual root-causes. Thus, an ambiguity group represents a partial diagnosis result.

An ambiguity group could be sent to SymCure by an external system that performs preliminary diagnosis. (This might include another instance of Symure – providing a mechanism to distribute the analysis). Then SymCure would perform further detailed diagnosis by identifying and running additional tests to resolve the suspect faults.

For example, multiple systems using different diagnostic techniques or different models with varying levels of abstraction could be managing a network. An entire network might be managed by a SymCure system, using high-level fault propagation models that would narrow down the root-causes to a set of faults in a particular sub-network. This result could then be sent as an ambiguity group to a SymCure system that is managing individual sub-networks, using detailed fault propagation models for further correlation and diagnosis.

In addition, managing huge network systems requires distributed implementations to handle the computational complexity. In the case of distributed implementations, the diagnostic information could be exchanged by passing ambiguity groups.

### *Specific Domain Model*

The Specific Domain Model (SDM) is an object-oriented model of the external managed system. An SDM might model physical equipment, as well as more abstract entities affecting system performance, such as controllers or software applications. The SDM includes domain object classes and their instances representing the managed domain entities, their connectivity, containment, and other relationships.

For example, if the domain model represents a company’s computer network,

domain objects might include hardware, network infrastructure, operating system level software, and other software applications running on the system. Examples of connectivity, containment, and other relationships that are important to fault propagation include passes-data, requires-service-from, or layered-over for protocols.

Specific domain models exist independently of SymCure. New SDMs can be created using the Integrity developer interface. In addition, SDMs already existing in other applications, external databases or files can be imported and merged in to a SymCure application.

### *Generic Fault Propagation Models*

Generic fault propagation models (GFPMs) consist of the generic cause-effect relations among the fault, symptom, and test events in the domain objects. A GFPM of a domain-object class defines the propagation of failures within its instances and to domain objects of other classes via relationships.

The GFPM of a domain object class contains the knowledge about:

- The faults that can occur in the domain object.
- The symptoms and tests that are the effects of the faults in that domain object and related objects.

The GFPMs can be developed from first-principles models, experience-based knowledge or Failure Modes and Effects Analysis (FMEA) results. The GFPMs can be constructed graphically in SymCure by creating the faults, tests, symptoms and their views, configuring and connecting them using causal-links. Appropriate propagation-relations must be specified in the causal-links connected to view nodes.

An event view represents a view of a corresponding event of the same name defined in the GFPM of another superior or sub class of object. Views represent a many-to-one relationship, which enable the propagation of a particular event throughout the GFPMs of various classes of objects. An event view node is used for modeling the propagation of events across related objects, from the GFPM of one class to another GFPM via a propagation-relation defined between those objects.

An action method can be defined for Symptoms and Tests to be executed automatically when the event receives a specified or inferred value. For example, this action method can be used to send an email, or message notifying an operator, when SymCure predicts the effects of a fault on the downstream symptoms/tests. The test, mitigation and recovery actions and the methods for calculating the cost of tests can be defined using OPAC procedures or G2 methods.

In addition, the event nodes and causal-links in the GFPM can be made to be conditionally dependent on the states of the domain objects by defining appropriate conditional methods on the nodes and links. During event correlation, SymCure evaluates these conditional methods, and only the nodes that are valid for the current states of the domain objects are used for correlation.

**GFPM Debugging and Analysis Tools**

SymCure provides a “configuration checker” debugging tool to check the GFPMs developed for an application for various types of configuration errors. In addiction, model analysis tools are provided to find the signature of a fault and to check the the detectability and isolatability of the faults, to determine if the GFPMs are adequate for diagnosis and fault isolation.

A fault signature consists of all of the effects in the form of symptoms and tests that, if observed, indicate that the fault is a root-cause failure. SymCure separates the signature into a symptom signature and a test signature. SymCure creates a fault signature, starting from the fault event, by traversing the GFPMs downstream across corresponding event and view nodes and vice versa.

A fault must have at least one test or symptom defined as its effect in the GFPM for it to be detectable. In addition, a fault's signature must be unique for the fault to be isolated. Thus, each fault should have at least one unique test or symptom effect in the model to isolate that fault from other faults. Having unique test effects of faults in the model is better for isolating faults as root-causes, since a test result can be obtained on request; whereas symptoms might not show up, depending on the monitoring frequency. SymCure’s isolatability analysis tool will detect and warn the users about faults with identical signatures that cannot be isolated during diagnosis.

### *Specific Fault Propagation Model*

The specific fault propagation model (SFPM) is a fault propagation model that describes the propagation of fault, symptom, and test events within and across specific domain objects in the specific domain model at runtime. SymCure constructs a specific fault propagation model, starting from the incoming events, by combining the GFPMs and the SDM of the managed system and creating specific event nodes and the causal links connecting them to account for possible causes and effects of the incoming events.

# 4.How SymCure Works

At runtime, SymCure receives various symptom, test, fault, and ambiguity group events. “Event Detectors,” which are external applications that monitor and analyze the numerical data trend in the managed system, detect and generate appropriate symptom events for input to SymCure.

Based on these incoming events, SymCure:

- Builds a specific fault propagation model (SFPM), using the generic fault propagation model (GFPM) and specific domain model (SDM) specifications for the managed system.

- Creates correlation managers to manage groups of correlated events.

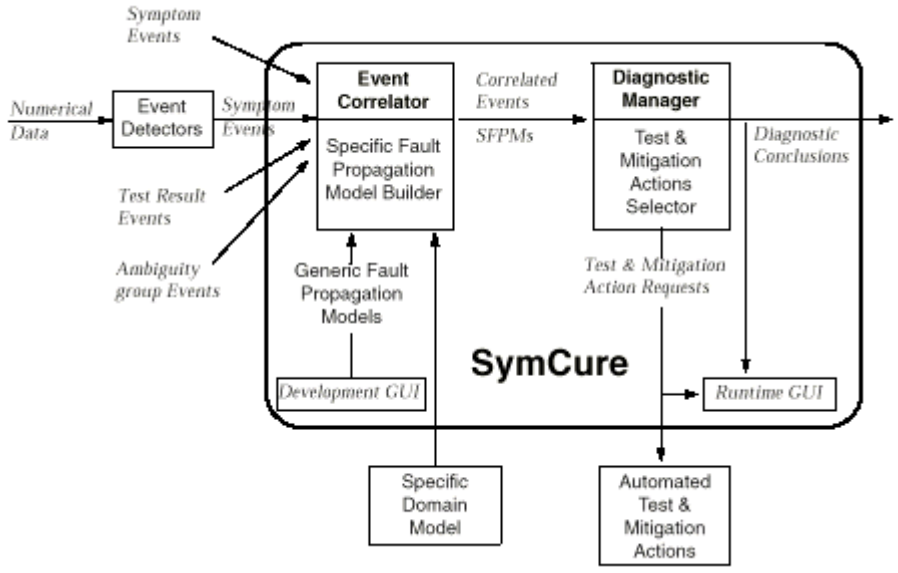
- Correlates the incoming events by propagating their values across the SFPM

• Identifies suspect faults, which are possible root-causes of the incoming events.

• Performs diagnosis by selecting and executing appropriate test and mitigation actions to resolve suspect faults.

The outputs from SymCure are the diagnostic conclusions in the form of identified root-cause faults, and test and mitigation action requests. When SymCure has determined the root cause(s), it runs mitigation procedures to recover from those faults. Correlation and diagnosis end when either all of the suspect faults are ruled-out/confirmed or no tests remain to isolate the faults. SymCure provides a runtime interface for viewing the status of the diagnosis progress, and run tests and mitigation actions.

The following diagram illustrates the components of SymCure, where any arrow crossing SymCure represents interaction with external systems:




### *Building the Specific Fault Propagation Model*

Starting from incoming events, the Specific Fault Propagation Model Builder constructs a specific fault propagation model (SFPM), which describes the propagation of the effects of faults to symptom and test events within and across specific domain objects in the specific domain model. SymCure constructs the SFPM by appropriately combining the generic fault propagation models (GFPMs) and the specific domain model as follows:

• Perform a linear breadth-first traversal of the GFPM in the upstream and

downstream directions, starting from the generic event corresponding to the incoming event.

- Create specific events that correspond to the generic events, where each specific event is created for a particular target domain object and connected via causal-links.

- Collect the domain objects related via the propagation relation and creating specific event nodes for those related domain objects and connecting the events via causal-links, whenever a generic event view is encountered in the GFPM.

The specific event nodes and the causal links between them account for all of the possible causes and effects of the incoming events.

### *Creating Correlation Managers*

SymCure creates correlation managers to manage the current diagnosis status of a group of correlated events. The correlation manager object class has two subclasses:

- The Diagnosis Manager, which is created to manage the correlation and diagnosis of a group of correlated incoming symptom, test, and fault events. There is a one-to-one relationship between a specific fault propagation model and a diagnosis manager.

- The Ambiguity-group Manager, which is created to manage an incoming ambiguity-group. Each unique incoming ambiguity group has its own ambiguity-group manager.

These correlation managers are the central repositories of correlation and diagnosis information, such as the faults that are currently suspected as the root-causes, the candidate tests to resolve those faults, and the known symptoms and tests. Users interact with SymCure and monitor the progress of diagnosis via the correlation managers. Access to the information in the correlation managers is available at runtime through the SymCure browser.

### *Correlating Incoming Events*

Using the specific fault propagation model, the Event Correlator correlates asynchronous symptom, test result, fault, or ambiguity group events input to SymCure. The event correlator recognizes that a group of events are related to each other, based on their connectivity criteria in the SFPM, such as the existence of a directed path or the fact that the events could be caused by common faults.

SymCure propagates the value of the incoming event in the SFPM by performing a breadth-first upstream graph traversal, starting from the specific node corresponding to the incoming event (if it has a new value), to infer the values of the upstream events and to identify the suspected faults.

SymCure also performs a downstream breadth-first graph traversal, starting from the specific node corresponding to the incoming event (if it has a new value), to predict the values of the downstream events.

During event propagation, whenever the value of an event is predicted or inferred by the event correlator, the information is stored in the correlation manager managing that event. This information and the SFPM is then input to the Diagnostic Conductor.

During event propagation, whenever the value of an event is predicted or inferred by the event correlator, the information is stored in the correlation manager managing that event. This information and the SFPM is then input to the Diagnostic Conductor.

### *Propagation Logic*

The value of a specific event can be a float value between 0.0 and 1.0, indicating a fuzzy truth-value where:

0.0 is false
1.00 is true
0.5 is unknown

SymCure propagates event values during correlation as follows:

- For an event with a single input event, the value of that event is equal to the value of its input event.

- OR logic downstream propagation: The value of an OR event equals the maximum value of all of its input events.

- AND logic downstream propagation: The value of an AND event equals the minimum value of all of its input events.

- OR logic upstream propagation: If the value of an OR event is > 0.5, the status of its input event becomes “Suspect.”

Specific events and causal links that are not active due to state-conditional dependencies are not considered for propagation.

Using this logic, propagation depends on the input/output connectivity status of the specific events being fully connected (the SFPM should have been completely built to/from the events). Thus, the SFPM is completely built from/to that event.

### *Identifying Suspect Faults*

The Event Correlator identifies the suspect faults by searching the specific fault propagation model upstream from the incoming fuzzy-true symptoms and test results with values between 0.5 and 1.0. Whenever a fault becomes a suspect, the information is stored in the correlation manager managing that event.

### *Performing Diagnosis*

If the available incoming symptoms cannot isolate the problem, the Diagnostic Manager tries to resolve the suspected faults by repeatedly executing the loop of:

- Identifying appropriate candidate tests.
- Requesting or automatically executing the tests.
- Propagating asynchronous test result values in the SFPM.
- Checking for the termination of diagnosis.

Candidate tests are those tests that are the effects of the suspect faults, which when executed, provide additional information regarding those faults. The Test Actions Selector identifies these tests by searching breadth-first downstream from the suspect faults in the SFPM. When a candidate test is identified, this information is stored in the correlation manager that is managing the test.

In the case of an automated test, SymCure sends a request to execute the automated test method. Otherwise, SymCure calculates the cost of the test and presents the test to the operator. Tests presented can be ranked by cost.

Operator interactions with SymCure via a runtime graphical user interface (GUI) include:

- Running the defined G2/OPAC test method for the test.
- Asking SymCure for an explanation of “why a test was selected as a candidate?”

The test results are asynchronously input back to the event correlator for further correlation to reduce the number of suspect faults.

When SymCure receives a test result, it stores the test result value of the performed test in the correlation manager that is managing the test. If the value of a candidate test is inferred by SymCure, based on the correlation of other incoming events, then the information that the test value has been inferred is also stored in the Correlation manager that is managing the test.

Based on the correlation of the test results, the suspected faults are either ruled out or concluded to have occurred. These diagnostic conclusions are output from SymCure to the operator and can be sent to other external systems. This information is also stored in the correlation manager managing the faults.

Whenever SymCure concludes that a fault is a suspect or has occurred, the Mitigation Actions Selector will execute appropriate suspect or fault mitigation actions specified for the fault. Similar to test methods, these mitigation actions can be automated or may require operator intervention. Mitigation actions requiring operator intervention are presented to the operator via a runtime graphical user

interface (GUI).

The diagnostic Manager checks for diagnostic termination of each group of correlated events managed by a correlation manager as follows:

- Termination of diagnosis based on no remaining “suspect” faults - all the suspected faults are proven either true or false. Multiple true faults are also possible.

- Termination of diagnosis due to lack of tests - there are remaining suspect faults, but no remaining candidate tests. In this case, the remaining suspects form an ambiguity group that can not be isolated any further.

SymCure can be configured to automatically delete the correlation manager and the associated specific fault propagation model, when diagnosis is complete and all faults identified as root causes are fixed, as indicated by a false value, due to corrective actions.

### *Incremental Model Building and Diagnosis*

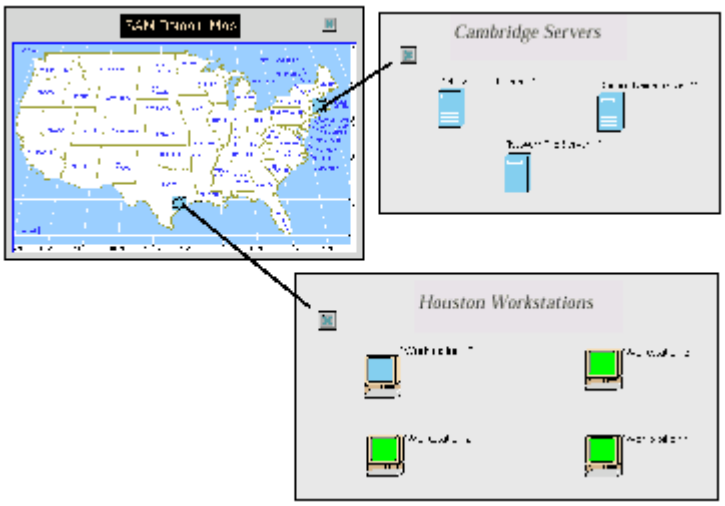
Starting from the incoming events, SymCure builds locally focused specific fault propagation models (SFPMs) to quickly diagnose the problem. The user can control the incremental SFPM construction by configuring SymCure’s model building parameters. In addition to the incremental model building that starts from each incoming event, if the specific model initially constructed is not adequate, SymCure further builds the model incrementally to complete diagnosis and to propagate the effects of root-cause faults.

# 5. The System Availability Management (SAM) Demo

The System Availability Management (SAM) demo provides an example of a SymCure application for managing e-mail services.

### The Specific Domain Models

The System Availability Management (SAM) demo domain map contains two specific domain models: one for the Servers in Cambridge and one for the work-stations in Houston. Both the server domain objects and the workstation domain objects are subclasses of the COMPUTER class in these specific domain models. A relation, named “the-sam-server-of,” is defined between the servers and workstations, as well as the inverse of this relation, named “sam-served-by.” The Houston workstations are related to the Cambridge servers via these relations.

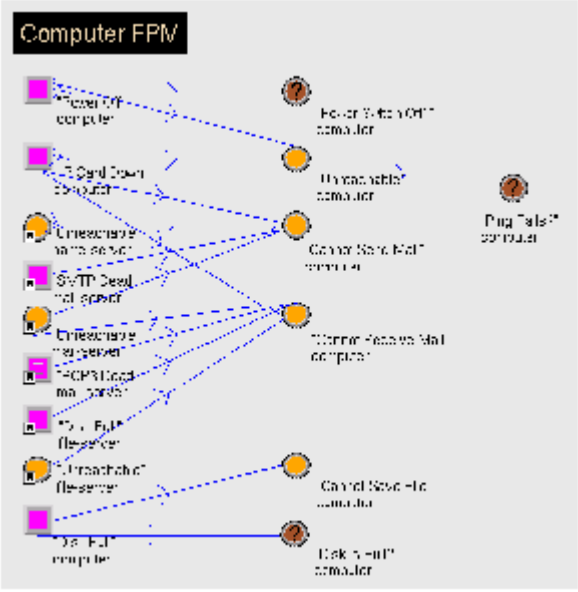



## The Generic Fault Propagation Models

The SAM demo contains two generic fault propagation models (GFPMs):
- GFPM for the Computer class
- GFPM for the Mail Server class

### The Computer GFPM

This is the GFPM for the computer:




In the GFPM for the computer:

- The fault “Power Off” (true) in a computer will cause the inferred-only

symptom “Unreachable” (true) and the test result “Power Switch Off” (true) in that computer.

• The fault “IP Card Down” (true) in a computer will cause the inferred-only symptom “Unreachable” (true) in that computer, and in this case will cause the test result “Ping Failed” (true) in that computer.

• The fault “Disk Full” (true) in a computer will cause the symptom “Cannot Save File” (true) and the test result “Disk Full Test” (true) in that computer.

Fault-views and symptom-views define the faults and symptoms of the related objects belonging to different classes in the computer GFPM. Since servers and workstations are subclasses of COMPUTER, they inherit the GFPM defined for the computer.

Faults and symptoms related to the Mail-Server, File-Server, and Name-Server will propagate to a computer via the sam-server-of relation. These faults and symptoms are defined by the fault-views and symptom-views in the Computer GFPM as explained below:

• The fault “SMTP Dead” (true) in a Mail-Server represented by a fault-view will cause the symptom “Cannot Send Mail” (true) in every computer served-by the Mail-Server.

• The fault “POP3 Dead” (true) in a Mail-Server represented by a fault-view will cause the symptom “Cannot Receive Mail” (true) in the computer served-by the Mail-Server.

• The fault “Disk Full” (true) in a File-Server represented by a fault-view will cause the symptom “Cannot Receive Mail” (true) in the computer served-by the File-Server.

• The inferred-only symptom “Unreachable” (true) in a Mail-Server represented by a symptom-view will cause the symptom “Cannot Receive Mail” (true) and “Cannot Send Mail” (true) in the computer served-by the Mail-Server.

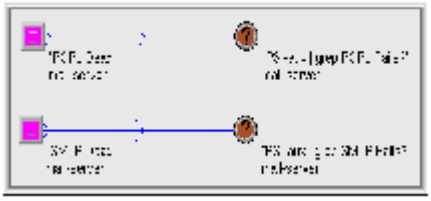
• The inferred-only symptom “Unreachable” (true) in a File-Server will cause the symptom “Cannot Receive Mail” (true) in the computer served-by the File-Server.

• The inferred-only symptom “Unreachable” (true) in a Name-Server represented by a symptom-view in the GFPM of the computer will cause the symptom “Cannot Send Mail” (true) in the computer served-by the Name-Server.

The causal-link connection of each fault-view and symptom-view in the GFPM for the computer specifies the propagation relation, the-sam-server-of.

**The Mail Server GFPM**

This is the GFPM for the mail server:



In the GFPM for the server class:

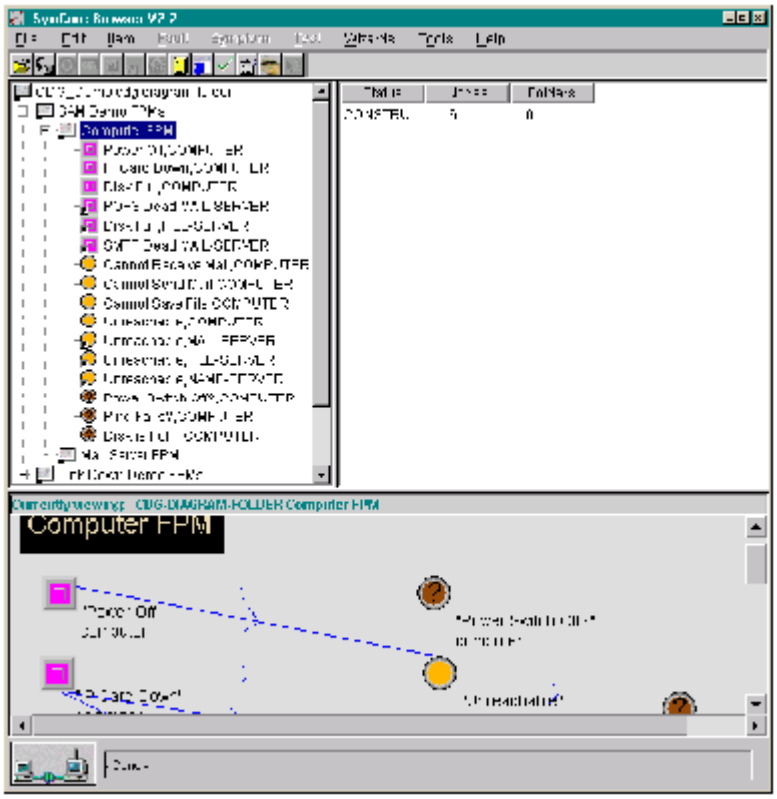
• The fault “POP3 Dead” (true) in a mail-server causes the test result “PS -aux | grep POP3 Fails?” (true) in that mail-server.

• The fault “SMTP Dead” (true) in a mail-server causes the test result “PS -aux | grep SMTP Fails?” (true) in that mail-server.

### SymCure Browser for Development

The user can create, view, configure, and interact with the GFPMs using the SymCure browser as shown below.

The browser tree expands to display the contents of the GFPM, and the model appears




in the object view section of the browser.

## SymCure Message Browser

A SymCure application by default uses three message servers, which the user defines, to create and store default messages related to:

- Incoming events
- Correlation managers
- Fault diagnosis

These servers contain messages generated by SymCure whenever it receives an incoming event, creates a Correlation manager, and arrives at a diagnostic conclusion, such as identifying the root-cause fault. A message browser displays the messages in one or more message servers. When the user starts a new application, a default SymCure message browser is created.

The SymCure message browser, named cdg_demo-cdg-browser, in the SAM demo subscribes to these SymCure message servers:

- CDG_demo Correlations Managers
- CDG_demo Diagnostic Messages
- CDG_demo Incoming Events

## Event Correlation and Diagnosis in the SAM demo

As an example, let us consider the case when SymCure receives a true value for the “Cannot Receive Mail” symptom against Workstation1 as an incoming event, via the cdg-accept-event API.

Whenever SymCure receives a symptom against a specific domain object, it first determines whether that symptom relates to a diagnosis that is already in progress. If not:

- SymCure creates a Diagnosis Manager to manage the correlation and diagnosis, and by default sends a message to a correlations-manager server, indicating that it has created a new diagnosis manager. In this example, SymCure sends the message to the Cdg_Demo Correlation Managers server.

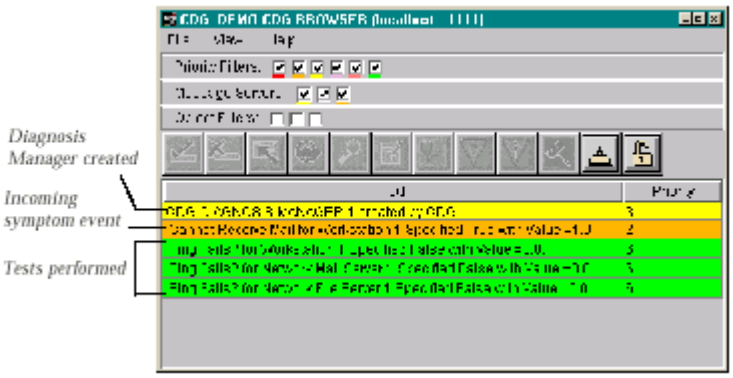
- SymCure then uses the generic fault propagation models and the specific domain map to construct the specific fault propagation model (SFPM), consisting of other correlated events, including the possible faults that could have caused the incoming symptom and the candidate tests whose results will either rule-out or confirm each of those faults.

- Based on the SFPM, SymCure then runs the automated candidate tests to isolate suspected faults. The default test result messages and incoming symptom messages from SymCure are sent to the Cdg_Demo Incoming Events server.

In this example, based on the SFPM constructed, the faults “Power Off” and “IP Card Down” in Workstation1, Network-mail-server-1 or Network-file-server-

1, are among the possible root-causes suspected by SymCure. SymCure then finds and automatically runs the “Ping Failed?” candidate test in the respective computers for those suspected faults.

These test results are false, as it can be sees from the test result messages on the cdg_demo-cdg-browser shown below:




If an incoming symptom or test result relates to a diagnosis that is already in progress, SymCure does not create a new diagnosis manager. Instead, it simply updates that existing diagnosis manager accordingly, propagating the new information through the already existing SPFM for that diagnosis manager.

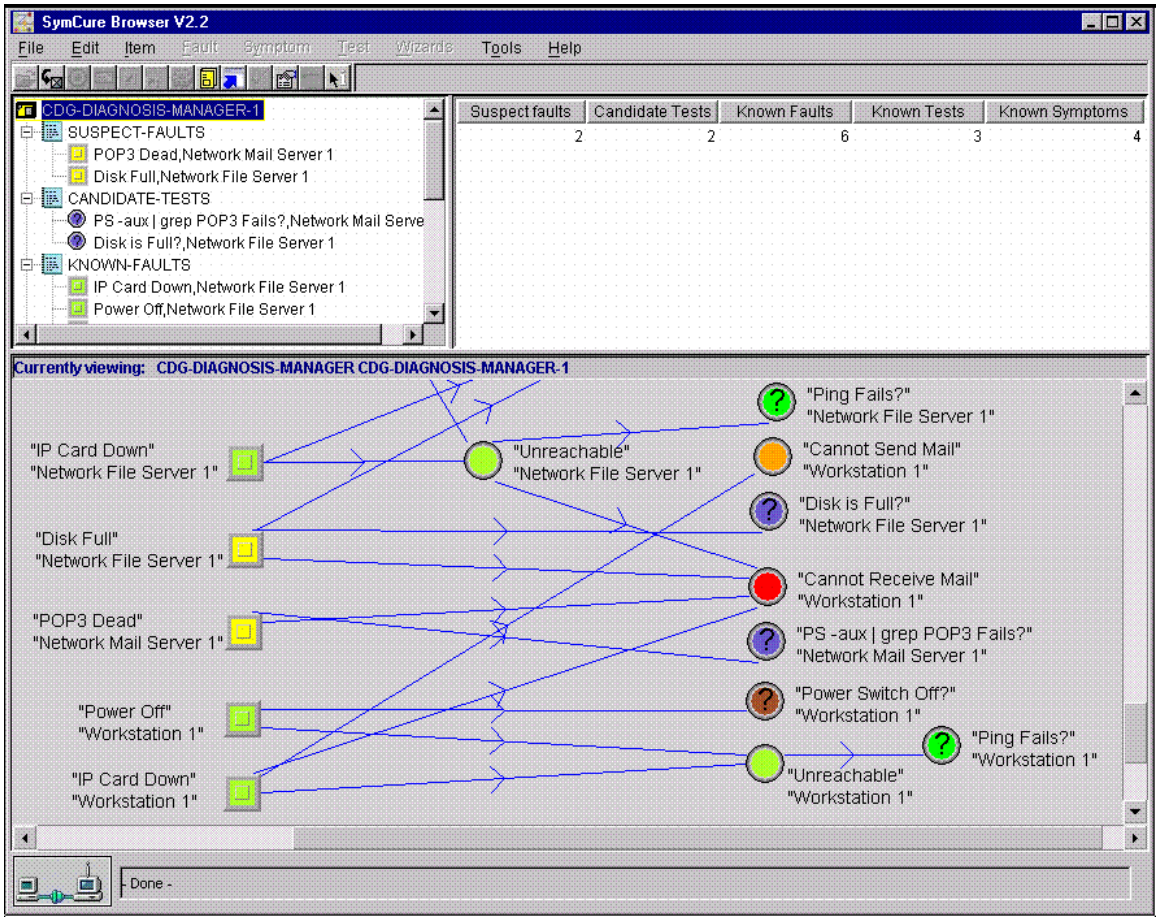
For example, the false test results for “Ping Failed?” tests leads to false values for the “Power Off” and “IP Card Down” suspect fault causes in respective computers due to propagation, and those faults are ruled out as possible root causes.

The user can view the status of the diagnosis and the specific fault propagation model, using the SymCure Runtime browser.

**SymCure Browser for Runtime**

The SymCure Runtime Browser is the interface to the diagnosis manager while SymCure performs event correlation, testing, fault isolation, diagnosis, and fault mitigation. The browser provides a view of the diagnosis status of a group of correlated events managed by a diagnosis manager by providing detailed information on:

- Known Symptoms - these are the symptoms with a known value.
- Suspected Faults - these are the suspected root-cause faults with unknown value upstream of an incoming symptom or test with a fuzzy-true value.
- Candidate Tests - these are the tests with unknown value downstream from the suspect faults.
- Known Tests - these are the tests with a known value.
- Known Faults - these are the faults with a known value.

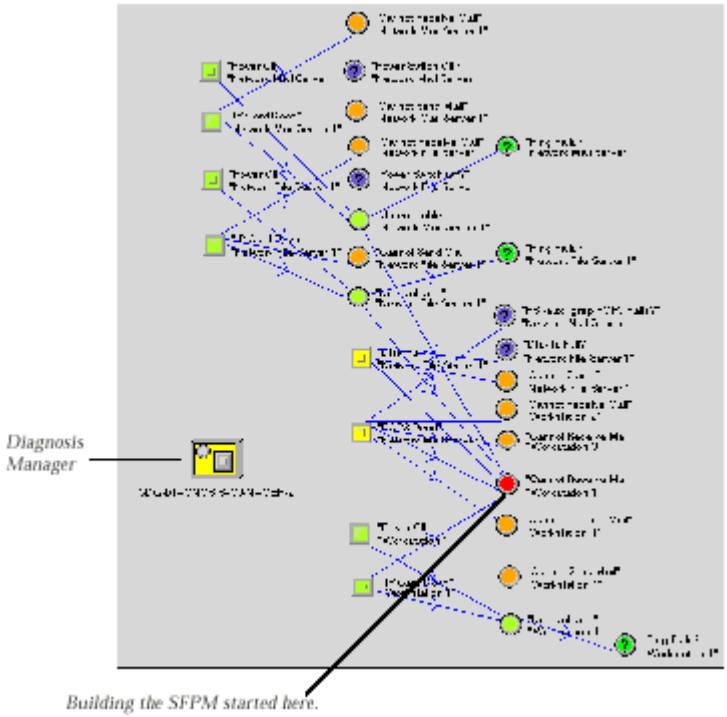



The SFPM built by SymCure, starting from the incoming symptom "Cannot Receive Mail" from Workstation1, and the diagnosis manager object created by SymCure to manage this diagnosis are shown below as they appear in the Object View area of the runtime browser.

**Graphical SFPM Colors**

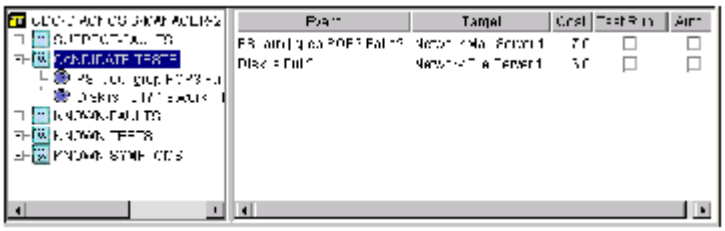
The default "Unknown" status region colors for a fault, symptom and Test are magenta, orange and sienna respectively. SymCure changes the status-region color of the specific events corresponding to their status during event value propagation as follows:

| **Color** | **Status Value** |
|---|---|
| Red | Specified true or fuzzy true. |
| Indian-red | Inferred true or fuzzy true. |
| Yellow | Suspect. |
| Green | Specified false or fuzzy false. |
| Green-yellow | Inferred false or fuzzy false. |
| Slate Blue | Candidate test or event value specified or inferred unknown. |




**Running a Candidate Test**

Candidate tests are those tests which are identified by SymCure to resolve the remaining suspect faults. The candidate tests for the current diagnosis as displayed in the SymCure browser are shown below:



For each candidate test, the information display includes:

- The name of the test event
- The target domain object.
- The cost of running the test.
- Whether the test is currently executing.
- Whether the test is automated or must be run manually.

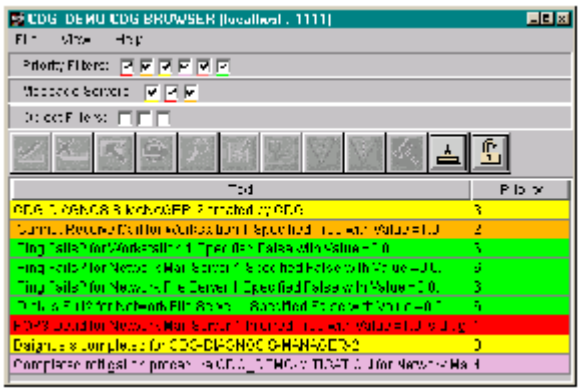
The user can run a Candidate test from the Test menu of the SymCure browser. For example if the test “Disk is Full?” in Network-File-Server-1 is executed and if the test result is false, SymCure infers that the:

- Suspect fault “Disk Full” in Network-Mail-Server-1 is false.

• Suspect fault “POP3 Dead” in Network-File-Server-1 is true and concludes that this is the root-cause fault.
• Candidate test “PS -aux | grep POP3 Fails?” in Network-File-Server-1 is true.

When the diagnosis is completed, SymCure executes the automated mitigation procedure defined for “POP3 Dead” fault for Network Mail Server 1.

On completion of the mitigation procedure, the message browser displays a message about the completed mitigation procedure as shown below:



The diagnosis manager tree in the SymCure browser reflects the completed diagnosis by moving the candidate tests to Known Tests and suspect faults to Known Faults.

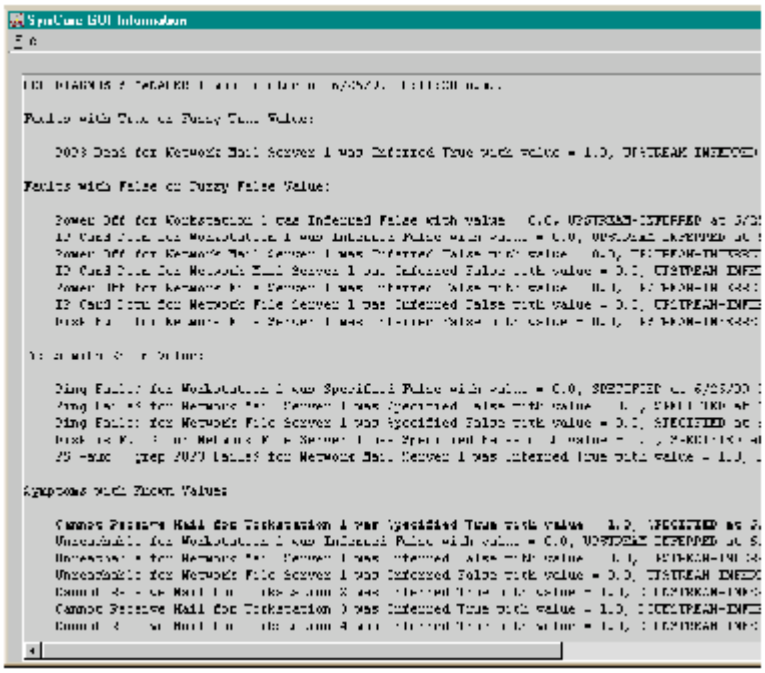
**Running a Mitigation Procedure**

Mitigation procedures perform a set of actions to resolve either a specific suspect or root-cause fault. These procedures, which are defined for the corresponding fault events in the generic fault propagation model, can be either OPAC procedures or G2 methods.

SymCure executes automated mitigation procedures. If the procedure is not automated, it is presented to the end-user, such as an operator, for manual execution. Mitigation procedures can be ranked, based on cost, in order to execute lower-cost mitigation methods or mitigation methods of faults with a high failure cost first. The mitigation procedures can be executed from the SymCure runtime browser.

## Viewing the Diagnosis Summary

The user can view a “diagnosis summary” of the diagnostic manager at any time from the SymCure runtime browser. SymCure displays the summary information in a separate window. For example this is the summary information related to a diagnosis manager in the SAM demo:



Similar explanation information help is available for other events displayed in the SymCure browser tree.

**Cleaning Up After the Diagnosis**

When the diagnosis is complete, the correlation manager can be deleted. When a correlation manager gets deleted, all of the events, the specific fault propagation model, and the diagnosis and correlation information associated with that manager is deleted. SymCure can also be configured to delete the correlation managers automatically when the diagnosis completes.

# 6. Conclusions

A generic fault propagation modeling approach has been developed for automating on-line event correlation and interactive diagnosis. The proposed modeling technique consists of novel concepts such as defining test and mitigation actions as part of the model and in-built state conditional dependencies. Based on this approach, a software product named SymCure has been developed for real-time fault and availability management in large-scale systems. SymCure is part of Gensym's overall Integrity product line for e-Infrastructure availability management.

SymCure appropriately combines generic fault propagation models with specific domain representation and builds focused specific models to investigate observed asynchronous symptom events. Using these specific models, SymCure recognizes that a group of events are correlated to each other, identifies suspected faults that could have caused the symptoms, and selects and executes candidate tests and mitigation actions to resolve the problems. SymCure also provides graphical user interfaces for the development of the models, and for operator interaction. The test and mitigation actions can be defined using graphical procedures in Integrity.