SymCure: A Model-Based Approach for Fault Management with Causal Directed Graphs

Conference Paper *in* Lecture Notes in Computer Science · June 2003

DOI: 10.1007/3-540-45034-3_59 · Source: DBLP

1 author:

Ravi Kapadia
General Atomics
17 PUBLICATIONS **121** CITATIONS

SymCure: A model-based approach for fault management with causal directed graphs

Ravi Kapadia
Gensym Corporation, 1776 Yorktown Ste 830, Houston TX 77056
rkapadia@gensym.com

Abstract. SymCure is an object-oriented graphical model-based fault management methodology that integrates automated and interactive fault diagnosis, testing, recovery, and impact prediction. SymCure allows domain experts to define class level fault models over generic events that are represented in the form of causal directed graphs. At run time, SymCure combines these generic fault models with incoming events and a specific domain representation, which describes a particular system configuration and relations among specific components, to diagnose root causes for abnormal system behavior. This methodology can be used for fault management in domains as diverse as enterprise wide communications and manufacturing processes. This paper describes SymCure's architecture, its diagnostic knowledge representation, fault management procedures, and discusses its contributions.

1 Introduction

Fault management plays a vital role across a broad spectrum of commercial and industrial applications, ranging from service level management and telecommunications network management in the Information Technology (IT) world, to abnormal condition management in manufacturing, chemical, oil and gas industries. The size and complexity of these applications often necessitates automated expert system support for fault management. A small number of root cause problems in IT communication networks often result in a large number of messages and alarms that cannot be handled by human operators in real time. Failure to identify and repair the root cause problems results in increased system downtime and poor service levels. Abnormal conditions in manufacturing and processing plants may result in unplanned shutdowns, equipment damage, safety hazards, reduced productivity, and poor quality products. The US National Institute of Standards and Technology (NIST) estimates that in the absence of adequate fault management, billions of dollars are spent in addressing the problems caused by equipment failure, degradation, process drift, and operator overload [1].

Fault management across these industries shares some common goals, such as improving application availability and utilization, reducing operator overload, and minimizing operation costs. In order to achieve these goals, it is necessary to develop fault management tools with the following capabilities.

1. *Symptom monitoring*. Symptoms are manifestations of underlying root causes and must be monitored to detect the occurrence of problems as soon as they happen.
2. *Diagnosis* identifies the root causes of known symptoms. (Diagnosis is also often referred to as fault isolation.) Some studies have shown that 80% of the fault

management effort is spent in identifying root causes after the manifestation of symptoms [2].

3. *Correlation* is the process of recognizing and organizing groups of events that are causally related to each other - usually such events share one or more root causes - for diagnostic inference and presentation to system operators.
4. *Prediction*. Early prediction of the impacts of underlying root causes before the effects are manifested is critical for proactive maintenance, safety, and optimal system utilization.
5. *Testing*. In large systems, it is impractical and sometimes impossible to monitor every variable. Instead key observable variables are monitored to generate symptom events. Diagnostic inference typically identifies a set of suspected root causes. Additional variables can then be examined by running associated tests to complete the diagnosis process.
6. *Automated recovery*. Identifying and automating recovery procedures allows for growth in equipment, processes, and services, without increasing the supervisory burden on system operators.
7. *Notification*. Operators must be notified of the presence of root causes and their potential impacts. Raw alarms, which can overload an operator with redundant information, must be replaced with concise diagnostic summaries of root causes and their impacts.
8. *Postmortem*. Information from the diagnostic problem solving is fed back to the fault management system for historic record keeping and proactive fault management in the future.

SymCure (derived from "Symptom Cure") is a tool that addresses a number of fault management functions, including diagnosis, correlation, prediction, testing, automated recovery and notification. It provides a powerful object oriented model-based framework to specify diagnosis knowledge in the form of a persistent, generic (i.e., class-level), graphical, fault propagation model library. It performs diagnosis and prediction by combining the fault propagation models with specific domain information and incoming events at run time. It detects and resolves multiple system failures, and notifies the results of its diagnostic reasoning to external systems using messages and other suitable means. SymCure's methodology is domain independent and it has been used for fault management in diverse applications across different industries, including abnormal condition management for heaters and service management for enterprise wide software systems[1].

This document describes the key elements of SymCure's architecture (Section 2), which includes diagnostic knowledge and diagnostic reasoning. Diagnostic knowledge is represented by a combination of fault models and fault management procedures (Section 3). Fault management algorithms use this knowledge to correlate events, hypothesize and verify root causes, predict impacts, and fix problems at run time (Section 4). The document concludes with a discussion of our contributions towards the development of intelligent fault management systems (Section 5).

---

[1] SymCure is the diagnostic reasoning engine for Gensym Corporation's Integrity and Optegrity products [3], which are directed towards fault management in the IT infrastructure and manufacturing process worlds, respectively.

2 Architecture

Fig. 1 shows the input, processing, and output elements of a SymCure application.

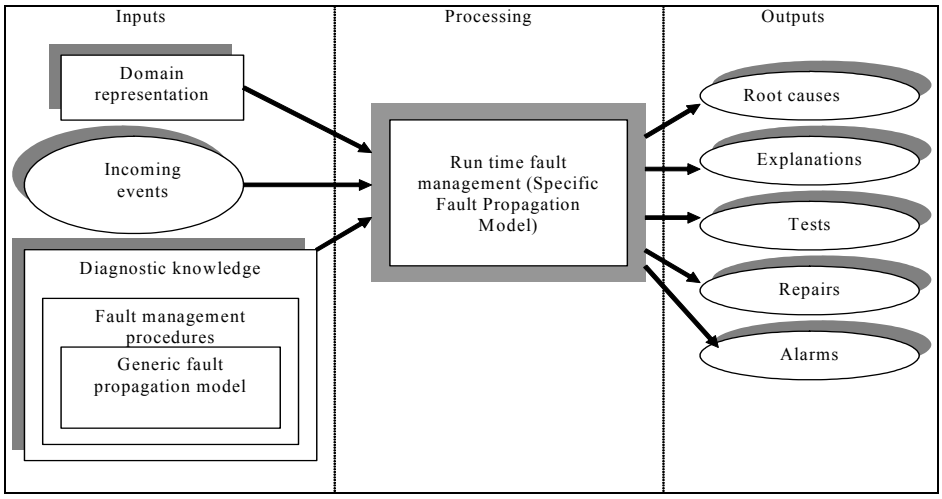

Fig. 1. Architecture of SymCure

Domain representation is a graphical object oriented representation of the domain objects that are being managed by SymCure. It includes class definitions for domain objects[2], their specific instances (i.e., the managed domain objects), and relationships between these instances, including connectivity (e.g., one object is connected upstream of another) and containment (i.e., one object is contained inside another). Fig. 2 shows the managed objects and their connections for a heating system. In this example, a furnace F-1 is connected downstream of two pumps P-1 and P-2.

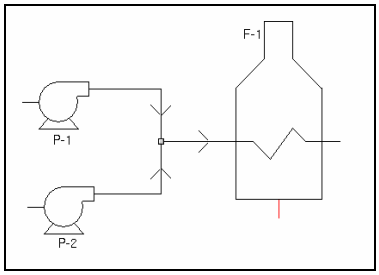

Fig. 2. Sample domain representation

The *incoming events* stream includes both symptoms that indicate the presence of problems and test results that validate or rule out suspected events. Symptoms are manifestations of underlying root causes on domain objects. They are detected by external procedures that monitor, aggregate, filter and analyze numerical data, sensor readings, network management traps, signals, and other forms of raw data to generate

[2] Integrity and Optegrity [3] contain a library of class definitions, icons, palettes, and behaviors that can be used to quickly create and configure a domain representation.

events for SymCure[3]. SymCure responds to the stream of incoming events with the following actions:

1. Diagnose the root causes of the incoming event stream.
2. Predict the impact of the root causes on other events.
3. Report the results of diagnosis and impact prediction to system operators.
4. Initiate any tests required to verify the occurrence of suspected root causes.
5. Initiate recovery actions to repair the target object of the root causes.

Diagnostic knowledge comprises of generic fault models that are used to reach diagnostic conclusions and procedures that verify and respond to these conclusions. Section 3 describes SymCure's representation of diagnostic knowledge.

Processing: At run time in response to incoming events, SymCure instantiates generic models for a specific domain representation by constructing a specific fault model. It uses the specific fault model for diagnosis, correlation, impact prediction, and to select and execute tests and repair actions. Section 4 describes SymCure's run time fault management process in detail.

Outputs: SymCure generates root causes, alarms, and explanations, and proposes tests and repair actions to resolve and recover from faults. Events can be configured to send notifications to operators. SymCure sends requests for tests and repair actions to a rudimentary workflow management component, which schedules and executes them.

3 Diagnostic knowledge

A key challenge for knowledge-based fault management expert systems is to develop powerful knowledge representation schemes that permit human experts to specify their domain knowledge. When there is sufficient understanding of failure modes of system components and their effects on the behavior of the system, diagnostic reasoning for fault management is based on fault models (e.g., [2], [4], [5], [6]). Alternative diagnosis techniques often described as consistency based methods (e.g., [7], [8]) use models of "normal" behavior, where diagnostic reasoning focuses on identifying causes for discrepancies between the normal behavior predicted by the model, and the aberrant behavior manifested by the device. Fault models can be far more abstract than models of normal behavior, and are therefore easier to construct. For example, fault models can easily capture causal relations such as "if an IP card fails, the device cannot communicate", and "if a pump fails, flow stops", without requiring detailed models that simulate normal behavior. Causal fault models capture paths of causal interactions between root causes (i.e., faults) and their effects (i.e., symptoms). Diagnosing root causes from known symptoms is achieved by tracing upstream along the causal pathways from the symptoms to the faults. Predicting the impact of root causes is performed by propagating downstream from causes to effects.

SymCure integrates two kinds of diagnostic knowledge for fault management.

1. Fault propagation knowledge uses Generic Fault Propagation Models (GFPMs) that capture fault propagation knowledge in the form of causal relations among events defined over generic classes. This knowledge is used for diagnosis, correlation, and impact prediction.

[3] Integrity and Optegrity [3] provide a combination of generic and domain specific tools to generate SymCure events from raw data.

2. Procedural knowledge supplements the fault models by specifying processes for responding to diagnostic conclusions and predictions. It includes test mechanisms for resolving suspected root causes and repair actions for recovering from identified, predicted, and suspected failures.

A GFPM for a class of domain objects defines the propagation of failures within its instances and to instances of other classes via generic domain relationships (e.g., Fig. 3. shows a GFPM for a generic furnace). Events may represent observable symptoms, alarms that are used to alert operators of potential problems, underlying failures (i.e., root causes), or may be introduced simply for modeling convenience. GFPMs are intended to be independent of any specific collection of domain objects and relationships actually present at any particular site. Thus the diagnosis knowledge can be made impervious to changes in system topology and operating modes and GFPMs can be reused across different applications. GFPMs can be inherited from parent classes in a class hierarchy in accordance with Object Oriented Programming principles. They can be developed from "first principles" models, expert knowledge, or Failure Mode Effects Analysis (FMEA) results.

GFPMs are stored in containers called *diagram folders*. Typically, there is one diagram folder for each domain object class definition, but generic events for a class definition may be distributed across different diagram folders. Propagation from an event in one folder to an event in another folder is achieved by *event views*, which act as bridges among different diagram folders.

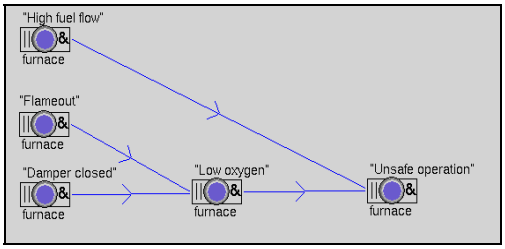

Fig. 3. Generic Fault Propagation Model for furnace

SymCure supports two types of fault management procedures which supplement GFPMs.
1. Tests are used to verify the occurrence of an underlying event.
2. Repair actions are used to recover from failures.

Fault management procedures have a set of associated actions that may be applied to the managed system. The actions may be automated, or may simply be a request to an operator, or a combination of the two. Such actions include "pinging" an IP address to test for network connectivity, extracting a data point from a database, and even sending a repair technician to a remote site to conduct manual tests and repairs. Upon completion, a test action must return a result (true or false) to SymCure.

The specification for a fault management procedure has three parts:
1. An attribute specification that includes its name, its target domain object, the conditions for activating the procedure, and the resources, costs, and information necessary for scheduling the procedure.

2. A relation linking it to an event in the GFPM (the procedure is invoked in response to suitable state changes of an underlying event).
3. A procedural specification (written in any computer programming language) that lays down the sequence of actions that must be performed to obtain the result for a test or to fix a problem.

SymCure provides a graphical user interface to create, configure, and compile GFPMs, and to create tests and repair actions, configure them, and relate them to generic events.

Figs. 3 and 4 show sample fault propagation models for a furnace and a pump, respectively. A *generic event*, represented by a node in the GFPM, applies to a class of objects. It is uniquely identified by a combination of its name (e.g., "Pump overload") and its target class (e.g., pump)[4]. A directed edge between two events specifies a *causal relation* (the direction of the edge specifies the direction of causality). In Fig. 4 the event "Pump overload" causes the event "High temperature" on any pump. An edge may also specify a propagation relation between instances of the target class of the cause and the target class of the event. For example, consider the causal relation between the event "Pump overload" on any pump and the event view "High fuel flow" on cdg-domain-object. (Cdg-domain-object is the parent class for all target objects.) The view acts as a bridge to a corresponding event defined in Fig. 3. The propagation relation "connected-upstream" on the edge specifies that "High fuel flow" on any domain object is caused by "Pump overload" on any pump that is connected upstream of the domain object.

"Test pump overload" and "Repair pump overload" are a generic test and a generic repair action, respectively. Both are associated with the generic event "Pump overload". (These associations are not depicted in the figure to avoid clutter.) "Test pump overload" may be run whenever it is suspected that "Pump overload" may have occurred for any instance of a pump. The test may involve extracting a sensor reading from a database or perhaps scheduling a visit by an operator to the pump to do a manual check. "Reset pump" may be executed whenever it is detected that "Pump overload" has occurred for any instance of a pump. This may be achieved simply by sending a control command to the pump or perhaps by ordering the operator to manually twist a knob on the device.

Causal propagation may be characterized by the logical relationships between a target event and the set of events that can cause the target to occur. Traditional causal modeling approaches (e.g., [2], [4], [5], [6]) use one or both of the following forms of causal propagation: *disjunctive* propagation (in which a target event is true if any one of the events in its set of causes is true — this is often referred to as OR logic) and conjunctive propagation (in which the target event is true if every one of the events in its set of causes is true — this is often called AND logic). These forms of propagation are unsuitable for a variety of real world applications that require more subtle reasoning. For instance, an event may be true only if a certain fraction of its causes are true, e.g., internet service providers often have a number of backup servers and internet service degradation may occur only when over a certain fraction of the servers go out of service. Noisy operating environments can mask the symptoms of a root cause causing misdiagnoses, unless the

[4] In this document, whenever the target of an event is obvious, we refer to the event by its name only.

fault model can capture this behavior. Causal propagation may depend on the state of the system (e.g., propagation through a switch may depend on its position – ON or OFF) in addition to the causes of the event. SymCure overcomes a number of problems encountered in traditional causal modeling methodologies by providing for a wide range of causal propagation logic to fit many real world diagnostic situations that arise from incomplete models, uncertain propagation, and context dependent failure propagation. Further discussion of SymCure's propagation logic is beyond the scope of this paper; a detailed description is provided elsewhere [9].

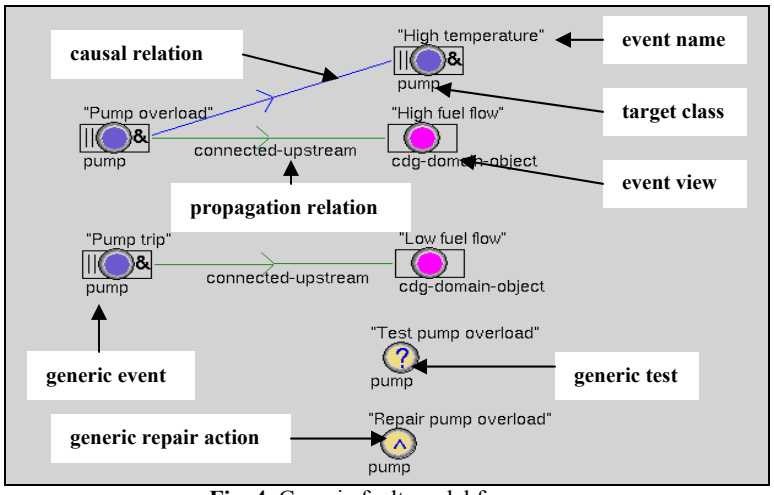

Fig. 4. Generic fault model for pump

4 Run time fault management

SymCure's fault management algorithms respond to incoming symptoms by hypothesizing and identifying root causes, predicting their impacts, running tests and repair actions, and notifying operators. At the heart of fault management is a causal model constructed from the generic fault propagation models and specific domain objects. Diagnosing root causes from known symptoms is achieved by tracing upstream along the causal pathways from the symptoms to the faults. Predicting the impact of root causes is performed by propagating downstream from causes to effects.

SymCure combines the generic fault models with the domain representation and builds focused Specific Fault Propagation Models (SFPMs) to investigate observed symptoms. Using the SFPMs, SymCure recognizes that a group of events are correlated to each other, identifies suspect faults that could have caused the symptoms, and selects and executes tests and repair actions to resolve the problems. The SFPM describes causal interactions among events within and across the specific domain objects. It also captures the current state of the diagnostic process and can be used to generate explanations for symptoms, diagnostic conclusions, and tests and repair actions. For the sake of efficiency, in response to an incoming event SymCure builds the minimal set of specific events upstream of the event to diagnose possible causes and downstream of the event to predict impacts.

Like generic fault models, SFPMs are also represented as causal directed graphs. A specific event, represented by a node in an SFPM, is a statement about a specific target object and is uniquely identified by the combination of its name and target object. An event in an SFPM can have the following values:

1. true, i.e., the event is known to have occurred;
2. false, i.e., the event is known to not have occurred;
3. unknown, i.e., it is not known whether the event has occurred; or
4. suspect, i.e., it is suspected that the event may be true.

SymCure uses heuristic best first search to propagate event values for an SFPM. At a very high level, starting from an incoming event, the logic for propagating event values in an SFPM from an incoming event is as follows.

for any event *e* when its value changes **do**
propagate the value of the event upstream to all *Causes* (where *Causes* = all causes of *e*);
propagate value of the event downstream to *Effects* (where *Effects* = all effects of *Causes* + *e*);
end for

The time complexity of the propagation algorithm is linear in the number of events and edges of the SFPM. The maximum number of events in an SFPM is bound by product of the number of managed domain objects and the size of the largest generic fault model. In practice, since SymCure constructs only the events that are correlated to incoming symptoms, the actual size of an SFPM is usually a small subset of the maximum possible size.

Fig. 5 shows the SFPM created by combining the domain representation of Fig. 2, the generic fault models of Figs. 3 and 4, and the symptom "High fuel flow" on F-1. A *root cause* event in an SFPM is an event that is not caused by any other event (e.g., "Pump overload" on P-1) while a *symptom* (e.g., "High fuel flow" on F-1) is an event that is caused either directly or indirectly by a combination of one or more root cause events. In this example, SymCure initially treats "Pump overload" on both P-1 and P-2 as suspect and executes "Test pump overload" for each suspected event, which leads to the identification of "Pump overload" on P-1 as a root cause.

SymCure is able to reason over relations among different objects and across different GFPMs to build an SFPM. Thus, SymCure can correlate a symptom on F-1 to its potential causes on P-1 and P-2. Note that it only builds the events relevant to the diagnosis for "High fuel flow", ignoring "Pump trip", "Low oxygen" and other events that are not required for diagnosing the causes of the incoming event.

Adding new equipment, reconnecting or removing existing equipment does not affect SymCure's generic fault models. In response to dynamic domain object changes, if an SFPM exists as a result of an ongoing diagnosis, SymCure can dynamically update it by adding or deleting the necessary events and edges.

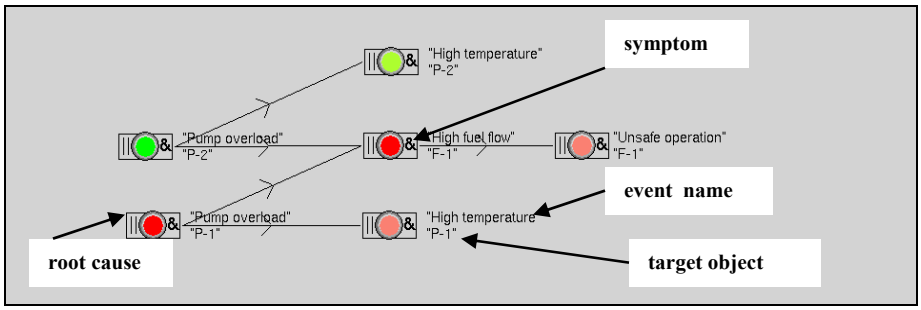

Fig. 5. Specific Fault Propagation Model (SFPM)

5 Discussion

SymCure is an object-oriented, graphical, model-based fault management methodology that performs automated and interactive fault diagnosis, testing, recovery, and impact prediction. It has been implemented in G2, Gensym Corporation's graphical object-oriented platform for building expert system applications. It has been used for fault management in diverse applications including offshore drilling platforms [9], heating systems [10], and enterprise wide software systems [11]. This section discusses SymCure's strengths, limitations, and lessons we've learned from applying the technology to real world problems.

Diagnosis is essentially an iterative process that generates hypotheses in response to incoming symptoms and refines the hypotheses by obtaining additional information (through tests and additional symptoms). SymCure integrates test and repair management with fault diagnosis which allows the fault management system to optimally obtain additional information to identify and repair root causes. Unlike diagnosis techniques that implicitly assume that there is a single point of failure (e.g., [5]), SymCure can handle multiple simultaneous failures and situations where new faults overlap with existing ones. Complex systems are often composed of a large number of interconnected and interrelated components. While a problem may originate on one component, often it is manifested on some other related component. SymCure's specific fault models are built at the system-level, which permits it to diagnose root causes and predict events that propagate across components. SymCure's fault management methodology is scalable for large complex systems because it employs an efficient diagnosis algorithm with linear complexity, and "management by exception" to only instantiate a localized specific model at run time when incoming events indicate the existence of a problem. SymCure overcomes the problems encountered in traditional causal modeling methodologies by providing for a wide range of causal propagation logic to fit a number of real world diagnostic situations that arise from incomplete models, uncertain propagation, and context dependent failure propagation.

Like any knowledge based reasoning system, the accuracy of SymCure's diagnostic inference is constrained by the correctness of the underlying fault models and the availability of instrumentation to observe symptoms and perform tests to resolve diagnostic candidates. Obviously, SymCure cannot handle faults that are not part of the fault models but it can identify novel combinations of known faults. Because SymCure

processes an event as soon as it is received, diagnostic results are susceptible to the order of incoming events. In theory, this can cause problems over short time spans if the values of observed events are inconsistent (because of propagation delays, noise, and faulty sensors). In practice, this has not been a significant issue. SymCure allows domain experts to represent events in their terminology at an arbitrary level of abstraction suitable to their application. However, the burden of detecting the occurrence of an event is placed on external monitoring mechanisms, which may require sophisticated filtering and aggregation techniques. SymCure does not explicitly model propagation delays or the likelihoods (i.e., probabilities) of events.

Not surprisingly, our greatest challenge in building SymCure applications has been in acquiring fault models. The bulk of this effort is in getting domain experts to specify their diagnostic knowledge in terms of causal fault models rather than inverse causality rules of the form "if this symptom, then conclude that fault". (We believe that causal fault models are better suited to bi-directional fault management than rule based approaches, which require separate sets of rules for diagnosis and impact prediction that may interfere with other leading to circular conclusions [12]. Furthermore, as the size of the rule base increases, it becomes increasingly difficult to maintain consistency.) Building reusable applications requires constructing fault models in a manner such that they are completely independent of system topologies. Like good software engineering practice, this requires a lot of discipline on the part of the application developer and is easier said than done.

References

1. Siegel, D.: Abnormal Condition Management: Minimizing Process Disruptions and Sustaining Performance through Expert Systems Technology. In: Natl. Petroleum Refiners Assn. 2001 Comp. Conf., Dallas, USA (2001)
2. Stanley, G., Vaidhyanathan, R.: A Generic Fault Propagation Modeling Approach to On-line Diagnosis and Event Correlation. In: Proc. of the 3rd IFAC Workshop on On-line Fault Detection and Supervision in the Chemical Process Industries, Solaize, France (1998)
3. Integrity SymCure Developer's Guide. Version 3.6 Rev 0. Gensym Corporation (2002)
4. Porcheron, M., Ricard, B.: Model-based diagnosis for reactor coolant pumps of EDF nuclear power plants. In: Proc. of the 10th Intl. Conf. IEA/AIE 97, Atlanta, USA (1997) 411-420
5. Kliger, S., Yemeni, S., Yemeni, Y., Ohsie, D., Stolfo, S.: A Coding Approach to Event Correlation. In: Proc. of the 4th Intl. Symp. on Integrated Network Management (1995)
6. Finch, F., Oyeleye, O., Kramer, M.: A robust event-oriented methodology for diagnosis of dynamic process systems. Comp. and Chem. Engg. 14 (12) 1379 (1990)
7. Biswas, G., Kapadia, R., Yu, X.: Combined Qualitative-Quantitative Diagnosis of Continuous valued Systems. IEEE Sys. Man, and Cybernetics. Vol 27, No. 2 (1997) 167-185
8. Hamscher, W., Console, L., de Kleer, J. (eds.): Readings in Model-based Diagnosis. Morgan Kaufman (1992)
9. Kapadia, R.: Causal modeling for fault management with SymCure. Gensym Corporation Technical Report (2003)
10. Noureldin, H., Ruveta, F.: Using Expert System and Object Technology for Abnormal Condition Management. BIAS 2002 Intl. Conf. Milan Italy (2002)
11. Warpenburg, M., Stanley, G., Vaidhyanathan, R.: PATROL ROOT CAUSE ANALYSIS: A New Standard for Event Automation in an Enterprise Computing Environment. Gensym Corporation Technical Report (1999)
12. Rich, E., Knight, K.: Artificial Intelligence. 2nd ed. McGraw-Hill (1991)