Real World Model-based Fault Management

Ravi Kapadia
General Atomics, 16969 Mesamint Street, San Diego, CA 92127
Greg Stanley
Greg Stanley and Associates, 8619 Tranquil Park Drive, Spring, TX 77379
Mark Walker
General Atomics, 16969 Mesamint Street, San Diego, CA 92127

Abstract: Real world fault management applications encompass a number of diagnostic activities such as symptom monitoring, root cause analysis, impact prediction, testing, and recovery. They motivate powerful knowledge representation schemes to capture domain expertise and the development of intelligent algorithms that can exploit this knowledge. There are vast opportunities for the application of state-of-the-art fault management in commercial settings and, with billions of dollars at stake, industries are eager to embrace intelligent knowledge based solutions. Over the past decade, we have developed an object-oriented model-based domain-independent methodology for real world fault management, called SymCure. In this paper, we use this experience to generalize a set of requirements for real world fault management. We present an overview of the architecture and the modeling language of SymCure. We review a sample of projects where we have applied this approach, and share the motivations, challenges, successes and failures that have been our companions along this memorable journey.

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. A study funded by 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 [7].

There is increasing demand for the application of state-of-the-art fault management across a broad spectrum of industries. Over the past decade, we have applied model-based reasoning to address several fault management functions, including diagnosing root causes, testing them, predicting their impacts, and recovering from failures. This experience has guided us in the development of an object-oriented model-based domain-independent world fault management methodology, called SymCure (derived from "Symptom's Cure").

Section 2 describes our requirements for addressing large scale commercial fault management applications. Section 3 describes SymCure's architecture, reasoning algorithms, and modeling language. Our diagnostic methodology is largely domain independent and its applications range from abnormal condition management for offshore platforms that drill for oil at the bottom of the ocean to network management for satellite systems in space. Section 4 describes some of these applications. Section 5 concludes with a discussion of the lessons we have learned while applying our model-based diagnostic methodology in the real world.

2 Background

Fault management across various 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.

Symptom monitoring. Symptoms are manifestations of underlying root causes and must be monitored to detect the occurrence of problems as soon as they happen.

Diagnosis identifies the root causes of known symptoms. (Diagnosis is also often referred to as fault isolation.) Complex systems are often composed of a large number of interconnected and interrelated components (e.g., networks of computers or satellites, electrical power generation plants, offshore oil drilling platforms). While a problem may originate on one component, often it is manifested on some other related component. In large-scale systems, it is not uncommon for multiple failures to overlap. Studies on such systems have shown that typically up to 80% of the fault management effort is spent in identifying root causes after the manifestation of symptoms [10].

Correlation. Modern systems are often richly instrumented with a large number of sensors that provide copious amounts of information in the form of messages and alarms. Often a small number of root causes result in a large number of messages and alarms that cannot be handled by human operators in real time. Therefore it is necessary to provide them with concise notifications of underlying root causes. Correlation is the process of recognizing and organizing groups of events that are related to each other. Usually such events share one or more root causes.

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. System operators need to know not just what impacts are predicted by the application but also when those impacts are expected to occur so they can plan for appropriate system repair and recovery.

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. A test planning facility is needed to select additional variables to be examined to isolate the root causes. The fault management application then needs to request or run these tests, and utilize their results to complete the diagnosis. A test, as originally defined in [8], can incorporate arbitrarily complex analysis and actions, as long as it returns a true or false value.

Automated recovery. Identifying and automating recovery procedures facilitates rapid response to problems and allows for growth in equipment, processes, and services, without increasing the supervisory burden on system operators.

Notification. System operators require notifications of all critical fault management activity, especially the identification of root causes, and causal explanations for alarms, tests, and repair actions in a manner that they can follow easily. Sometimes they need to distinguish between what is observed by system sensors versus what is inferred by the underlying fault management application.

Postmortem. Information from diagnostic problem solving is fed back to the fault management system for historic record keeping and proactive fault management in the future.

24 hour, year-round fault management. The system topology may change at run-time over the life span of the fault management application, e.g., components may be added, removed, replaced, and modified. Commercial applications often require fault management on a 24 hour, year-round basis, so it may not be feasible to take the fault management system off-line each time that there is a change in the system topology.

A goal of ours for the past decade has been to develop a domain independent model-based fault management methodology to address these requirements.

With some effort at knowledge elicitation, it is often feasible to develop a high-level qualitative understanding of failure modes of system components and their effects on the behavior of the system. Such knowledge facilitates diagnostic reasoning based on qualitative fault models (e.g., [3], [5], [10], [11]). Alternative diagnosis techniques often described as consistency based methods (e.g., [1], [6]) use models of "normal" behavior, where diagnosis 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 can be easier to construct, comprehend, and customize to a domain expert's specification. 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.

In a number of commercial applications, such knowledge may be gleaned from domain experts, Failure Modes Effects and Analyses (FMEA) studies [9], and operational and product manuals. Domain experts need tools that allow them to model and analyze the structure and the diagnostic behaviors of their system. Object oriented graphical causal fault models facilitate understanding, capturing, updating, and reusing key elements of their diagnostic knowledge.

Our methodology, as detailed in the following section, utilizes graphical object-oriented qualitative causal fault models as the basis for several fault management functions, including diagnosis, correlation, prediction, testing, automated recovery, and notification.

3 SymCure Methodology

SymCure is implemented in G2 which is Gensym Corporation's graphical, object-oriented platform for developing and deploying expert systems applications.

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

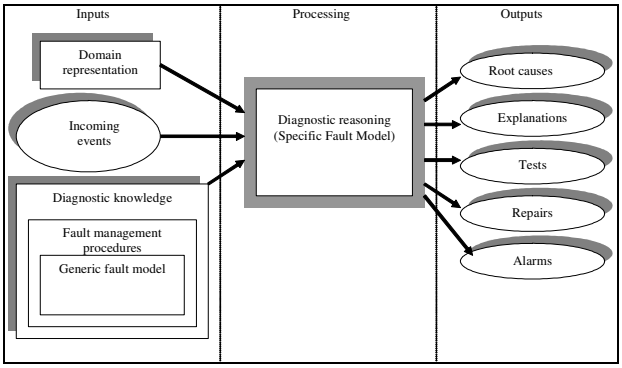

Figure 1. SymCure architecture

Domain representation is a graphical object oriented representation of the domain objects being managed by SymCure. It includes class definitions for domain objects, and a system description (also referred to as a *domain map*) comprised of 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). Figure 2 shows two simplified illustrations of domain maps that are comprised of similar components (a furnace and a pair of pumps) but differ in their connectivity (i.e., Domain map 1 connects both pumps directly to the furnace, while Domain map 2 connects the pumps to the furnace in series).

Diagnostic knowledge comprises a declarative component, i.e., generic event-based fault models that are used to reach diagnostic conclusions, and a procedural component, i.e., test procedures that verify these conclusions, and recovery procedures that respond to these conclusions. We believe that separating out the declarative aspect of diagnostic knowledge (i.e., how do things fail, how do failures propagate) from the procedural aspect (i.e., what to do when something is suspected to have occurred, what to do when something is known to have failed) facilitates the articulation, acquisition, representation, and management of domain expertise.

A SymCure event[1] is a statement about a domain object that indicates the presence or absence of a problem. Generic fault models are composed of events (that are defined at the class-level) and their causal relations. Domain experts define these events using terminology they are familiar with, and at levels of abstraction suitable to their application. The fault models are graphical, thus they are easy to understand and little or no programming experience is necessary to build them. Another advantage of this approach is that the diagnostic knowledge does not require any reconfiguration whenever there are changes in specific instances of equipment, system topology (as illustrated by the two domain maps of Figure 2), or system operating modes, and they can be reused easily in different applications.

SymCure identifies a generic event as a unique combination of event name and target class (e.g., in the generic fault model for a pump in Figure 2, "Low flow" is the event's name and PUMP is its target class). Causal relations are represented by edges between generic events. Figure 2 shows simplified examples of generic fault models for pumps and furnaces. In the furnace generic fault model, low fluid flow into a furnace causes the temperature of the fluid to rise. At high temperatures, the fluid may undergo undesirable chemical reactions that, over time, cause carbon deposits inside the furnace tubes (i.e., fouling). Damage to a pump's impeller hinders its ability to impart motion to the fluid in the pump, thus causing low fluid flow through the pump. Inlet strainers are devices that keep debris out of a pump that might otherwise damage or clog it. Low flow through the pump is also caused by a plugged inlet strainer, which is also responsible for low inlet pressure. In these simplified models, low fluid flow into either the pump or furnace is caused by low flow in any domain object connected upstream of the pump or furnace. Propagation delays may be specified by configuring the properties of causal links; such delays are used to infer the occurrence time for any event. Although this is not shown in Figure 2, for illustrative purposes, we have assumed that there is a 3 hour delay between the onset of rising temperature in a furnace and the onset of fouling in the furnace.

The procedural component of a fault model includes the following elements.

1. Tests are used to verify the occurrence of an underlying event. Upon completion, a test action must return a true or false value for an associated event.
2. Repair actions are used to recover from failures.

Tests and repair actions are associated with underlying events. They are enabled or disabled by transitions of the values of their associated events. For example, in Figure 2, the test "Check inlet pressure" may be used to determine if the "Low inlet pressure" on a pump is true; and the repair action "Unclog inlet strainer" may be used to set in motion the process whereby a plugged inlet strainer is unclogged. In general, such actions may be automated, or may simply be a request to an operator, or a combination of the two, e.g., extracting a data point from a database or sending a repair technician to a remote site to conduct manual tests and repairs. A detailed discussion of tests and repair actions is beyond the scope of this paper.

[1] For the rest of this paper, we refer to a SymCure event simply as event.

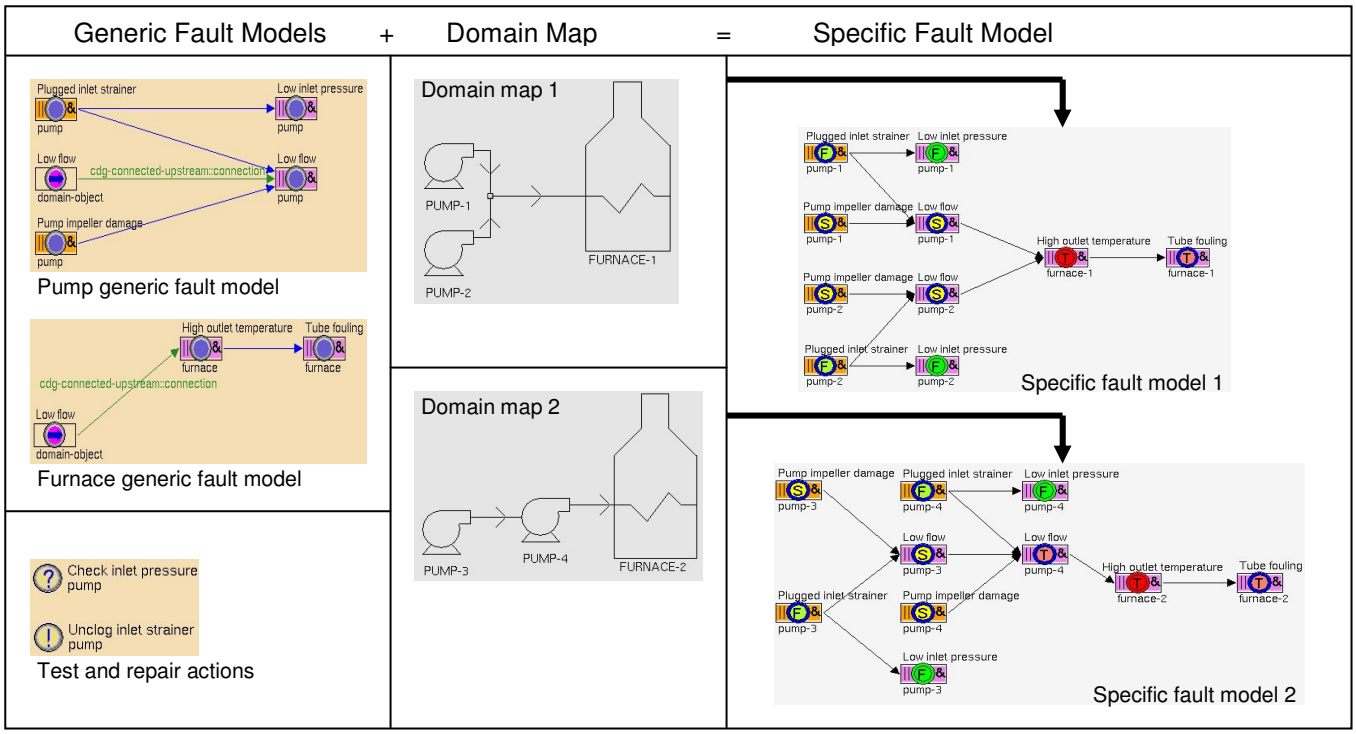

Figure 2. Generic fault models, domain maps, and specific fault models

The *incoming events* stream includes symptoms that indicate the presence of problems. Symptoms are manifestations of underlying root causes on specific domain objects. They are detected by application specific procedures[2] that monitor, aggregate, filter and analyze numerical data, sensor readings, network management traps, signals, and other forms of raw data. SymCure responds to an incoming event by diagnosing the root causes of the incoming event stream, predicting the impact of the root causes on other events, reporting the results of diagnosis and impact prediction to system operators, initiating any tests required to verify the occurrence of suspected root causes, and initiating recovery actions to repair the target objects of the root causes. Incoming events also include test results that validate or rule out suspected events.

SymCure's diagnostic algorithms dynamically combine the domain representation, diagnostic knowledge, and incoming events to produce a *specific fault model* that applies to the specific managed domain objects. A specific fault model is composed of specific events and their causal relations. In Figure 2, specific fault models 1 and 2 are derived from domain maps 1 and 2, respectively, and the generic fault models for pumps and furnaces. SymCure uniquely identifies a specific event by its name (e.g., "Low flow") and the domain object on which the event occurs (e.g., PUMP-1). For diagnostic reasoning purposes, specific events may take on the following symbolic values:

- true, i.e., the event is known or inferred to have occurred; such an event is depicted with "T" in the specific fault model (e.g., "High outlet temperature" on FURNACE-1 in Figure 2),
- false, i.e., the event is known or inferred to not have occurred; such an event is depicted with "F" in the specific fault model (e.g., "High outlet pressure" on PUMP-1 in Figure 2),
- unknown, i.e., it is neither known nor inferred that the event has occurred; such an event is depicted with "U" in the specific fault model, and
- suspect, i.e., it is suspected that the event has occurred; such an event is depicted with "S" in the specific fault model (e.g., "Pump Impeller damage" on PUMP-1 in Figure 2).

The specific fault models in Figure 2 are initiated by assuming that high outlet temperature is observed in each furnace and manual testing shows the inlet pressures on all pumps are normal. For each specific fault model, SymCure infers that the impellers on one or both pumps must be damaged. For each scenario, SymCure also predicts that tube fouling will occur.

We use best first search to propagate event values within a specific fault model. At a very high level (for details refer to [3]), starting from an incoming event, event propagation is achieved by the following algorithm.
for any event *e* if its value changes **do**

[2] Event detection is outside the scope of this paper; refer to [3] for further details.

1. propagate the value of the event upstream to all *Causes* (where *Causes* = all causes of *e*);
2. propagate the value of the event downstream to *Effects* (where *Effects* = all effects of {*Causes* + *e*});

end for

This for-loop is invoked each time a symptom is manifested and the result of a test becomes available. The propagation steps in the for-loop can be configured to terminate as soon as SymCure identifies a set of root causes that explain all observed incoming events, even before a complete specific fault model has been constructed. To speed up the search process, in steps 1 and 2 SymCure explores events on the same domain object before it moves on to events defined on different domain objects. The time complexity of the for-loop is linear in the number of events (i.e., size) of the specific fault model. The maximum number of events in a specific fault model is bound by the product of the number of managed domain objects and the size of the largest generic fault model. In practice, since we construct only the events that are correlated to incoming symptoms, the actual size of a specific fault model is usually a small subset of the maximum possible size.

As demonstrated in Figure 2, SymCure constructs system wide specific fault models, i.e., it is able to reason about interactions among the individual system components. We generate a visual representation of the underlying specific fault model on demand. This allows modelers and system operators to understand the results of the diagnostic reasoning process. However, graphical models can rapidly overwhelm users as the number of nodes increase. Thus, we provide options to organize the visual representation of a specific fault model by focusing on relevant portions of the graph at any given time. For example, the abridged specific fault model in Figure 3 shows only the potential causes and effects of "High outlet temperature" in FURNACE-1.

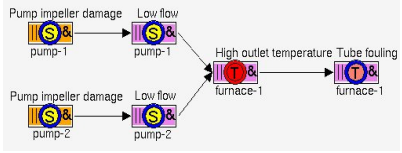

Figure 3. A focused view of Specific fault model 1

SymCure generates messages for root causes and alarms, and proposes tests and repair actions to resolve and recover from faults. Such messages are presented to system operators in one or more diagnostic console browsers. Figure 4 shows the alarms, root causes, tests, and repair actions for Specific fault model 1 in Figure 2. Note that tube fouling is predicted to occur 3 hours after high outlet temperature in the furnace has become true. Alarm messages can be suppressed in favor of root causes so that operators are not flooded with alarms. Requests for tests and repair actions may be sent automatically to a rudimentary workflow management component, which schedules and executes them.

Severity	Message
ALARM	Tube fouling on furnace-1 will become true (downstream inferred) at 2/23/2007 13:14:18
ALARM	High outlet temperature on furnace-1 has become true (specified) at 2/23/2007 10:14:18
ALARM	Low flow on pump-1 is suspect (upstream inferred)
ALARM	Low flow on pump-2 is suspect (upstream inferred)
ALARM	Low inlet pressure on pump-1 is false (specified)
ALARM	Low inlet pressure on pump-2 is false (specified)
REPAIR-ACTION	Unclog inlet strainer on pump-1 is CREATED
REPAIR-ACTION	Unclog inlet strainer on pump-2 is CREATED
ROOT-CAUSE	Pump impeller damage on pump-1 is suspect (upstream inferred)
ROOT-CAUSE	Pump impeller damage on pump-2 is suspect (upstream inferred)
ROOT-CAUSE	Plugged inlet strainer on pump-1 is false (upstream inferred)
ROOT-CAUSE	Plugged inlet strainer on pump-2 is false (upstream inferred)
TEST	Check inlet pressure on pump-1 is INACTIVE
TEST	Check inlet pressure on pump-2 is INACTIVE

Figure 4. Alarms, repair actions, root causes, and tests for Specific fault model 1

SymCure's modeling language. Early versions of SymCure provided two kinds of events for creating fault models: OR event and AND event. Their behaviors were governed by the following rules (where X_i, Y_j, and Z_k are events in a fault model; the edge $X_i \rightarrow Y_j$ implies that X_i causes Y_j; and propagating downstream requires determining a value for Y_j from X_i, while propagating upstream requires determining X_i from Y_j).

For any OR event Y_i

1. While propagating downstream, if $Y_i \rightarrow Z_1,\ldots,Z_n$, and Y_i is true, then $Z_1,\ldots,Z_n$ are all true.
2. While propagating upstream, if $X_1,\ldots,X_m \rightarrow Y_j$, and Y_i is true, then at least one of $X_1,\ldots,X_m$ must be true.

For any AND event Y_i:

1. While propagating downstream (identical to the case of an OR event), if $Y_i \rightarrow Z_1,\ldots,Z_n$, and Y_i is true, then $Z_1,\ldots,Z_n$ are all true.
2. While propagating upstream, if $X_1,\ldots,X_m \rightarrow Y_i$, and Y_i is true, then all of $X_1,\ldots,X_m$ must be true.

The limitations of a causal modeling language comprised solely of these two events were quickly exposed in several early projects. In the real world, fault models are rarely ever complete, i.e., frequently there are causal influences that are not fully understood or cannot be modeled. Propagation delays and noise often result in discrepancies between inferences made by the underlying fault model and observed events. This may result in root cause events being exonerated prematurely or being implicated falsely. We have tried to address these issues in later versions of SymCure by associating OR and AND logic at the input and output of an event as shown below:

1. While propagating downstream, if $Y_i \rightarrow Z_1,\ldots,Z_n$, and Y_i is true, then $Z_1,\ldots,Z_n$ are all true (we say that Y_i uses output AND logic).
2. While propagating downstream, if $Y_i \rightarrow Z_1,\ldots,Z_n$, and Y_i is true, then at least one of $Z_1,\ldots,Z_n$ must be true (we say that Y_i uses output OR logic).
3. While propagating upstream, if $X_1,\ldots,X_m \rightarrow Y_i$, and Y_i is true, then at least one of $X_1,\ldots,X_m$ must be true (we say that Y_i uses input OR logic).

4. While propagating upstream, if $X_1,\ldots,X_m \rightarrow Y_i$, and Y_i is true, then all of $X_1,\ldots,X_m$ must be true (we say that Y_i uses input AND logic).

We can also specify input and output logic with percentages to reason over a progression of values that may represent partial degradation. By combining these rules, we have created 7 different events are capable of root cause analyses and impact predictions that cannot be modeled with traditional OR and AND events alone. Further modeling enhancements, including utilizing NOT logic, mutual exclusion, and specifying state dependent behavior. See [3] for detailed examples of SymCure's capabilities and modeling language.

4 Applications

SymCure's diagnostic knowledge representation and reasoning methodology is domain independent and it has been applied to solve problems in domains ranging from networks of telecommunication satellites to offshore oil drilling platforms. We present a sample of such applications emphasizing, wherever applicable, the challenges they posed to our methodology and its consequent evolution.

Satellite network fault management (1995-2000). Iridium is a global telephony network of low earth orbit satellites and ground stations where calls are switched in the sky from satellite to satellite; as the satellites orbit the earth, communication links are periodically broken and are re-formed with different satellites. Diagnosing the root causes of communication failures required reasoning over dynamically changing communication links, so Motorola (Iridium's original owner) adopted an early version of SymCure for its satellite network fault management application. Experience with live monitoring before satellite launches, prior to full-scale deployment suggests that, had the fault management application been fully deployed, it might have saved some satellites that were lost just after launch [11]. To the best of our knowledge, the application was in use until 2000. Later, beset with financial difficulties and other business problems, Iridium ran out of funds to support the application.

Enterprise-wide monitoring of networks and applications (1998-2000). BMC built a line of products (aimed at Windows 2000 and Microsoft Exchange servers) for diagnosing faults and predicting their impacts on enterprise-wide computer networks and applications. The requirement to automatically diagnose and predict events over dynamically changing network topology and software installed on servers, led them to use SymCure as their fault management reasoning engine. This work tested some of the limitations of earlier versions of SymCure's modeling language because of timing delays along paths of causal propagation (primarily due to the lag times between measurements and their time-averaged thresholds), noise, and incomplete causal models.

Highway traffic management equipment monitoring (2001-02). L.E.E. developed a SymCure application to monitor and manage Paris' highway traffic equipment including thousands of road sensors, message displays, cameras, telecommunications equipment, hierarchical subsystems, energy systems, and supporting network infrastructure. Diagnostic conclusions are required within a short (2 minute) event polling cycle, of which only a fraction (approximately 20 seconds) is available for diagnostic processing. Early versions of SymCure built visual representations of complete specific fault models and reasoned over them. Drawing and updating events at run-time is inefficient and contributed to the difficulties in meeting the performance requirements. We realized that sound software design is as vital as "intelligent" technology for a successful application. In subsequent versions of SymCure, visual representations are separated from underlying specific fault models; they are generated on demand only and usually focus on a narrow section of a specific fault model as illustrated by Figure 3 in the previous section. Along with the introduction of best first search and efficient data structures like hash tables instead of lists, this has resulted in a five-fold improvement in the speed of diagnostic reasoning.

Heaters in oil refineries (2001-02). A middle eastern oil refinery installed an application for managing abnormal conditions, focused on diagnosing failures in a generic class of heaters. The application defined 80 root causes that account for over 240 different kinds of operator messages, and provided tests and repair actions that rapidly guide operators to return a heater to normal operation. Noureldin and Roveta [4] describe this application in detail including how they acquire domain knowledge from human experts. They concluded that the net savings from using this application substantially exceeded the cost of the project in less than one year.

Offshore oil production platform (2002-03). Halliburton KBR created an application for monitoring the health of offshore oil production platforms. Their application focused on monitoring the health of the gas compressors on a platform and demonstrated its ability to proactively predict compressor failures that could potentially save millions of dollars that would otherwise be sacrificed to production losses resulting from compressor shutdowns [2].

Mineral processing plant (2005-present). SGS MinnovEX is developing an application to diagnose and respond to root causes of sub-optimal operations in mineral processing plants. System components include equipment such as compressors and pumps, controllers, and other sub-processes. The application analyzes

historical operational data to detect deviant operational events and then diagnoses the causes of such events. This application uses SymCure's enhanced event logic and preliminary results indicate success in identifying faults such as sensor drift and poor controller tuning in the face of incomplete models and noisy data. However enhancing the event logic complicates the behaviors of the fault models. This motivated us to build a graphical fault model debugger to help domain experts to step through, analyze, and test their fault models.

Electrical power and energy distribution systems (2005-present). General Atomics is developing a generic fault management library for mission critical power and energy distribution systems. Though a significant degree of redundancy is typically designed into critical power systems, the penalties associated with loss of availability are so high that any software assistance in the detection of the onset of failure becomes invaluable. Generic fault models have been developed for electrical system components including generators, motors, circuit breakers, transformers, uninterruptible power supplies, transfer switches, critical load centers, and DC links. These models enable the diagnosis of complex power system anomalies such as ground faults, which typically cause a flood of alarms. The strengths of generic fault modeling, in conjunction with dynamic instantiation of energy flow relationships between electrical components, have effectively diagnosed problems in dynamically reconfigurable power systems. Combining SymCure's fault management methodology with models of normal behavior that analyze trends and detect discrepancies between observed and predicted sensor values, facilitates the prediction of the onset of equipment failure. SymCure's graphical fault models facilitate conveying, communicating, and validating the details of vendor-supplied Failure Modes and Effects Analysis (FMEA).

General Atomics' fault management library serves as the basis for several ongoing projects with the US Navy, which is increasing its reliance on high-energy electromagnetic components for next generation shipboard systems. This includes projects to build electromagnetic systems to impart the momentum necessary to launch airplanes from an aircraft carrier and that arrest the motion of the planes to bring them to a halt when they land. (The runway in an aircraft carrier is too small for airplanes to be able to take off and land without such assistance.) As availability and reliability are critical for such systems, General Atomics is leveraging SymCure's fault management methodology to diagnose and predict the health and life expectancy of aircraft launch and landing gear components.

General Atomics is also collaborating with the National Aeronautics and Space Administration (NASA) for fault management of its Rocket Engine Test Stand (RETS). RETS tests the structural integrity of a rocket before it is launched, and typically provides NASA's engineers their last chance to detect and correct any flaws in the fully assembled rocket. The fault management application is targeted for deployment at NASA's Stennis Space Center facility in southern Mississippi. The subsystems associated with the testing of solid and liquid fueled rocket engines involve complex mechanical, electrical, hydraulic, pneumatic, and thermodynamic processes. All of these processes, along with their associated system components, lend themselves well to the generic fault modeling capabilities of SymCure. The fault management application will be used to diagnose and predict RETS anomalies, but will also provide real-time advisories associated with the quality of results obtained from the RETS tests.

5 Conclusions

In this paper, we have described a generic model-based fault management methodology to address fault management applications in several diverse domains. We summarize our contributions to the application of model-based diagnostic techniques in the real world, limitations of our methodology, and share our thoughts on the lessons we have learned.

Contributions. SymCure integrates causal fault models for root cause analysis and impact predictions with test and repair management thus providing a comprehensive set of capabilities for developing fault management applications. SymCure's modeling language has evolved over time to address problems with propagation delays, sensor noise and threshold problems. SymCure's reasoning process is capable of detecting and resolving multiple system failures. SymCure's fault models are generic, so they do not require any reconfiguration whenever there are changes in equipment, system topology, or system operating modes and they can be reused in different applications.

Limitations. 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. 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 the likelihoods (i.e., probabilities) of events.

Lessons learned. There are tremendous and exciting opportunities for the application of state-of-the-art fault management techniques in commercial settings and, with billions of dollars at stake, industries are eager to embrace intelligent knowledge based solutions. However, as we have learned over the years, there are many challenges and pitfalls of which practitioners need to be aware.

The fundamental challenge for knowledge based expert systems applications is the formalization of the domain knowledge both before and during the development of the application. Often the knowledge base evolves incrementally, starting from a small prototype that grows with further testing and development as new requirements become evident. Commercial applications require ease of use so they can be mastered with minimal training and cost. This motivates not just the development of powerful user interfaces, but also modeling languages that can be understood by users whose expertise is limited to their domain. Modeling aids to efficiently search for, navigate to, and debug diagnostic knowledge are essential. If such an application is to be adopted successfully in a commercial setting, it must be able to demonstrate a reasonable return on investment. There is a premium on "out of the box" solutions, where the diagnostic knowledge is pre-packaged. However, it is imperative that this knowledge be configurable by end users, since we have never seen two domain experts ever agree on everything. Visual representations and explanations are a crucial ingredient for success; a picture is worth not just a thousand words but potentially millions of dollars, if it helps system operators to avert an impending problem. Efficient computational techniques and sound software engineering principles are as important as the artificial intelligence that underlies its reasoning capabilities. Last, but not the least, human factors, including sky high expectations, funding cuts, resistance to adopt new technology, political infighting, and employee turnover (especially, if it involves someone who is a champion of the technology) are, not infrequently, serious impediments to success.

As described in the previous section, we have plumbed the depths of oceans (with offshore oil production platforms) and orbited the heavens (with telecommunication satellites). We believe that the road ahead promises to be just as exciting. As the future unfolds, we are embarking on the development of generic fault model libraries of components in different domains that can be reused across applications and other innovative techniques to automate knowledge discovery and to create adaptive fault models that learn from successful and failed diagnoses.

References

1. G. Biswas, R. Kapadia, and X. Yu. "Combined Qualitative-Quantitative Diagnosis of Continuous valued Systems". In IEEE Sys. Man, and Cybernetics. Vol 27, No. 2, pp. 167-185, March 1997.
2. R. Guddeti. "An Expert System for Real-Time Monitoring of Offshore Platform Equipment". GUS 2003 Gensym User Society Worldwide Meeting, Boston, Massachusetts, April 2003.
3. R. Kapadia. "SymCure: A model-based approach for fault management with causal directed graphs", Proc. of the 16th Intl. Conf. IEA/AIE-03, pp. 582-591, Loughborough, UK. June 2003.
4. H. Noureldin and F. Roveta. "Using Expert System and Object Technology for Abnormal Condition Management", BIAS 2002 International Conference, Milano, Italy. November 2002.
5. M. Porcheron and B. Ricard. "Model-based diagnosis for reactor coolant pumps of EDF nuclear power plants". Proc. of the 10th Intl. Conf. IEA/AIE 97, pp. 411-420, Atlanta, USA, June 1997.
6. Readings in Model-based Diagnosis. W. Hamscher, L. Console, J. deKleer, eds. Morgan Kaufman, 1992.
7. D. Siegel. "Abnormal Condition Management: Minimizing Process Disruptions and Sustaining Performance through Expert Systems Technology". Natl. Petroleum Refiners Assn. 2001 Comp. Conf., Dallas, Texas, October 2001.
8. W. Simpson and J. Sheppard. "System Test and Diagnosis". Kluwer Academic Publishers, Boston 1994.
9. D. Stamatis. "Failure Modes and Effect Analysis", ASQ Quality Press. 2003.
10. G. Stanley and R. Vaidhyanathan. "A Generic Fault Propagation Modeling Approach to On-line Diagnosis and Event Correlation". Proc. of the 3rd IFAC Workshop on On-line Fault Detection and Supervision in the Chemical Process Industries, Solaize, France, June 1998.
11. R. Stewart, G. Stanley, and V. Vasudevan. "Integrated Fault Management in a Satellite-Based Global Telecommunications Network", Software Engineering Symposium 1995, Ft. Lauderdale, Florida, June 1995.