State based Diagnostics of System Internal Fault Origins by using Boolean Rules

Peter Tondl, Patrick Rammelt, Ulrich Siebel
Carmeq GmbH
[Peter.Tondl | Patrick.Rammelt | Ulrich.Siebel]@carmeq.com

Abstract

The following paper introduces a diagnostic concept which is able to find the origin cause of an observed failure pattern by applying Boolean rules. The suggested method allows for and benefits from combining failure notifications of several observers, expressed by indicators. Failure notifications about observed objects are consolidated to one diagnostic result in order to provide a maintenance system with the best fitting failure explanation pattern.

Kurzfassung

Das Betreiben technischer Einrichtungen und Systeme in einer Form, die es ermöglicht alle zugedachten Funktionen erwartungsgemäß und sicher auszuführen, erfordert leistungsfähige und systemspezifische Überwachungs-, Diagnose- und Wartungsprozesse. Diese Prozesse werden in aller Regel parallel zu den operativen Funktionen ausgeführt. Sie sind sowohl für das Erkennen ungewollter oder unerwarteter Systemzustände verantwortlich, als auch für die Lokalisierung der Fehlerursache im System und deren Kommunikation nach außen. Die Definition einer umfassenden Überwachung, Diagnose und Wartung schließt die Forderung nach Zuverlässigkeit, Wartbarkeit, Sicherheit und die Erfüllung von Benutzeranforderungen ein.
Der folgende Artikel stellt ein auf Bool'schen Regeln basierendes Diagnosekonzept vor, das in der Lage ist, die systeminterne Ursache eines von außen beobachtbaren Fehlers sicher zu ermitteln. Die vorgeschlagene Methode kombiniert dabei Zustandsmeldungen verschiedener Beobachter zu einem Gesamtbild. Fehlermeldungen über beobachtete Objekte können in ein gemeinsames Diagnose-Resultat konsolidiert und für effiziente Wartungs- und Reparaturvorgänge in dafür jeweils optimalen Abstraktionsgraden kommuniziert werden.

1 Introduction

The operation of equipment belonging to a certain system in a manner that it performs all of its intended functions requires monitoring, diagnostics and maintenance processes adapted to this system and is usually running parallel to its operational functions. These processes are used and responsible for detecting unwanted or unexpected states as well as for isolating and communicating their origins. The definition of comprehensive monitoring, diagnostics and maintenance processes has to fit terms like reliability, maintainability, safety and user requirements in order to decrease system down times and life cycle costs.

2 Basic Principles

Figure 1 shows the relationship between a monitoring task at the beginning and maintenance at the end of the process chain. Start and end tasks are completed by confirmation, isolating and reporting. Each predecessor task has to provide its respective successor with all information needed to provide an efficient and successful operation of the entire chain.

2.1 Monitoring

A basic condition for detecting and isolating faults – which mean the inability to perform a required function and the requirement of unscheduled maintenance action to correct this inability [1, 2] – and

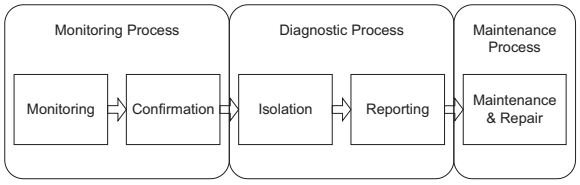

Figure 1: Monitoring, Diagnostics and Maintenance Task Chain

failures – which mean the termination of the ability of an item to perform its required function [1, 3] – is an adequate working monitoring. Monitoring itself means the process of continuously, periodically or on-demand scanning of a defined set of equipment functions. Often, at least regarding to complex electronic and mechanical systems, monitoring is realized as a software function hosted on internal or external equipment. However, this is not necessarily true in any case. The term "monitoring" is much more general and stands for a lot of different kinds of automatically or manually performed actions in order to determine the current state of a monitored system.

Continuous and periodic monitoring is usually performed during normal operation of the monitored unit under the condition that the responsible monitoring task must not disturb operational functions. For units where this requirement cannot be met monitoring has to be performed at a particular time when operational functions are not activated, e.g. between power-up and operational function activation. On-demand monitoring can be seen much more uncritical than the other types described before. Nevertheless it fulfills an important part of the whole process, e.g. in case of a failure confirmation or when it is performed under testing conditions.

2.2 Confirmation

A monitoring function is responsible for detecting unwanted or unexpected states according to its monitored area. It transmits its observations to its process level successor. In order to increase the reliability of these detections and to avoid false alarms a confirmation of an observed state shall be done. Such a confirmation can be realized, e.g. by using a monitored area depending time delay. During this time a failure pattern must not change or disappear. On the other hand, time cannot be used for confirmation if a fault or failure is the result of a singular event. Moreover, a repetition is often not necessarily representative, e.g. when a fault can cause an observation only in a specific, not reproducible context. In such cases it is recommended to activate a self-test dependent on the observation under safe conditions.

In case of latched failures – which mean failures where a disappearance cannot be detected – a confirmation test has to be most efficient and reliable to be "acceptable" when a confirmation result indicates that the failure cannot be confirmed. Thus, failure latching should not be used unless it is fully required for safety reasons.

A confirmation process allows an interpretation of observations with a certain level of confidence. However, if an observation cannot be confirmed, there shall be no further consequences on the system as well as no message transmissions to the next process level.

2.3 Isolation

A single fault can cause multiple observations, i.e. a complex failure pattern. Ideally, an isolation process taking all observations into account leads to an identification of exactly this single fault. A fault isolation algorithm should therefore take advantage of all information available, e.g. system design knowledge, in order to produce acceptable results.

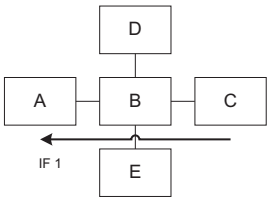

Figure 2: Isolation by Observation – one information flow IF1

Figure 2 shows a simple system where each component $comp = A, B, C, D, E$ is able to communicate with each other. Wirings between these components are not considered in the following descriptions.

Example 1 (Isolation – one information flow IF1) Assume there is only one information flow IF1 at time where component C transmits data packets to component A via component B. In case of an interruption, i.e. lost or damaged data packets, each component of the IF1 chain can cause an observed failure pattern "No valid data packets received from C" observed by A. Thus, A, B and C can be the cause of the fault and have to be presented by a failure explanation result. □

Example 2 (Isolation – two information flows IF1 and IF2)
Under the condition of a second information flow IF2 as depicted in fig 3 a much more precise failure explanation result is possible: In case of a stable

 Published in: Bäker, Unger (Hrsg.), "Diagnose in mechatronischen Fahrzeugsystemen IV", Dresden, Germany, May 2011, ISBN: 978-3-8169-3068-6

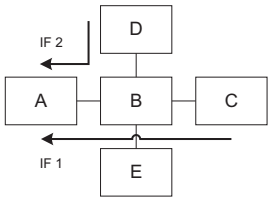

Figure 3: Isolation by Observation – two information flows IF1 and IF2

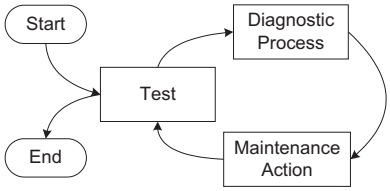

Figure 4: Cooperation of Diagnostics and Maintenance (Principle)

data packet delivery from D to A, components A and B are no longer able to explain the observed failure pattern due to their successful involvement in IF2. Thus, only C remains as possible cause of the fault. On the other hand, if IF2 is disrupted as well, component C *alone* is no longer able to explain the current failure pattern but now only in combination with component D: "No valid data packets received from C **and** no valid data packets received from D" observed by A. However, much more likely as this double fault is a single faulty component A or B. □

2.4 Reporting

Failure explanation result reporting can be organized in different modes. As one option in a basic mode a one-way only communication may be established between an equipment and its higher level entity for transmitting message frames more or less regularly. As second option a two-way communication can be used as extended mode for obtaining more detailed or additional information and performed only under safe conditions.

2.5 Maintenance and Repair

The success of a certain monitoring task and especially of an isolating task implementation can be measured from the relevance and accuracy of messages that are sent to the maintenance process. Sufficient results can only be achieved through an efficient implementation of these tasks.

Finally and regarding to [1], maintenance stands for "all actions taken to retain material in or to restore it to a specified condition. It includes inspection, testing, servicing, and classification as to serviceability, repair, rebuilding and reclamation as well as all supply and repair actions taken to keep a force in condition to carry out its mission."

The diagnostic concept described in this paper is able to find the origin cause of an observed failure pattern. In most cases there is only one iteration step necessary to fulfill this task successfully.

Nevertheless, it cannot be stressed enough that this is not necessarily true in any case. Diagnostics can only be successful in cooperation with adequate maintenance actions and is not finished until the system state is completely shifted back to "healthy" independently of the number of required iteration steps. Figure 4 shows this principle of cooperation between diagnostics and maintenance.

3 Fault Isolation by Boolean Rules

As described in the chapter above monitoring and confirmation are the basics of a diagnostic system. Both tasks can be seen as parts of a monitoring process performed usually by internal or external equipment or system users like humans.

Figure 1 shows the monitoring process at the beginning of the process chain. The diagnostic process – consisting of an isolation and a reporting task – is its successor and surely the heart of the diagnostic system. The isolation task itself can be subdivided into data acquisition, state detection, diagnostic engine and result post-processing parts.

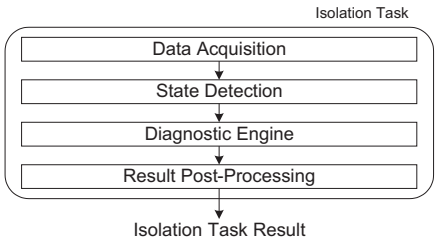

Figure 5: Isolation Task

3.1 Indicators

A failure pattern like "No valid data packets received from C" as used in the example above to explain an actual situation *to a human reader* is usually not very suitable to explain the same circumstance to a software function. Thus, a more standardized and regular pattern should be used in

order to perform the inter-process communication faster and more reliable.

A simple solution to fulfill this requirement is the use of indicators. An indicator is nothing more than a placeholder for a certain observation or fault, expressed by an integer. Therefore, "No valid data packets received from C" can be described simply by sending an indicator like "10001" to the receiving entity in case of that both parts – sender as well as receiver – know the meaning of "10001".

In a more advanced version of this concept indicators may possess properties, too. Such indicators can then be used as operational state indicators to "transport" environment data values like temperature degrees or voltage values.

3.2 Data Acquisition

The aim of a data acquisition function is to collect data from all available sources to provide all sub sequenced blocks with diagnostic relevant information. Figure 6 shows the data acquisition block, acquiring fault and operational states by using, e.g. application programming interfaces of system modules, where it receives messages from queuing and sampling ports.

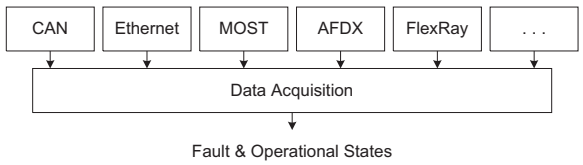

Figure 6: Data Acquisition

The Data Acquisition receives two main types of information: Fault state information and operational state information. Fault states are fault detections made by a monitoring process. Operational states may indicate a faulty state as well but in general they do not have to. In opposite to fault states operational states have to be processed using the state detection function first to indicate a fault state. Finally, a table is generated including all current states – represented by their indicators – and additional values in case of operational states.

3.3 State Detection

The state detection part of the isolation task is responsible for manipulating this table entries referring to observations which cannot be made by an ordinary monitor. This means, e.g. the distinction between a valid and a non valid observation of a certain monitor depending on a respective system configuration can be made by the State Detection as in example 3. Furthermore, under certain circumstances there may be kinds of observations which can only be made by the State Detection function. In other words, this function acts as a system monitor.

Example 3 (State Detection as higher level monitor)
Given two temperature values of *different* system parts, e.g. 50°C and 80°C. Both values are valid data within their respective allowed temperature ranges. Thus, regarding to their own measurements, anything is perfectly all right. Nevertheless, this is not necessarily true for the *whole* system when measuring an identical item. A difference of nearly 60 % between both values, as in this example, is then certainly a symptom of a real problem. □

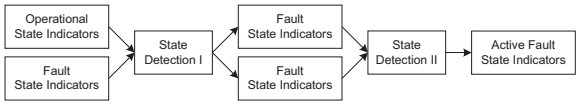

Figure 7: State Detection

The state detection function prepares all indicators which are needed by a subsequent diagnostic engine. The indicators are based on fault and operational states and extracted from messages received by the Data Acquisition as depicted in fig. 6. The State Detection receives a table including all "interesting" indicators from the Data Acquisition block as input and calculates a list of *active* fault state indicators. The meaning of "Active" in this case is that all fault indicators – either sent by devices or determined by the State Detection itself – *are set* when they are necessary to find the origin(s) of the observed failure pattern. To increase flexibility the calculation process can be divided into sub-parts as shown in fig. 7. The Diagnostic Engine is triggered only in case of a non-empty active fault state indicators list.

3.4 Diagnostic Engine

The following section gives a short introduction of some basic ideas behind a possible Diagnostic Engine which is explained in more details in [4]. This function allows for and benefits from combining failure notifications of several observers, expressed by *Fault State Indicators*, as well as combining failure notifications about several observed objects to one consolidated diagnostic result.

Despite of concentrating on only one approach in the examples below other solutions for diagnostic engines are of course possible too, e.g. [5, 6, 7, 8].

Dividing the isolation task into almost completely independent sub-parts allows a large flexibility in choosing the best fitting engine for a certain problem or system.

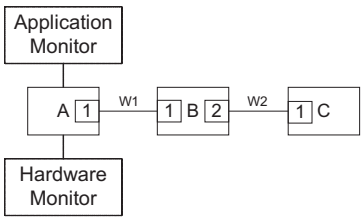

Figure 8: Simple Test System

It will be helpful to consider the simplified system depicted in fig. 8 as an example first. There, a device A hosts a diagnostics application and receives data from a device C via a third device B. In addition, A hosts one application based monitor as well as one hardware based monitor. The diagnostics application itself gathers information about observed failures from both monitors. Each device with its respective drivers is seen as functional unit which cannot be split up furthermore.

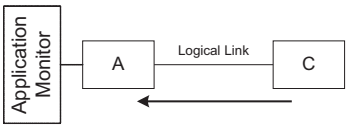

Figure 9: Application based Monitor View

In spite of having some components in common the system view of an application based monitor differs totally from the system view of a hardware based monitor. An application monitor is only able to observe its communication counterpart from a logical point of view, i.e. whether it receives data from the other side or not. It possesses no knowledge about components that implement the link physically between the monitor itself and the origin of received information.

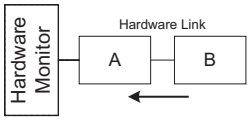

Figure 10: Hardware based Monitor View

In opposite to the logical system view of an application monitor a hardware monitor is characterized by a more physical system view. This means, a hardware monitor's view is always a point-to-point view between the hardware monitor itself and the next data packet termination point. A termination point is usually implemented by a device which is able to generate or terminate data packets, e.g. a regenerator, router or gateway. Thus, a hardware monitor's view is restricted to the next active physical component (fig. 10), whereas an application monitor's view is more related to a system wide view (fig. 9).

Example 4 (Application and hardware based monitoring)
Figure 11 uses a general data packet design to clarify the different views of application and hardware monitors. In order to detect a failure a hardware monitor has to analyze at least the checksum of the packet. By using the sum a valid packet can be distinguished from a damaged packet. The checksum itself is generated by the related data packet assembling device, e.g. the intermediate device B in fig. 8.

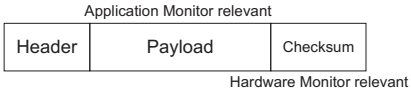

Figure 11: Relevant Parts of a Data Packet for Hardware and Application Monitor

Each time a data packet is split up into its components due to a necessary conversion between different physical communication links, the checksum has to be calculated again for the new data packet. From a hardware monitor's point of view it does not matter whether the payload of the packet makes sense or not, only the fact of a successful packet validation is important. Therefore, a hardware monitor of a certain device is only able to observe the physical link to its next data packet termination point where the current checksum has been generated before.

In case of receiving a valid data packet usually only the payload is passed through to the respective destination application. Thus, a receiving application does not concern about packet headers or packet checksums. However, there may be a mechanism within the payload in order to validate the payload itself. In opposite to a packet checksum a "payload checksum" must not be generated again or changed by a component of the communication link between the origin application and the respective destination application. Therefore, an application monitor of a certain application is only able to observe the logical communication link to its respective data source. □

The suggested diagnostic engine takes advantage of these different system views especially in a simple system as shown in fig. 8 above by using both observations for calculating the isolation task result.

Table 1 shows the "Affect Chains" of the application monitor and the hardware monitor. An affect chain in general consists of all components – hardware and software as necessary and useful – that belong to a physical or logical link between the monitor itself and the component observed by the monitor. Referring to fig. 8 the affect chains consist of device drivers and wirings between the application monitor and device C on the one hand and between the hardware monitor and device B on the other hand.

Observer	Affect Chain
AppMon	$A1, W1, B1, B2, W2, C1$
HardMon	$A1, W1, B1$

Table 1: Affect Chain Example

Example 5 (Fault Scenario $B1$)
Assume driver 1 of device B of our simple test system (fig. 8) causes a problem. A resulting failure pattern is then "No data received from device C" by the application monitor **and** "No data received from device B" by the hardware monitor. A combining of these observations leads to an isolation task result of "$A1$ or $W1$ or $B1$" as possible origins of the failure pattern. $B2, W2$ and $C1$ can be excluded from the result, because no one of them is able to explain *both* observations. □

Example 6 (Fault Scenario $C1$)
Assume driver 1 of device C causes now a problem. The resulting failure pattern is then "No data received from device C" by the application monitor **and** *no complains* by the hardware monitor. A combining of these observations leads to an isolation task result of "$B2$ or $W2$ or $C1$" as possible origins of the failure pattern. $A1, W1$ and $B1$ can now be excluded from the result, because none of them can be faulty and explain that the hardware monitor is still happy at the same time. □

3.5 Result Post-Processing

For maintenance purposes a result like "$A1$ or $W1$ or $B1$" has to be read as: "Check or replace component $A1$. If the failure disappears maintenance action is completed. If the failure does not disappear proceed by applying the same procedure to $W1$. If the failure disappears maintenance action is completed. If the failure does not disappear proceed by applying the same procedure to $B1$". In a best case scenario a failure is found and corrected by checking or replacing the first accused component of the diagnostic result. To increase the success level of this step-by-step procedure, components of a diagnostic result shall be presented to maintenance as a sorted list. A simple failure cause is in most cases more likely than a complex failure cause. Thus, an appropriate list sorting criterion is "n-component-failures before (n+1)-component-failures".

In a worst case scenario a failure does not disappear until checking or replacing each component or component combination of the result list. The reason for this behavior is caused by the Boolean characteristics of the list. This means, a presented result like "$A1$ or $W1$" does always include the result "$A1$ **and** $W1$", too. An "or" connection between two components or component combinations of a result list must not be interpreted as an "exclusive or" connection. On the other hand, a result that consists of a component combination like "$A1$ **and** $W1$" is only presented in case of that only a combination of both components leads to an exhaustive explanation of the observed failure pattern. In such a case always both components have to be checked or replaced before a failure disappearing test should be run by maintenance. The main advantage of the proposed result presentation is a higher efficiency level of maintenance actions. By presenting a weightily sorted list it is highly unlikely that each single component of the entire list has to be checked or replaced before the observed failure disappears. Moreover, the possibility of finding the cause of a failure after only one maintenance step might be very high.

4 Conclusions

A basic condition for detecting and isolating faults and failures is an adequate working monitoring. Monitoring itself means a process of continuously, periodically or on-demand scanning of a defined set of equipment functions. Usually, regarding to possible complex electronic and mechanical systems, monitoring is realized as a software function hosted on internal or external equipment.

A monitor function is responsible for detecting unwanted or unexpected states in its monitored area. It transmits observations to its process level successor. In order to increase the reliability of these detections and to avoid false alarms a confirmation of an observed state has to be made by the monitor before. If an observation cannot be confirmed, there should be no further consequences on the system as well as no message transmissions in most cases.

A single fault can cause multiple monitored observations. Ideally, an isolation process taking all observations into account leads to an identification of exactly this single root fault. A fault isolation algorithm should take advantage of system design knowledge in order to produce good results.

The success of a certain monitoring and isolation implementation depends on relevance and accuracy of messages sent to the maintenance process. Sufficient results can be achieved through an efficient monitoring and confirmation process in combination with a highly powerful isolation process.

Monitoring and confirmation are performed at the lowest level in touch with hard- and software of the monitored items. Thus, it is reasonable to put both tasks together to a monitoring process. Isolation and reporting can be processed inside of the system ("on-board") or even outside ("off-board"). As before, there are good reasons for joining both tasks to a common diagnostic process, as depicted in fig. 1. At the highest level remains the maintenance process.

Because of being process level successor, quality and accuracy of the maintenance process is highly dependent on the quality and accuracy of the diagnostic process. On the other hand, because of being process level predecessor of the diagnostic process quality and accuracy of the monitoring process has a deep impact to the quality and accuracy of the remaining process chain.

References

[1] NATO. *NATO R&M terminology applicable to ARMPs*, 1 edition, July 2001.

[2] Ministry of Defence, UK Standardization. *Defence Standard 00-42 Part 4: Reliability and Maintainability (R&M) Assurance Activity: Testability*, 2 edition, February 2008.

[3] Department of Defense - United States of America. *Definitions of Terms for Reliability and Maintainability*, June 1981.

[4] Peter Tondl, Patrick Rammelt, and Ulrich Siebel. An Efficient Diagnostic Algorithm based on Boolean Rules. Scheduled to be published in 2012.

[5] Klaus Lange and Ulrich Siebel. Ein neuer Diagnoseansatz für moderne Fahrzeugsysteme. *ZfAW – Zeitschrift für die gesamte Wertschöpfungskette Automobilwirtschaft*, 3:72–76, 2009.

[6] Patrick Rammelt, Peter Tondl, and Ulrich Siebel. Diagnostic under Uncertainty. Scheduled to be published in 2012.

[7] RODON – A Model Based Reasoning (MBR) Tool. (http://www.uptimeworld.com/Rodon.aspx, last visited at 19 March 2010).

[8] OCC'M. (http://www.occm.de, last visited at 19 March 2010).