Development of an Automated Technique for Failure Modes and Effect Analysis

Blanke, M.; Borch, Ole; Allasia, G.; Bagnoli, F.

Publication date:
1999

Document Version
Også kaldet Forlagets PDF

Development of an automated technique for failure modes and effect analysis

M. Blanke & O. Borch,
Department of Control Engineering, Aalborg University, Denmark

G. Allasia & F. Bagnoli
D'Appolonia S.p.A., Italy

ABSTRACT: Advances in automation have provided integration of monitoring and control functions to enhance the operator's overview and ability to take remedy actions when faults occur. Automation in plant supervision is technically possible with integrated automation systems as platforms, but new design methods are needed to cope efficiently with the complexity and to ensure that the functionality of a supervisor is correct and consistent. In particular these methods are expected to significantly improve fault tolerance of the designed systems. The purpose of this work is to develop a software module implementing an automated technique for Failure Modes and Effects Analysis (FMEA). This technique is based on the matrix formulation of FMEA for the investigation of failure propagation through a system. As main result, this technique will provide the design engineer with decision tables for fault handling that show how fault migration can be stopped.

1 INRODUCTION

Faults in one subsystem of an automated plant have often undesired effects on other subsystems if remedial actions are not taken promptly after a fault occurs. Dependability of a control system can be obtained by giving it ability to detect and isolate faults and react with actions that accommodate the control system to the fault. Fault accommodation is predetermined at the design stage: a control system can freeze to a safe state or the controller can be reconfigured. This can be done e.g. by replacing the measured signal from a faulty sensor by an estimate obtained from remaining available signals, together with known analytic relations of the particular part of the plant.

Handling of faults in open loop systems is technically straightforward, but the reactions used to accommodate a fault need to be designed with careful consideration to safety and availability of the total plant. Optimization at a local level may easily violate an overall safety goal. Handling of faults in closed loop components is a more difficult and challenging task. Properly designed systems can accommodate the effects of faults whereas less careful designs can let fault effects propagate to other subsystems.

Analysis of failure of parts of a system is a classical discipline, and the failure mode and effects analysis (FMEA) is widely used and appreciated in industry. The FMEA method was developed as a mainly manual approach which was then suggested computerized (Lege 1978, Herrin 1981, Yuan 1985, Bell 1989). The traditional FMEA approach does not support the handling of faults in automated systems where the ultimate goal is autonomous handling of a fault to obtain continued operation of a plant. A formal link to fault detection and isolation (FDI) models was suggested in (Blanke 1996), (Blanke et al. 1997). These papers dealt with the algebraic description of fault propagation analysis. A further development with case studies was done by Bøgh (1998).

A conclusion of this and other case studies was that dedicated tools would be needed to cope with complexity and assure correctness of analysis and later implementation.

The present paper presents a development effort to automate the analysis using an object oriented approach to the design of a prototype tool for automated analysis of fault propagation.

2 THE FMEA MATRIX

For the reasons given above, fault analysis needs to incorporate analysis throughout a system. In order to do this a component-based method was introduced (Blanke 1996), in which possible component faults are identified at an early stage of design. The method uses an FMEA description of components as a starting point. The lowest level of analysis, in this context, this is sensors, valves, motors and similar components. Programmable parts are considered as

consisting of separate function blocks that can be treated similarly to physical components in the analysis, bearing in mind that their properties may be changed by software modifications if so desired. All potential faults and their effects are determined.

An FMEA scheme for each component shows how fault effects out of the component relate to faults at inputs, outputs, or parts within the components. This is illustrated in (1).

Using f_{ci} for component faults and e_{ci} for the effects, the FMEA scheme can be expressed as:

$$e_{ci} \leftarrow A_i^f \otimes f_{ci} \tag{1}$$

where A^f_i is a Boolean matrix representing the propagation. The index 'i' is a component identifier and ⊗ the inner product disjunction operator. The operation carried out by the operator is equivalent to the scalar Boolean disjunction "∨" and the inner product to the "∧", i.e., row no. k of (1) is:

$$e_{cik} \leftarrow (a_{ik1} \wedge f_{ci1}) \vee (a_{ik2} \wedge f_{ci2}) \vee \ldots \vee (a_{ikn} \wedge f_{cin}) \tag{2}$$

When some faults are effects propagated from other components, we get:

$$e_{ci} \leftarrow A_i^f \otimes \begin{bmatrix} f_{ci} \\ e_{c(i-1)} \end{bmatrix} \tag{3}$$

System descriptions are obtained from interconnection of component descriptions. The description of a system with three components and open loop structure is:

$$e_{c3} \leftarrow A_3^f \otimes \begin{bmatrix} f_{c3} \\ e_{c2} \end{bmatrix}; e_{c2} \leftarrow A_2^f \otimes \begin{bmatrix} f_{c2} \\ e_{c1} \end{bmatrix};$$

$$e_{c1} \leftarrow A_1^f \otimes [f_{c1}] \tag{4}$$

The combined fault-effect description for this example is constructed in three steps:

$$e_{c3} \leftarrow A_3^f \otimes \begin{bmatrix} f_{c3} \\ e_{c2} \end{bmatrix};$$

$$e_{c3} \leftarrow A_3^f \otimes \begin{bmatrix} I & 0 \\ 0 & A_2^f \end{bmatrix} \otimes \begin{bmatrix} f_{c3} \\ f_{c2} \\ e_{c1} \end{bmatrix};$$

$$e_{c3} \leftarrow A_3^f \otimes \begin{bmatrix} I & & 0 \\ & & \\ 0 & A_2^f \otimes \begin{bmatrix} I & 0 \\ 0 & A_1^f \end{bmatrix} \end{bmatrix} \otimes \begin{bmatrix} f_{c3} \\ f_{c2} \\ f_{c1} \end{bmatrix} \tag{5}$$

Effects are seen to be propagated to the next level of analysis and act as parts' faults at that level. This is continued until the system level is reached. The schemes give a surjective mapping from faults to effects: there is a unique path from fault to end effect, but several different faults may cause the same end effect.

Reversal is obtainable through finding the generalized transpose A^b of A^f. The matrices A^f and A^b are each other's pseudo inverse in the Boolean sense. When there is no feedback involved, the result is the capability of isolation of fault effects at any level.

The FMEA scheme for a set of components connected in a closed loop is principally described as:

$$e_{ci} \leftarrow A_i^f \otimes \begin{bmatrix} f_{ci} \\ e_{ci} \end{bmatrix} \tag{6}$$

Looking at the logic operation of this equation, it is obvious that the solution, if it exists, is:

$$e_{ci} \leftarrow A_i^f \otimes [f_{ci}] \tag{7}$$

The implication is that an automated analysis will need to consider closed loops as special cases. The interpretation of a closed loop in a fault-propagation analysis is merely the observation that closed loop operation may amplify or attenuate the effect of a fault. Which of the two happens depends on the dynamic properties of the control loop. This question can not be answered by the simple Boolean matrix analysis.

In the design tool, it was chosen to automatically locate closed logical loops, let the user cut them open and automatically create additional elements of the faults and effects vectors. Mathematically, this means to replace (6) by:

$$\begin{bmatrix} e_{ci} \\ e_{2ci} \end{bmatrix} \leftarrow A_i^f \otimes \begin{bmatrix} f_{ci} \\ f_{2ci} \end{bmatrix} \tag{8}$$

where e_{2ci} and f_{2ci} are the fault vector and the effects vector elements, respectively, of the signals at the place where a loop has been cut open. It is noted that each of the two vectors can have multiple elements.

The user will manually need to investigate these additional faults and effects, treating them as extra input faults and output effects of the subsystem considered.

3 A PROTOTYPE TOOL FOR AUTOMATED ANALYSIS

Using this method, a module for performing automated FMEA has been carried out. This module is integrated within a tool, the Prototype Design Tool (PDT), which supplies a graphical user interface (GUI) for the definition of the system using a dedicated component library. This is a database of components including graphical image, attributes, behavior description when failure occurs and mathematical models. An overview of the PDT can be found in Figure 1.

In the users' eyes the design tool is a GUI with views and interactive features. Sub-components of this GUI are distributed among the parts of several modules of the PDT. The users are able to interactively change working context.

By this GUI the user can build a new system topology (or load a previously stored one since serialization is supplied) and then perform the FMEA analysis on it.

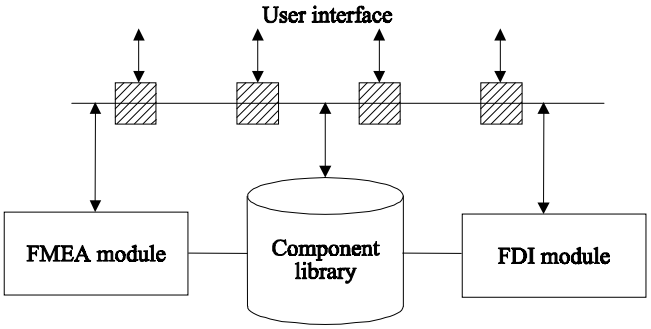


Figure 1. Overview of the PDT.

The FMEA module can be launched within the PDT once that the system topology is completed. The FMEA module supports an independent GUI that allows the user to perform the following interactive actions:

1 Identification of closed loops.

2 Cut of closed loops (the user can decide where to cut).

3 Restoration of the cut connections.

4 Calculation of FMEA matrix of the system and performing reverse analysis.

When a closed loop is identified in the topology the user is able to open it. So, when all the closed loops are cut, the analysis of the open system can be carried out.

If the component is a brand new one, the user will be requested to fill the FMEA matrix of each component of the topology, and he can also decide to change the failures that can affect a component and, hence, be eventually propagated forward.

At the end of these operations, the automated analysis can be carried out.

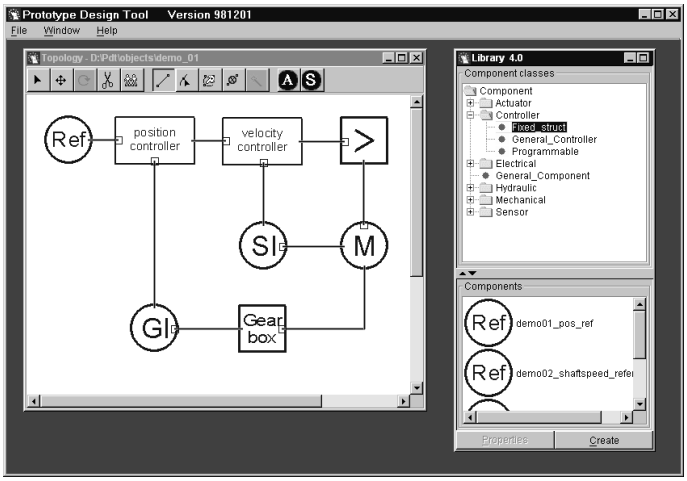


Figure 2. Main GUI of the PDT.

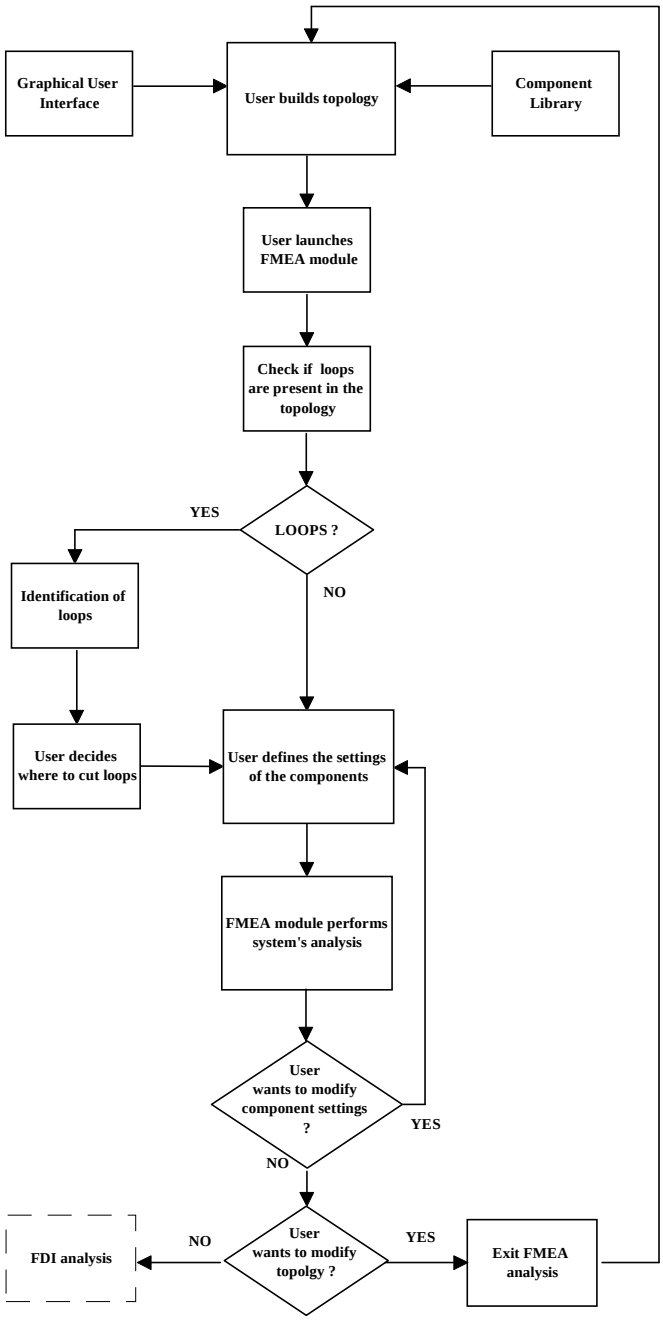


Figure 3. Flowchart of the main operations performed by the FMEA Module.

The result of the analysis is the whole FMEA matrix of the system, which establishes a relationship between faults and end effects of the system (including the additional faults and effects, which are generated by the cut of the closed loops).

Reverse analysis can be performed and it is possible to highlight the propagation path of each fault. The result is that, for each end effect of the system, it is possible to have a list of all the faults, which generate this end effect, and a consequent list of all the components involved in the propagation of each of these faults. In this way, the user is able to edit the FMEA matrix of the components eligible to implement the blocking of failure propagation (i.e. programmable ones), and modify the matrix accordingly. In the actual programmable function block, this would imply that fault detection and isolation is applied to the particular signal and that remedial action is made in software to accommodate the fault.

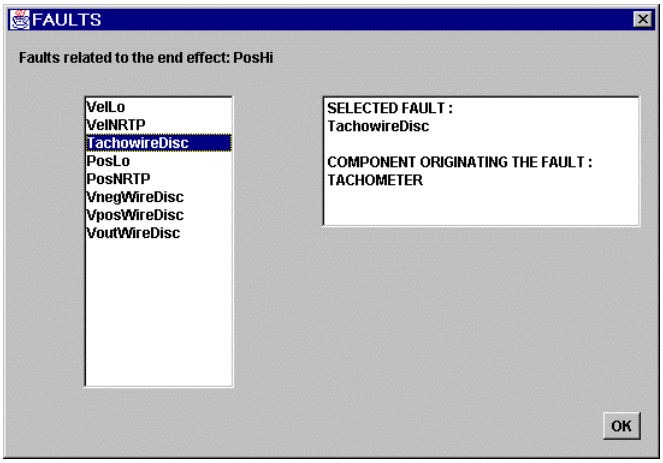


Figure 4. GUI representing the list of fault related to a selected end effect.

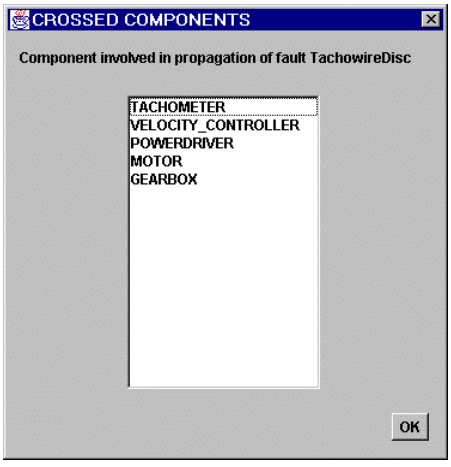


Figure 5. GUI representing the list of component crossed by the propagation of a selected fault.

This technique is a basic design methodology for construction of fault-tolerant systems in both plant monitoring and control.

In Figure 2 the main GUI of the PDT tool in which the user can build or load a topology and launch the FMEA Module is represented. A flowchart of the main operations performed by the FMEA Module is shown in Figure 3, while Figure 4 and Figure 5 show the GUIs for the representation of the results of the reverse analysis.

4 IMPLEMENTATION

4.1 *Object Oriented Model*

The FMEA module and the entire PDT has been designed following the Object Oriented Programming (OOP) paradigm, and the Java™ has been chosen as the implementation language. The Java source code has been compiled using version 1.1.5 of the Java Developer Kit (JDK) and Swing 1.0.3 has been adopted for aiming the development of the GUI.

The core of the FMEA module is a multithreading application in which all the fault propagations can run contemporary in a multithreaded environment.

The relationships among the main classes of the FMEA Module are shown in Figure 6.

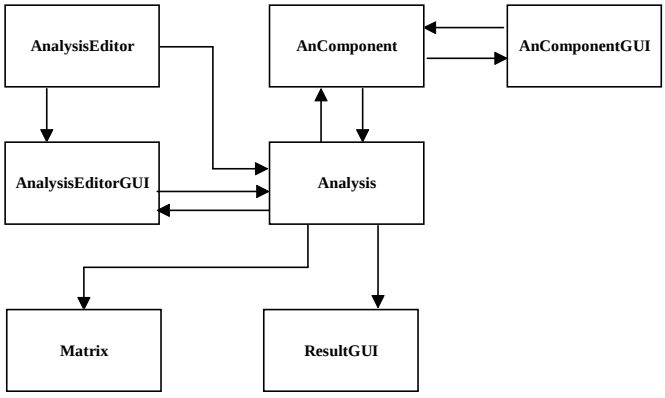


Figure 6. Relationships among the main classes of the FMEA Module.

4.2 *Implementation description*

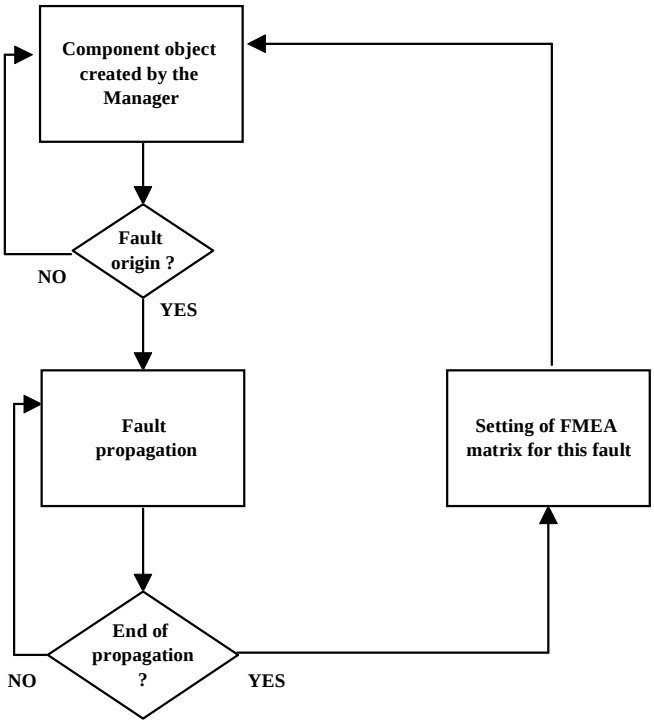
The basic idea is that the components of the topology which have intrinsic failures or input failures external to the system are considered as starting points for the analysis. In the same way the components that have output connections with elements not belonging to the topology under analysis are considered as target points. In this case the failures propagated through these connections constitute the ef-

fects of failures at topology level, and hence, the target of the failure propagation analysis.

An object manager has the task of collecting the results and building the whole FMEA matrix of the system.

Each component object knows its origins and destinations, i.e. the components with which it is connected in input and output, and, independently from the others, verifies if it has any failure (intrinsic or input ones), i.e. if a start origin is found. If this check has a positive outcome the propagation for that fault starts, and the failure is spread through the system, according to its corresponding row of the FMEA matrix of the component. The component which receives this fault executes the propagation operations, sending it to other components with its original identifier. This action is performed until a target component is reached and the result is collected by the manager. A flowchart, which explains this operation of propagation, is given in Figure 7.

Figure 7. Flowchart of the propagation algorithm.



The Manager object knows all the information about the topology. This information can be represented as:

1. Identifiers of all the components of the topology: the PDT module has a method by which the Manager can get all these univocal.
2. Connections between components: for every component the PDT module gives all the information about the components connected in input and output.
3. Inner failure matrices: for every component the user is able to edit the inner failure matrix by a dedicated GUI.
4. Faults of the components: for each component the user can add a fault, so a row is added to the failure matrix and the component become a starting point for this new fault (in the same way a fault can be removed from a component).
5. Opening of closed loops: for each closed loop the user is able to decide where to open it, so the Manager adds new inputs (practically new faults, but "fictitious" ones) to the component placed where the loop has been cut, and also new fictitious end effects are considered for the topology.

Another basic interface between FMEA module and the PDT module is the one regarding the serialization phase. In fact the Topology object of the PDT module has a reference to the Matrix object of the FMEA module. In this object the FMEA module stores the information added by the user and the one obtained as a result of the analysis phase (i.e. the global FMEA matrix), this information regards:

1. Inner failure matrices of the single components.
2. Names of input and output variables of the components
3. FMEA matrix of the system.

When the PDT serializes the current topology, all this information is serialized together with the topology.

5 THE LINK TO FAULT DETECTOR DESIGN

The above analysis provides a list of al fault effects which are required to be detected. This has been obtained using a consistent method throughout the analysis, so completeness is guaranteed to the extent that component failure models are complete. This does not provide completeness in a mathematical sense. It is, however, as good as other reliability assessments using well established failure mode descriptions of components.

The list of effects to be detected is used in the design of fault detection. Simple means to detect single sensor faults were treated by Blanke et al. (1997), When redundant information is needed to detect more complicated faults, analytical FDI can be employed. A recent reference among many good books is Gertler (1998).

Automatic generation of the entire modeling for FDI is a research topic, and at present, the FDI filters need to be designed using an engineering effort. The list of faults to detect, and the required actions to accommodate a fault, as provided by the PDT is an important step forward.

6 CONCLUSIONS

The paper has introduced the concept of automated analysis of fault propagation using FMEA descriptions of components as the basis for the analysis. A technique to cope with the problem of logic analysis of fault propagation for closed loops was suggested and tested in the prototype tool. Experience from case studies using the prototype tool has shown the feasibility of the approach.

The salient feature of this approach is that ability to make a reverse analysis. This enables the designer to pinpoint the faults with high severity and determine where the propagation of a fault can be stopped. This in turn shows exactly which faults should be detected and by which programmable components this should be done. It also shows to the designer which remedial actions should be taken (automatically) to accommodate a particular fault.
A second generation prototype tool with further integration between topology definition and enhanced facilities for user interaction will be a natural next step. Simultaneously, application on larger test cases will bring the tool closer to industrial use.

7 ACKNOWLEDGEMENTS

This work was partially funded by the ATOMOS project under EU DG7 contract no. WA-95-SC.205.

REFERENCES

Bell, T.E. 1989. *Managing Murphy's Law: Engineering a Minimum-Risk System*. IEEE Spectrum, June 1989.

Blanke, M. 1996. *Consistent Design of Dependable Control Systems.* Control Engineering Practice, 4(9): 1305-1312.

Blanke, M., R. Izadi-Zamanabadi, S.A. Bøgh & C.P. Lunau 1997. *Fault Tolerant Control - A Holistic View*. Control Engineering Practice, 5(5): 693-70.

Bøgh, S.A. 1998. *Fault-tolerant Control Systems - Development Method and Real Life Case Study*. Ph.D. dissertation, Department of Process Control, Aalborg University, Denmark.

Borch, O., M. Blanke, J. Buck, F. Bagnoli & T.F. Lootsma 1998. *ATOMOS II Prototype Design Tool*. EU ATOMOS project, task 2.6.3 report.

Bagnoli F. & G. Allasia 1998. *FMEA software documentation and demonstration of prototype FMEA software.* EU ATOMOS project, task 2.6.2 report.

Gertler, J. 1998. *Fault detection and Diagnosis in Engineering Systems*. Marcel Decker.

Herrin S.A. 1981. *Maintainability Applications Using the Matrix FMEA Technique.* IEEE Trans. on Reliability, 30(3).

Lege J.M. 1978. *Computerized Approach for Matrix-Form FMEA*. IEEE Trans. on Reliability. 27(1): 154-157.

Veillette, R.J., J.V. Medanic, & W.R. Perkins 1992. *Design of Reliable Control Systems*. IEEE Trans. Automatic Control, 37(3): 290-304.

Yuan, J. 1985. *Strategy to Establish a Reliability Model with Dependent Components through FMEA.* Reliability Engineering. 11(1): 37-45.