FMEAssist: A KNOWLEDGE-BASED APPROACH TO FAILURE MODES AND EFFECTS ANALYSIS

James R. Carnes
Dannie E. Cutts

Boeing Huntsville AI Center
PO Box 1470, MS JA-65
Huntsville, AL 35807

ABSTRACT

A Failure Modes and Effects Analysis workstation (FMEAssist) has been designed for use during development of the Space Station. It assists engineers in the complex task of tracking failures and their effects on the system. Engineers experience increased productivity through reduced clerical loads, reduced data inconsistency, and significantly reduced analysis time. System developments benefit from a more thorough analysis than was available using previous methods.

The wide variety of design information required to support the FMEA process is modeled by FMEAssist in a network of different discipline and design data views generated by a data base to knowledge base translation tool. System designs are displayed graphically allowing engineers to manipulate information or to induce and record failure on appropriate parts within the network. Propagation of functional effects for each node can be controlled from one or more nodes within the design to any desired level or until special conditions are encountered.

1. INTRODUCTION

Space Station design information is modeled by FMEAssist in a network of nodes (representing components) connected by arcs (representing relationships between the various parts of the design). The wide variety of information required to support the FMEA process is acquired from a view across several heterogeneous discipline and design data base tables and are mapped into a network structure through Foundation, a data base to knowledge base translation tool[4]. System designs are displayed graphically allowing engineers to analyze design information and to induce failure on appropriate parts within the network.

The architecture of this system is built upon a hierarchically decomposed functional model that determines "failure" through abnormal component behavior. This representation permits a more detailed description of "failure modes" beyond those typically pre-defined through either design or system engineering.

These functional failures are composed of component constraint violations. Propagation of functional effects is controlled from one or more nodes within the design to any desired level or until special conditions, such as a critical failure, are encountered.

2. APPLICATION

Previous work on FMEA automation used a frame-based approach where frames and slots contain pre-defined failure modes and first order effects. While this approach offered powerful descriptive capabilities, first-order analyses were performed manually and entered into the model. Since assertional capabilities were not provided, functional information was not modeled. Failures were propagated through pre-programmed frame connections using messages contained in failure mode slots.[3]

Alternately, a number of fault analysis systems have been developed using assertional models, that is, a rule-based functional approach. While assertional models provide an excellent medium for describing functional behavior, they do not provide the representational convenience of the structural model. Ongoing work has pursued a useful mixture of structural and function models.[1,6]

The FMEAssist approach integrates some of this work on the coupling of structural and assertional components by combining the connective strength of flavors with the expressiveness of logic. A failure mode is defined as the effect a group of abnormal properties has on a component. If the status of the component, due to the its properties, is "failed" or "abnormal", then these properties (and not the fact that the component has malfunctioned) are propagated to connected or neighboring components.[2,5]

Abnormal properties are grouped into failure modes at the component level, but only to promote analysis within the failed component. The knowledge of the mode or even the failure itself is not distributed to the connected or surrounding nodes, but it is the connection and environmental properties that carry a component's fate to neighboring components. That is, a malfunctioning component has no knowledge of how it affects other parts. If the malfunction changes any of its outputs, those values change for the connected component input ports. Input properties for components are the output results from previous inference upon abnormal properties. If these new properties constitute failure or can be classed in a failure mode, they are so recorded, but inference on component properties, normal or otherwise, continues.

"Effects analysis" becomes a deductive process, reasoning from the properties contained in a highly structured part model[5]. The need for pre-programmed propagation responses is replaced with a more refined description of an assembly or

component. This type of descriptive and behavioral information is readily available and can be captured during the design and system engineering process.

3. EXAMPLES

The examples in this section are designed to promote understanding of the descriptive structures and behaviors found within FMEAssist. Figure-1 illustrates how components are defined and described. The defcomponent macro in this example serves to create a component type PUMP-A which inherits characteristics (descriptive and behavioral) from another type called PUMP. The defport and defproperty macros are used to define port/port types and property/property values for a component type.

Some behavioral characteristics of a component type remain constant regardless of how and where instances of it are used in the system. These "generic" behavioral characteristics are shared by all instances of the component type. Examples of generic behavioral descriptions can be found in Figure-2 and Figure-3. Other characteristics however, might change depending on the operating conditions of the instance component. These behavioral characteristics are determined at the particular instance level.

While working with the FMEAssist application on a Foundation Workstation, engineers are able to graphically display networks from various perspective relations, such as subcomponent, failure, and connection spanning tree. Inference mechanisms provide the basis for the application's single, multiple-sequential, and multiple-parallel failure propagation. FMEAssist also provides graphical justification or narrative explanation for any inference produced during the failure propagation. Finally, various reports are generated on the analytical results from each engineering session.

```
; define some generic component descriptions
(defcomponent PUMP-A (generic-type 'PUMP))
(defcomponent VALVE-C (generic-type 'VALVE))

; define input and output for generic component descriptions
(defport PUMP-A ((port-1 'thermal) (port-2 'electrical)))
(defport VALVE-C ((port-1 'thermal) (port-2 'electrical)))

; define properties for port connections
(defproperty THERMAL
  (temperature (nominal high low))
  (pressure (nominal high low))
  (medium (air co2 water)))
```

FIGURE 1: Examples of descriptive definitions

```
; define some rules about components
(defrule PUMP-SHUTDOWN
  (IF [AND [PUMP-PRESSURE-STATUS =some-pump 'LOW]
           [PUMP-TEMPERATURE-STATUS =some-pump 'LOW]]
   THEN (tell [PUMP-STATUS =some-pump 'ABNORMAL])))

(defrule PROPAGATE-PRESSURE-STATUS
  (IF [PUMP-PRESSURE-STATUS =some-pump 'LOW]
   THEN (tell [PUMP-PRESSURE-OUTPUT =some-pump
                                    =some-port 'LOW])
        (tell [(connected-to =some-pump =some-port) 'LOW])))

(defrule PROPAGATE-TEMPERATURE-STATUS
  (IF [PUMP-TEMPERATURE-STATUS =some-pump 'LOW]
   THEN (tell [PUMP-TEMPERATURE-OUTPUT =some-pump
                                       =some-port 'LOW])
        (tell [(connected-to =some-pump =some-port) 'LOW])))

; create some part instances
(make-component PUMP-99 (component-type 'PUMP-A))
(make-component VALVE-46 (component-type 'VALVE-C))
(make-connection '(PUMP-99 VALVE-46)
                 '((port-1 port-1) (port-2 port-2)))
```

FIGURE 2: Examples of specific assertive definitions

```
;Output behavior rule
(defrule LEAKY-THINGS-CONTAMINATE-SURROUNDINGS
  (IF [AND [LEAKS =something]
           [> (PRESSURE =something)
              (PRESSURE (contained-in =something
                                      =something-else))]
   THEN (tell [CONTAMINATES (medium =something)
                            =something-else])))

;Input behavior rule
(defrule CONTAMINATED-ELECTRONIC-COMPONENTS-MIGHT-SHORT
  (IF [AND [CONTAMINATES =medium =area]
           [CONTAINED-IN =component =area]
           [CONDUCTOR =medium]
           [ELECTRICAL-COMPONENT =component]]
   THEN (tell [HAS-SHORTS =component])))
```

FIGURE 3: Examples of generic assertive definitions

4. CONCLUSIONS AND FUTURE DIRECTIONS

Engineers will experience increased productivity through reduced clerical loads, reduced data inconsistency, and significantly reduced analysis time with the added benefit of a more thorough analysis than was available using previous methods. FMEAssist is easy to use, produces FMEA and Critical Items List (CIL) reports, and keeps records of critical failures, as well as sequences of events leading to failures.

In the future, tools like FMEAssist will make it possible for initial failure analyses to be performed early during the system design phases. These tools will be able to identify significant failure modes for single and multiple-point failures. This will free engineers from the tedious task of enumerating the simple single failure mode groupings and to provide an extended capability of correlating complex failure modes with groups of components.

REFERENCES

1. Brachman, R.J., R.E. Fikes, and H.J. Levesque, KRYPTON: A Functional Approach to Knowledge Representation, IEEE Computer, Vol. 16(10), 1983, pp. 67-73.

2. de Kleer, J. and B.C. Williams, Diagnosing Multiple Faults, Artificial Intelligence, Vol. 32, 1987, pp. 97-130.

3. Kamhieh, C.H., D.E. Cutts, and R.B. Purves, Failure Modes and Effects Analysis Automation, Proceedings of the Conference on Artificial Intelligence for Space Applications, November, 1986, pp. 169-176.

4. Purves, R.B., J.R. Carnes, and D.E. Cutts, Foundation: Transforming Data Bases into Knowledge Bases, Proceedings of the Conference on Artificial Intelligence for Space Applications, November, 1987.

5. Reiter, R., A Theory of Diagnosis from First Principles, Artificial Intelligence, Vol. 32, 1987, pp. 57-95.

6. Rowley, S., H. Shrobe, R. Cassels, and W. Hamscher, Joshua(TM): Uniform Access to Heterogeneous Knowledge Structures, AAAI-87, Proceedings of the Sixth National Conference on Artificial Intelligence, July 13-17, 1987, pp. 48-52.