Automated FMEA based diagnostic symptom generation.

Neal Snooke[1,*], Chris Price

Department of Computer Science
Aberystwyth University
Penglais, Aberystwyth
Ceredigion, SY23 3DB, U.K.

Abstract

The comprehensive on-board diagnosis of faults in many aerospace and other engineered systems requires real time execution using limited computational resources, and must also provide verifiable behaviour. This paper shows how a diagnostic system satisfying these requirements can be automatically generated from the model based simulation used to produce an automated Failure Modes and Effect Analysis (FMEA). The resulting diagnostic system comprises a set of efficiently evaluated symptoms and their associated faults. The symptoms are complete in that they include all necessary observations required to determine applicable system operating states, unlike other work that finesses this problem by having models for each operating state and producing diagnostics for each operating state separately. The symptoms are also efficient because they abstract complex system behaviour based on observations available to the diagnostic system and only preserve sufficient symptom detail to isolate faults given these available observations.

This work has been done in the context of diagnosing autonomous aircraft, and is illustrated with examples from that domain. The models used as a basis for automated generation of diagnostics were originally produced to automate the production of a FMEA report, and the paper also considers the relationship between FMEA and diagnostics that provides verification of the failure effects predicted by the simulation and hence validation of the generated symptoms.

Keywords: Qualitative reasoning, Automated FMEA, Symptom generation, Diagnosability analysis

1. Introduction

1.1. Context

This paper describes research carried out on the ASTRAEA project, a pioneering £32 million aerospace programme which is addressing key technological and regulatory issues in order to open up non-segregated airspace to unmanned autonomous aircraft [9]. In order to operate autonomous aircraft safely in normal airspace, a high degree of confidence is needed in the capability of the aircraft to accurately carry out prognosis and health management (PHM). A key part of the PHM task is the ability to identify problems on board the aircraft and to diagnose those problems correctly.

The on-board diagnosis of complex vehicle systems is a challenging task, and qualitative model based diagnosis has been successfully applied to produce on-board systems that assist in the detection and isolation of faults [26, 4, 7]. The majority of these systems use executable models of the systems (either with or without fault models) simulated in parallel with the working system. The model is supplied with the actual inputs to the system and abductive reasoning is performed based on discrepancies between the predicted and observed system behaviour [8]. The reasoning approaches utilised to perform fault candidate generation from the discrepancy between predictions and observations are categorised in [6].

A second approach that has been adopted is to use the same type of model-based analysis as above, with a defined set of possible component faults, to generate a set of candidate component faults for ev-

*Corresponding Author
Email addresses: nns@aber.ac.uk (Neal Snooke), cjp@aber.ac.uk (Chris Price)
[1] *Tel:* +44 (0)1970 621782

ery possible system failure. This information forms the basis of an FMEA, for example earlier circuit based work by the authors in[21]. Indeed, any system that is capable of the necessary nominal and failure behavioural predictions can potentially be used to provide the data for symptom generation, for example the qualitative deviation model approach of [24]. That approach does not however provide any particular advantage with respect to efficiency of the FMEA production since the deviations do not replace the absolute model for the FMEA task[23].

There is some similarity to the 'association based' category of reasoning described in [6] whereby discrepancies are matched with stored symptom-failure associations. The significant differences in the present work are that possible discrepancies are precomputed (optimised) and the symptom-failure associations are generated automatically from the system description removing any maintenance issues. There are a number of advantages to the Automated FMEA based approach:

- The diagnostic system's capabilities can be characterized before deployment. Specifically, the scope and accuracy of the diagnostics can be checked and verified. This is important in our context, as stringent certification and regulation approval requirements exist.

- The technical challenges associated with execution of complex models and model accuracy checking is done off line, in advance, and once only and therefore the on board detection of faults is efficient allowing guaranteed response times.

- Additional symptoms that do not come from a model of the system can be used to augment the model-based diagnostics.

- The diagnostic weight of each symptom can be calculated to reflect reliability.

Component fault models and the identification of the operating states in which each component fault can be detected are required in order to produce good results. It is not sufficient merely to test pressures or voltages in a system, as such symptoms are often state-dependent — problems can only be observed when the system is in specific operating modes. Practical applications of this technology often link hand generated symptoms to component faults in order to produce efficient on-board diagnostic symptoms.

The research described in this paper improves the efficiency and consistency of this second approach in three ways. Firstly, it integrates the model-based diagnosis with model-based failure modes and effects analysis (FMEA): this means that all included component failures have been considered and agreed by engineers using the existing design process. Secondly, it provides algorithms for producing state-based symptoms that comprehensively characterise the observable failures in the system. Finally, it determines whether all system failures can be observed with the existing sensors, and indicates which component failures can be isolated with existing sensors.

Section 2 describes the overall structure of the process of producing useful symptoms, and section 3 explains the structure of the output of a model-based FMEA report.

Section 4 provides an efficient algorithm for the selection of symptomatic observations from a large set of possible candidates.

Section 5 addresses the problem of generalising observations from a representative set of system states to all possible states.

Section 6 outlines how the generated symptom-fault sets can be used to create a diagnostic system, followed in the final section by the results of applying the symptom generation technique to realistically sized systems.

2. Background

Figure 1 illustrates the main steps in the process of producing detectable symptoms for the on-board diagnostic system. The shaded items represent previously existing data and analysis.

Model-based generation of FMEA. This has been well-documented in previous papers [16] and is in daily use in industry to produce FMEA reports on automotive electrical systems. It simulates the qualitative system behaviour for all potential component failures, and uses functional reasoning [14] to abstract the high level consequences for each possible failure. The consequences of every failure are reported to the designers, and they can decide the steps needed to improve the safety and reliability of the system.

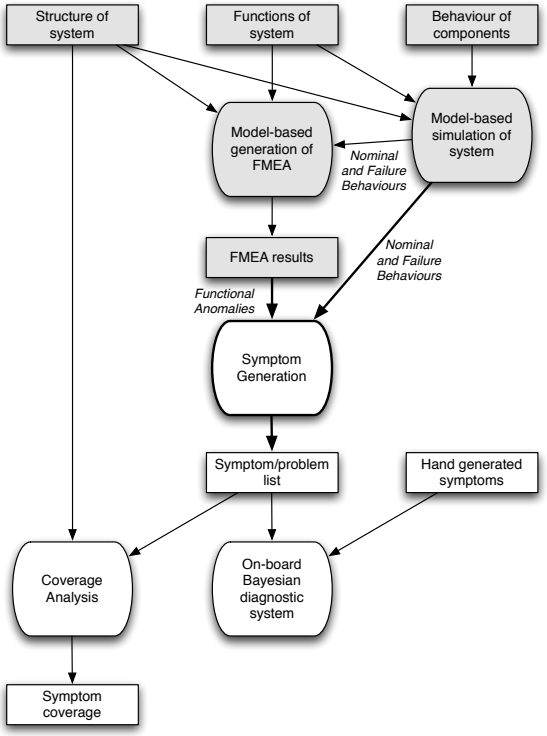

Figure 1: Model-based On-board Symptom Generation

Symptom generation. This is the main contribution of this paper. The FMEA results provide a link between the system failures and the component faults that caused them. For service bay diagnostics, these links can be used to find the cause of a problem with the system. However, in on-board diagnosis, the known information is limited to sensor values and system state, and it is necessary to decide how and when to use observable behaviour both to decide that there is a problem, and to deduce the root cause of that problem. Sections 4 and 5 of this paper describe how to use automated FMEA results to generate a set of characteristic symptoms to be monitored that will provide the maximum information about the state of the system.

On-board diagnostic system. One of the limitations of model-based reasoning is that it can only deal with problems that are within the scope of the modelling. For example, if problems are caused by cross-component interactions that are not modelled, then they would not be included in the set of symptoms to monitor [11]. The generated set of symptoms to be monitored can be augmented by further symptoms that compensate for the limitations of the chosen modelling. In our solution, all of the symptoms are included in an existing proprietary Bayesian network based diagnostic system developed by BAe Systems [19]. The process of using qualitative symptoms within this system is outside the scope of this paper, however several key features are provided in section 6 of this paper.

Coverage analysis. Ideally, the on-board diagnostic system would be able to detect and isolate every possible component fault that could occur. In practice, on-board systems have a limited set of sensors, and those sensors are often decided before an on-board diagnostic system is planned. It is however easy to modify the visibility of any model parameter or simulation variable in the system model. We have developed a tool that exploits this to allow investigation and analysis of the relationship between measurement availability and diagnosability, presenting the information to the engi-

neer as a visual cluster matrix together with summary information. These techniques and tools can be used during early design to quickly assess the impact of different sensor selection strategies and are the subject of a separate paper [22].

There are several specific advantages of this process for the applications and case studies targeted by this work :

- The simulation effort need be carried out once only as part of existing design analysis requirements, and does not need to be done on the on-board hardware. This means that the diagnostic system requires relatively simple on board processing.
- The ability to supplement the symptom set used for on-board diagnosis is valuable for several reasons. Symptoms can be included based on specialist sensors outside of the modelling domain of the system to be included, such as a temperature sensor measuring electrical component temperatures. In addition, outputs of components or subsystems that include their own black box diagnostic capabilities that cannot be modelled can be included in the diagnostic system. Fault prognosis tends to require specialist techniques and sensors to identify system degradation; if these exist then symptoms can be included that convert these aspects into a diagnostic output. In these cases the fault is not a hard fault but a degradation or future failure prediction, and the measurements and observations may include outputs from the specialist sensors or software. Finally, further symptoms can be included (or possibly excluded) where for pragmatic reasons modelling simplifications exist in the models used for the design analysis.
- Existing experience can be included if a Bayesian version of the on-board diagnostic system is used by allowing adjustment of the weighting of symptoms to reflect any uncertainty regarding the applicability of a symptom, for example where measurements might be considered less reliable.
- The diagnostic system's capabilities can be characterized in advance, and the sensing requirements and cost weighed against the diagnostic and fault isolation capability of the system. An engineer can experiment with sensor selection and assess the capability of the resulting diagnostic system [22].

3. Characterising FMEA Descriptions

The basis of the state-based symptom generation work is the application of the simulation results obtained during the generation of an FMEA. These simulations drive the system to explore a broad range of nominal and failure operating states.

This section provides a formal description of the process of generating a model-based FMEA report abstracted by function. This will then be used as a basis for the novel method of symptom generation.

A trivial electrical system comprising several lamps, switches and connecting wires are depicted in figure 2 will be used as a running example throughout the paper to illustrate the notation, algorithms and results. Only wire fracture faults will be included, and the system will be instrumented with the current flow in each wire as a basis for symptom generation. For reasons of space, results extracts will be used in the paper; the complete auto-generated results for the example are available as supplementary material with the online version of the paper.

3.1. FMEA generation

An FMEA report is produced by comparison of system function and behaviour for nominal and failure mode operation. The system is exercised through a number of states using a **scenario** defined by an engineer that (in general) activates each of the system functions and major functional operating modes. The following paragraphs develop a notation for the FMEA results to be used for symptom generation.

OBS is defined as a finite set of first order sentences representing all measurements (outputs or inputs, or observable component states) available from a target system. Some members of *OBS* may be complex expressions, for example to create 'virtual sensors', however the term **measurement** is used to refer to a single sentence because the majority contain a simple proposition that compares the value of a measurable quantity. Typical sentences in the qualitative representation of the system used in this work might be $\mathsf{temperature} = \mathsf{high}$ or $\mathsf{switch.position} = \mathsf{on}$ although it could also be a component state, software variable or any other observable.

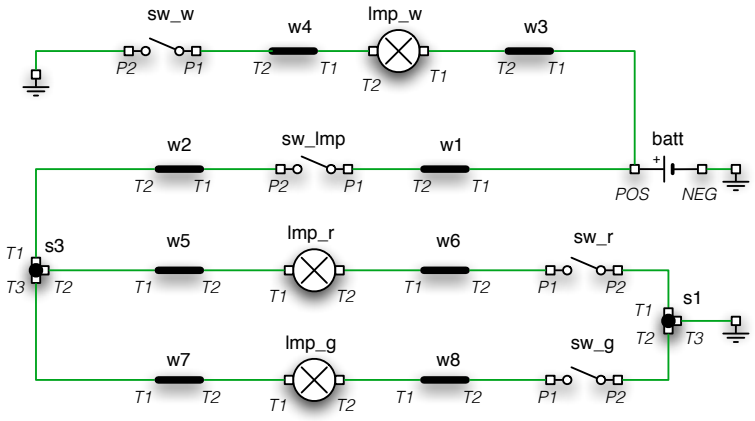

Figure 2: Simple electrical system

A scenario, SCN, will provide an ordered sequence of n input states for the system where $SCN_n \subset OBS$. That is, SCN_n defines a set of triggers. A simulation is performed for each SCN_n, and for systems that reach a steady state[2], an observation is produced (containing all available measurements). The notation OBS_n is used to indicate the subset of OBS produced (by simulation) for SCN_n with no faults present.

A target system is comprised of a set of constants $COMPS$ representing the components of the system. Each component has a set of mutually exclusive failure modes Modes(c) where $c \in COMPS$. Single faults are generally used for FMEA generation, not only because of the effort involved in considering the results of multiple faults, but also because most design issues are highlighted from the single fault cases [18]. The complete set of component failure modes $\mathcal{M}$ for a system is thus:

$$\mathcal{M} = \bigcup_{\forall c \in COMPS} \text{Modes}(c) \tag{1}$$

The simulation can support any number of simultaneous failures and on occasion an engineer may decide to include specific categories of multiple failure. Throughout this paper, M is used to represent a (simultaneous set of) component failures considered as a failure mode in an FMEA. $M = \emptyset$ is considered as the no failure case, and usually M contains a single component failure mode although generally $M \subseteq \mathcal{M}$ ensuring only one failure mode from each component:

$$\begin{array}{l}(\forall m, \forall n \in M, n \neq m, \forall c \in COMPS) \\ (m \notin \text{Modes}(c) \vee n \notin \text{Modes}(c))\end{array} \tag{2}$$

The notation OBS^M is used to refer to the set of observations obtained by simulation of the system for SCN with faults M present and OBS^M_n thus provides concise notation for a **failure mode observation**, a single set of measurements for SCN_n.

An example scenario with 8 steps for the running example is shown in the top section of figure 3[3]. The engineer has exercised the white, red and green lamps (lmp_w, lmp_r, lmp_g) individually by changing switch settings. All other information presented is derived automatically from the qualitative simulation combined with a functional model considered in the next section. In the lower part of the figure the first component fault ($M = \{\mathsf{w1 - fracture}\}$) is shown for two scenario steps SCN_5 and SCN_8 because these were the only observations containing abnormal measurements for this fault. For the purposes of FMEA the state of the **Failure Mode** (abnormal function state) is the primary concern and the detailed **behaviour** is usually hidden.

The functional model provides interpretation of the simulation observations and also plays a role in the symptom generation process and is therefore described separately in the following section. The electrical activity in the circuit is specified as active or inactive as a direct result of the qualitative

[2]Systems that do not achieve a steady state will produce a sequence of observations, however we will not consider those here.

[3]The complete FMEA for the example is available as supplementary information with the online version of this paper.

electrical simulation, and the risk factors (severity, detectability and occurrence) are derived from the functional and component models. For clarity, only abnormal measurements are shown on the human readable FMEA report however the nominal measurements are captured thus providing an observation, for example $OBS_5^{\{\mathsf{w1-fracture}\}}$. Further details of the FMEA generation are contained in [15, 14].

3.2. Function model

A functional interpretation model is used to allow the results of a component based qualitative simulation to be abstracted into effects meaningful to an engineer for the purpose of FMEA generation [14, 3]. The functional model also has a critical role in the ability to generate diagnostics from the simulation results produced during the FMEA because it can determine the relevance of measurements by operating state, thus allowing symptom generation using only the representative system states provided by the FMEA.

An engineer describes the system functions, $\mathcal{F}$ using a Functional Interpretation Language (FIL) as detailed in [3], by defining the *triggers* required to activate the function and the *effects* required for the function to be considered as achieved. The triggers and effects are linked to fragments of system behaviour for any system that implements the function. For the purpose of this paper it is only necessary to understand that each system function, $fn \in \mathcal{F}$, can be in one of four possible states provided by the predicates $\mathrm{Ac}(fn), \mathrm{In}(fn), \mathrm{Fa}(fn), \mathrm{Un}(fn)$, that represent the function having been Achieved, Inoperative, Failed or Unexpected. These function states are defined in terms of trigger (Tr) and effect (Ef) propositions, grounded in a subset of the possible system measurements within the FIL as follows:

$$\mathrm{Ac}(fn) \equiv \mathrm{Tr}(fn) \wedge \mathrm{Ef}(fn) \tag{3}$$

$$\mathrm{In}(fn) \equiv \neg\mathrm{Tr}(fn) \wedge \neg\mathrm{Ef}(fn) \tag{4}$$

$$\mathrm{Fa}(fn) \equiv \mathrm{Tr}(fn) \wedge \neg\mathrm{Ef}(fn) \tag{5}$$

$$\mathrm{Un}(fn) \equiv \neg\mathrm{Tr}(fn) \wedge \mathrm{Ef}(fn) \tag{6}$$

Some intuition as to the type of knowledge captured by the functional model can be seen in figure 4 that defines the produce_red_light function for the running example. The produce_red_light function provides a purpose named red to its external environment and requires a trigger activate_red and effect red_illuminated. There are similar definitions for **white** and **green** functions for the system. Any undefined identifiers are references to values provided by an associated system simulation. For example the proposition sw_lmp.Position == 'on' is associated with a switch input position and provides part of the trigger for the function, and the observation lmp_r.R1 == 'active' evaluates current flow in the resistance representing a lamp and is associated with the required effect (for pragmatic reasons, the FIL uses == as the equivalence operator and = for assignment).

Purpose information is not used in symptom generation but captures the teleological intent of the function within a deployment environment [5]. Generally the functional model will be a relatively simple identification of each system function, how it is activated and what effects are to be expected. The formal details of the FIL can be found in [3, 1].

```
FUNCTION produce_red_light {
    ACHIEVES red
    BY activate_red TRIGGERS red_illuminated}

PURPOSE red {
    DESCRIPTION 'provide red light to the user'
    FAILURE_CONSEQUENCE 'user does not see any red light'
        SEVERITY 5 DETECTABILITY 2}

TRIGGER sw_lmp.Position == 'on' AND sw_r.Position =='on'
    IMPLEMENTS activate_red

EFFECT lmp_r.R1 == 'active'
    IMPLEMENTS red_illuminated
    UNEXPECTED_CONSEQUENCE 'unwanted red light'
    SEVERITY 3 DETECTABILITY 1
```

Figure 4: FIL illustration

3.3. FMEA Abstraction

The engineer-level FMEA report is generated by:

- simulating the system with a variety of failure modes $M_1, M_2, ..$
- evaluating the state of system functions for each element in OBS^{M_i}
- reporting failed and unexpected functions using the definitions in equations 5 and 6.

The FMEA report for simulation step SCN_n of fault M will contain information relating to the following set of abnormal functional results:

$$RfnAb_n^M = \{fn \mid \begin{array}{l}\mathrm{Fa}(fn) \text{ evaluated for } OBS_n^M \\ \vee\, \mathrm{Un}(fn) \text{ evaluated for } OBS_n^M\}\end{array} \tag{7}$$

Scenario

This section lists the steps in the scenario. These are the major changes to the components that can recieve inputs.

Step_1 : Start - PRIOR TO FIRST EXTERNAL CHANGE

Setting:ALL COMPONENTS's INITIAL VALUES to DEFAULT

Step_2 : sw_w.Position - on

Setting:sw_w's Position to on

Function	Effects	Effect Details	Sev,Det
'produce_white_light' achieved		Function produce_white_light Achieved	0, 0

Step_3 : sw_w.Position - off

Setting:sw_w's Position to off

Step_4 : sw_lmp.Position - on

Setting:sw_lmp's Position to on

Step_5 : sw_r.Position - on

Setting:sw_r's Position to on

Function	Effects	Effect Details	Sev,Det
'produce_red_light' achieved		Function produce_red_light Achieved	0, 0

Step_6 : sw_r.Position - off

Setting:sw_r's Position to off

Step_7 : sw_g.Position - on

Setting:sw_g's Position to on

Function	Effects	Effect Details	Sev,Det
'produce_green_light' achieved		Function produce_green_light Achieved	0, 0

Step_8 : sw_g.Position - off

Setting:sw_g's Position to off

Details

The following table lists the results from the automatically generated FMEA

Item	Behaviour	Failure mode	Failure Causes	Sev	Det	Occ	RPN
F1	**At sw_r.Position - on (Step_5)** Observable: 'lmp_r.illuminated' — Value: off (on expected) 'w1.flow' — inactive (active expected) 'w2.flow' — inactive (active expected) 'w5.flow' — inactive (active expected) 'w6.flow' — inactive (active expected) **At sw_g.Position - on (Step_7)** Observable: 'lmp_g.illuminated' — Value: off (on expected) 'w1.flow' — inactive (active expected) 'w2.flow' — inactive (active expected) 'w7.flow' — inactive (active expected) 'w8.flow' — inactive (active expected)	Function: 'produce_red_light' failed — Effects: user does not see any red light — S,D: 5, 2 Function: 'produce_green_light' failed — Effects: user does not see any green light — S,D: 4, 2	w1 - fracture.	5	2		10

Figure 3: Automated FMEA fragment for circuit in figure 2

In addition, the related sets of failed and achieved functions are of interest for symptom generation:

$$RfnFa_n^M = \{fn \,|\, \mathrm{Fa}(fn) \text{ evaluated for } OBS_n^M\} \quad (8)$$

$$RfnAc_n^M = \{fn \,|\, \mathrm{Ac}(fn) \text{ evaluated for } OBS_n^M\} \quad (9)$$

The functional information provides adequate information to generate a traditional FMEA, however specific issues can also benefit from additional explanation with detailed behavioural abnormalities identified by comparison of the nominal and failure observations:

$$RobsAb_n^M = \{o \in OBS_n^M \mid o \notin OBS_n\} \quad (10)$$

The example, in figure 3 reports the **failure mode**:

$$RfnFa_5^{\mathsf{w1-fracture}} = \{\mathsf{produce\_red\_light}\}$$

and the **behaviour** comparison:

$$RobsAb_5^{\mathsf{w1-fracture}} = \{ \begin{array}{l} \mathsf{w1.flow : inactive}, \\ \mathsf{w2.flow : inactive}, \\ \mathsf{w5.flow : inactive}, \\ \mathsf{w6.flow : inactive}\} \end{array}$$

Once the FMEA results are considered acceptable by an engineer, symptom generation can be carried out. A prerequisite for FMEA production is that there should be no abnormal functions in the absence of a failure:

$$(\forall n)(RfnAb_n^{\emptyset} \Leftrightarrow \emptyset) \quad (11)$$

If proposition 11 is not satisfied for some *OBS* then the system has failed to implement its function description and this design verification issue should be addressed before any meaningful FMEA or symptom generation can be produced. In figure 3, $RfnAb^{\emptyset}$ (no fault) results are shown in the **scenario** section. Inoperative functions are not shown on the report for clarity, and there are no abnormal functions (Fa, Un) allowing symptom generation to proceed for the example.

4. Symptom generation

At first glance the FMEA appears to directly provide symptoms simply by associating abnormal behaviour $RobsAb_n$ and failure modes M. While this may sometimes provide a set of measurements, $\mathcal{E}$, that form a usable symptom, the majority of $RobsAb_n$ measurements remain within the range of values encountered during nominal operation for at least one other non failure state. Therefore most symptoms actually require one or more **nominal** measurements in addition to a **subset** of $RobsAb_n$ to provide a symptom that does not spuriously indicate faults.

If engineers generate vehicle on-board diagnostics by hand, they typically identify a sensor or set of sensors to observe, and measurements that indicate an operating state of the system in which the observations can be made. For example, in figure 2 if there is no current flow at lmp_w *and* sw_w=='on', then one possible fault is w3-fracture.

A fault may cause many abnormal observations and since it is beneficial for $|\mathcal{E}|$ to be as small possible for a given level of diagnosability and fault isolation, the abnormal observations in $|\mathcal{E}|$ may need to be a subset of those available in a specific abnormal observation. When manually creating symptoms from a hand generated FMEA, an engineer would make these selections and create conditional symptoms (discussed later) based on extensive system knowledge. The work described here automates this process and produces a consistent and comprehensive set of symptoms.

The automated symptom generation technique described in this paper does not require explicit external knowledge concerning the relevance of measurements to either faults or functions to compensate for the missing state information inherent in the FMEA. Rather, this information is extrapolated from analysis of behavioural consistency covering all of the faults and all of the system states encountered during the *entire* FMEA. Given a reasonably comprehensive FMEA that exercises all system functions and includes faults for the majority of components, the aim is to produce symptoms that are general enough to cover a great deal of the system state that is not explicitly included in the FMEA without producing spurious symptoms. Failing to do this will mean that the diagnostic system will only have the ability to diagnose faults in a very limited set of operating modes of the device. A typical example might be that diagnosis is only possible when a single system function is operating. The functional model (also used to interpret the FMEA) is the enabling concept that indirectly provides abstract guiding knowledge facilitating extrapolation of symptoms to system states that were not present in the FMEA.

The symptom generation therefore comprises two main parts as follows:

- Logical expressions on sets of observations must be created that implicate faults. This is essentially a search task that involves selecting sets of observations that implicate the smallest set of faults that do not occur in nominal operation. Sections 4.1-4.3 develop this task.

- Detailed behavioural information is only available for a representative set of system states and therefore some generalisations must be made. Section 5 develops a method that utilises functional information to produce generalised symptoms.

4.1. Identification of symptomatic measurements

This section provides an efficient algorithm for the selection of symptomatic observations from the candidates obtained from the FMEA. The algorithm generates a complete symptom set for a system where *every possible* state was encountered in an FMEA.

A symptom is defined as a tuple $S = (E, F)$ such that E is a first order sentence referred to as a **symptom expression** that when satisfied indicates $F = \{M_1, M_2, ...\}$ as **symptom faults**. A set of satisfied symptoms forms the diagnosis candidates $\mathcal{D} \subset \mathcal{M}$. $E(s)$ evaluates E on a specific system state $s \in OBS$. The term **simpler symptom** is used to refer to a symptom that has fewer measurements required to evaluate E. A more **complex symptom** requires a greater number of measurements than a simpler one. In this paper E is always a simple conjunction of equivalence propositions that represent measurements, and we use $\mathcal{E}$ to represent these individual measurements and therefore the simplicity measure for any symptom is $|\mathcal{E}|$. For example if $E = (\mathsf{sw\_w} \leftrightarrow$ 'on' $\wedge\ \mathsf{lmp\_w.flow} \leftrightarrow \mathsf{active})$ then $\mathcal{E} = \{\mathsf{sw\_w} \leftrightarrow$ 'on', $\mathsf{lmp\_w.flow} \leftrightarrow \mathsf{active}\}$ with $|\mathcal{E}| = 2$. If $|F|$ is smaller for symptom S_1 than S_2 then we refer to S_1 as more **specific** than S_2 and S_2 as more **general** then S_1.

The aim of the symptom generation is to produce a symptom set $\mathcal{S} = \{S_1, S_2, ...\}$ that will provide a diagnosis as a set of possible faults $\mathcal{D}$, such that if M is a given failure mode, then $M \in \mathcal{D}$ while avoiding spurious diagnoses $M \notin \mathcal{D} \wedge \mathcal{D} \neq \emptyset$. In addition the general principle used is to generate the simplest symptoms that provide a given level of fault detection/isolation to avoid redundant measurements forming part of a symptom. No diagnosis of a fault $\mathcal{D} = \emptyset$ is preferable to a spurious diagnosis but given a fault M, the more operating time or states that $\mathcal{D} \neq \emptyset$ the more powerful the fault detection, and the smaller $|\mathcal{D}|$ the better the diagnostic system fault isolation. These constraints define the basic goals of the symptom generation technique.

Every component fault M considered by an FMEA results in several sets of measurements OBS_n^M as a response to the steps in the scenario.

One or more measurements from OBS_n^M may be conjoined to form a symptom, however they must satisfy two constraints. Firstly, a symptom should *only* be present when the associated fault is present. Secondly, it is desirable to detect as many faults as possible in as many operating states as possible. This implies symptoms should be general and therefore contain the fewest measurements, such that the first constraint is satisfied.

Given a failure mode observation, $s \in OBS^M$, the aim is to find sets of measurements, $\mathcal{E} \subseteq s$, where $|\mathcal{E}|$ is minimised, such that:

$$\mathcal{E} = \{R \subseteq s \mid \nexists n : R \subset OBS_n\} \tag{12}$$

For each fault observation OBS_x^M related to a failure mode M, a set $\overline{N}$ is defined that itself contains sets of measurements that are abnormal with respect to each of the nominal observations. Each failure mode is considered individually so the fault mode superscript is omitted from the notation of $\overline{N}$ in the following description.

$$\overline{N_x} = \{\quad OBS_x^M \setminus OBS_1,\ OBS_x^M \setminus OBS_2,\ .. ,\ OBS_x^M \setminus OBS_{|SCN|}\} \tag{13}$$

where $|SCN|$ is the number of distinct steps in the scenario.

Any symptom must contain *at least one* measurement from every element of $\overline{N}$ to satisfy the requirement that the symptom does not contain only measurements that exist during nominal operation. Definition 12 can thus be written:

$$\mathcal{E}_x = \{R \subseteq OBS_x^M \mid \forall \overline{N}_i \in \overline{N} : \exists o \in \overline{N}_i \cup R\} \tag{14}$$

To efficiently select measurements, an index is assigned to each member of OBS^M according to which members of $\overline{N}$ it appears in. $\overline{N}$ is considered as an ordered set and each member is assigned an index value from a power 2 sequence. This facilitates a tree structured search that locates simple symptoms rapidly and allows the majority of longer

symptoms to be ignored. Each measurement o is assigned its index in the following way:

$$\text{index}(o) = \sum_{i=1..|\overline{N}|} \begin{cases} 2^{i-1}, & \text{if } o \in \overline{N}_i \\ 0, & \text{otherwise} \end{cases} \quad (15)$$

Figure 5 illustrates the assignment of indices for a small example with 3 scenario states and 6 possible measurements *a-f*. The nominal observations are shown at the top of each column representing a scenario state and the possible failure observations for the failure are shown below.

$\overline{N}_i$ for a specific failure observation is generated by removing the nominal mode observations for scenario step i from the given failure observations.

The ordering of the measurements is depicted by a matrix using rows to represent measurements and columns to represent each element of $\overline{N}$. The sum of the abnormal observation set indices for each row provides the order to search the measurements.

Since $|\overline{N}| = |SCN|$ and is related to the number of system functions, the index values have been small enough to compute explicitly for the systems we have applied the method to. A sorting algorithm that performs a sequence of partial sorts of s based on membership of each element in $\overline{N}$ would be equivalent for larger scenarios, however it is doubtful that an FMEA with such an excessive number of scenario steps would be useful, and a better solution might be to divide the analysis into subsystems. The case where $\text{index}(m) = \text{index}(n)$ for two measurements is considered in section 4.2.

The symptoms are now generated by traversing the ordered list of measurements, including additional measurements until a symptom is formed that includes at least one measurement from each set of $\overline{N}$:

1. Initially set $k = |\overline{N}|$
2. Each measurement m_p contained in $\overline{N}_k$ is considered as part of a new potential symptom expression, $\mathcal{E}_p$. Consider each $\mathcal{E}_p$ in turn:
3. *Completed symptoms check.* If $\mathcal{E}_p$ contains at least one element from every set in $\overline{N}$ (i.e. $\sum_{(\exists m \in \mathcal{E}_p)(m \in \overline{N}_i)} i = 2^{|\overline{N}|-1}$ then $\mathcal{E}_p$ is a completed symptom. $\mathcal{E}_p$ is converted into a symptom expression by conjoining all the elements $E = \bigwedge_{\forall m \in \mathcal{E}_p} m$. Consider any remaining $\mathcal{E}_p$ possibilities.
4. An $\overline{N}$ index, j, is chosen for $\mathcal{E}_p$ such that $1 \leq j < k$ and $(\forall m \in \mathcal{E}_p)(m \notin \overline{N}_j)$. In addition if $j < i < k$, then $(\nexists i)(\forall m \in \mathcal{E}_p)(m \notin \overline{N}_i)$. That is, $\overline{N}_j$ must be the next element in $\overline{N}$ not already represented by the symptom being constructed.
5. *Dead end symptom check.* If no j can be found that satisfies the constraints in step 4 then $\mathcal{E}_p$ cannot form a symptom and is removed from consideration. Consider any other $\mathcal{E}_p$ possibilities.
6. If the $|\mathcal{E}_p|$ has not equaled any completed symptom, it is used as the basis of a new sets of partial symptoms $\mathcal{E}_{p'}$, by adding an additional measurement, $\mathcal{E}_{p'} = \mathcal{E}_p \cup m_q$ where $min(\text{index}(m_q))$ such that $m_q \notin \mathcal{E}_p \wedge m_q \in \overline{N}_j$. That is, the lower ranking measurement not already forming part of the partial symptom in $\overline{N}_j$. For each $\mathcal{E}_{p'}$, recursively apply from step 3 setting $\mathcal{E}_p = \mathcal{E}_{p'}$, and setting $k = j$.

Only the simplest symptoms are required and the breadth-first strategy ensures all equally simple symptoms are located first thus terminating the search. Also contributing to the efficient selection is the characteristic that measurements contributing to the greatest number of remaining $\overline{N}$ elements are always earlier in the sequence, and because these are the measurements that distinguish the failure state from the maximum number of nominal measurements, they are the best candidates to form completed symptoms using fewest measurements.

The ordering of the selection from the $|\overline{N}|$ sets making up $\overline{N}$ is arbitrary, but once an order is chosen, it determines the indices of the actual measurements, and must then be applied consistently. The reason for ordering measurements and then selecting sets of measurements in this way is to produce symptoms that are guaranteed to not be part of any set of nominal observations.

The lower part of figure 5 illustrates the process. For OBS_1^M in the leftmost column, two symptoms are generated with two measurements required for both. Two initial **partial** symptoms $(b, \{\text{M}\})$ and $(d, \{\text{M}\})$ are possible as the initial nodes in the search tree. The measurement with the highest rank of 6 is b, and provides two elements $\{\overline{\text{N}}_3, \overline{\text{N}}_2\}$ from $\overline{N}$, requiring only measurements from $\overline{\text{N}}_1$ to complete $\overline{N}$ and produce a symptom. The search from b is continued by traversing further down the measurements list for measurements involving $\overline{\text{N}}_1$. d is the next candidate and completes the set $\overline{N}$. The completed symptom is therefore $(b \wedge d, \{\text{M}\})$.

For the second of the initial symptoms the next

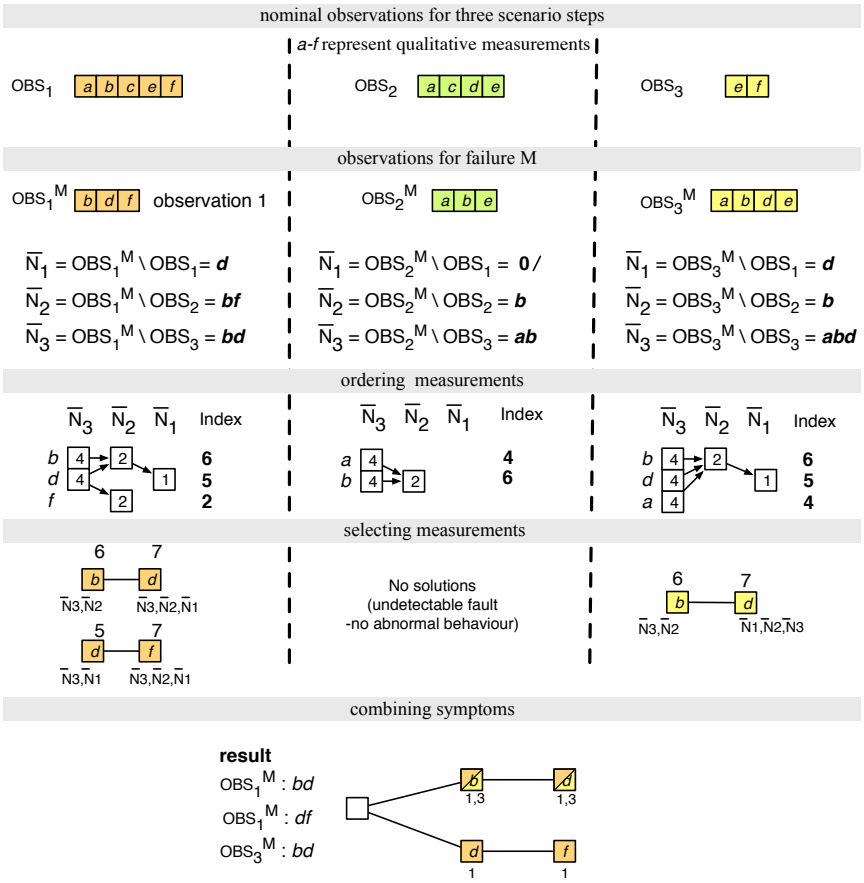

Figure 5: Symptom Search

element of $\overline{N}$ required is $\overline{N}_2$. Considering remaining measurements in $\overline{N}_2$, b, f are possibilities, and f also completes $\overline{N}$ for the symptom resulting in the additional symptom $(d \wedge f, \{\mathrm{M}\})$. The d then b is abandoned without further searching since if at all, it would provide complexity>2 and simpler symptoms have already been found for the observation.

The second failure mode measurement OBS_2^M produces $\overline{\mathrm{N}}_1 = \emptyset$ and hence no symptoms are possible because all of the measurements are seen in $\overline{\mathrm{N}}_1$ and manifests as an empty matrix column for $\overline{\mathrm{N}}_1$ in the figure. OBS_3^M produces a single symptom, because those starting d and a are abandoned at $\overline{N}_2$ on complexity measure. Finally, the symptoms generated from each observation are added to an ordered tree of measurements to allow the complete set of faults related to each symptom to be captured.

Using the running example we find that scenario step 5 provides the first symptoms for fault w1.fracture as shown in figure 6. The 8 scenario steps lead to $\overline{N}_{1-8}$ (this is notated *ND0-ND7* by the software implementation). Clearly sw_r.Position is only on in step 5 of the scenario (figure 3), and therefore belongs to all groups except $\overline{N}_5$ (i.e *ND4*). w2.flow is abnormally inactive in step 5 but would normally be active in step 7 and therefore belongs to *ND4* and *ND6*. The symptom tree firstly selects the top ranking measurement containing *ND0*. This is the sw_r measurement. Since this also includes *ND1-ND3* another measurement is sought from the list that includes *ND4*. This is the w2 measurement and completes the full set *ND0-ND7* and comprises a complete symptom: $S_1 = (\mathsf{sw\_r.Position{:}on} \wedge \mathsf{w2.flow{:}inactive}, \{\mathsf{w1.fracture}\})$. The process continues until no more symptoms are found.

It is desirable that symptoms are complete so that a valid symptom should indicate *all* detectable faults. Therefore given $S = (E, F)$ is a symptom

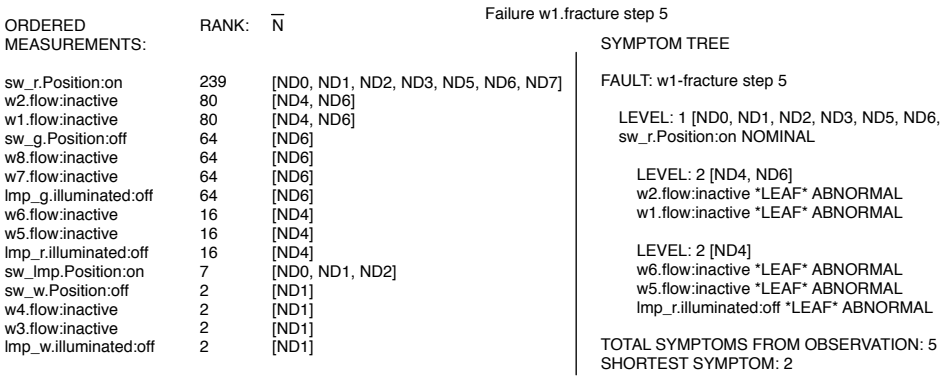

Figure 6: First w1-fracture observation producing symptoms

and $E(OBS)$ is the result of evaluating E on a set of observations:

$$(\forall OBS_n^M)((M \in F) \Leftrightarrow E(OBS_n^M)) \qquad (16)$$

This leads to an additional element in the symptom generation. Each completed symptom must be compared with the observations associated with all other failure mode observations and additional faults included in the symptom to ensure constraint 16 is satisfied. In practice this is a case of merging the symptom trees produced from each observation.

4.2. Equivalent measurements

It is common for several different measurements to have equivalent diagnostic power within a symptom produced from a single fault observation. A simple concrete example is provided by two wires connected in series; a lack of current flow in either one will indicate the same faults. The presence of equivalent observations leads to more than one measurement with identical index(o) values in equation 15. These symptoms can be handled in two ways. Either symptoms must contain disjunctive expressions that allow for any one of a number of measurements to be used, or several independent symptoms are generated, one for each of the equivalent measurements. The symptom generation algorithm therefore simultaneously includes all measurements with the same index number as m_p at step 2 and 6.

4.3. Fault exoneration

The diagnostic symptoms as generated in the previous sections should not be negated to perform fault exoneration - a symptom expression evaluating to false does not *necessarily* indicate fault absence. For example consider a lamp with a plausible symptom:

$$S = \begin{array}{l}(\mathsf{lamp} \leftrightarrow \mathsf{inactive} \wedge \mathsf{switch} \leftrightarrow \mathsf{on}, \\ \{\mathsf{lamp\ blown}, \mathsf{switch\ corroded}, \mathsf{wire\ fractured}\})\end{array}$$

If the switch is off then switch $\leftrightarrow$ on is false resulting in a false symptom expression, but this does not imply the lamp is OK - it could be blown.

Manually crafted symptoms are often **conditional** so that $\neg E$ will exonerate all of the faults in F. This is because engineers consider the implication of not seeing the symptom as well as its presence. The Bayesian net based diagnostic system for which the symptoms are being generated requires that symptoms can exonerate faults and therefore $(\forall M \in F, \forall n)(\neg E(OBS_n^M) \implies M \notin \mathcal{D})$. This also allows better fault detection by allowing evidence from nominally operating parts of a system to contradict evidence from general symptoms, hence reducing the set of possible faults.

Assigning $E_c \equiv$ switch $\leftrightarrow$ on and $E_o \equiv$ lamp $\leftrightarrow$ inactive, the earlier example can be reformed $(E_c \wedge E_o, \{\mathsf{lamp\ blown}, \mathsf{switch\ corroded}, ...\})$ and $(\neg(E_c \rightarrow E_o), \{\neg\mathsf{lamp\ blown}, \neg\mathsf{switch\ corroded}, ...\})$. The separate conditional part E_c must be satisfied for the symptom to be **valid**. Any valid symptom that is not satisfied (i.e. $E_c \wedge \neg E_o$) can be used to exonerate the associated faults. For the example if the switch is on and the lamp is active then we can predict that the lamp is not blown, the switch contact is not dirty, and the wire to the lamp is not fractured. A valid symptom that is satisfied implicates the associated faults. A symptom that is not valid provides no information.

To ensure negatable symptoms, E is partitioned to produce E_c and E_o to ensure the symptom is either satisfied or invalid for any observation OBS^M

where $M \in F$. That is, a symptom expression must not exonerate a fault for an observation where the fault exists. Therefore if $S = (E_c, E_o, F)$ measurements must be included in E_c to ensure:

$$(\forall M \in F, \forall n, \nexists OBS_n^M)(E_c \wedge \neg E_o) \qquad (17)$$

For symptoms formed as a conjunction of measurements, $\mathcal{E} = \{m_1, m_2, ...\}$ then $\neg E = \neg m_1 \vee \neg m_2 ...$ That is, if one of the measurements in the (proposed) symptom is *not* present in *any* of the observations for the fault(s) indicated by the symptom, then the symptom cannot be negated to exonerate the fault. To satisfy sentence 17 the symptom generator therefore partitions E for each symptom by finding candidate conditional measurements as follows: $\mathcal{E}_{c'} = \{o \in \mathcal{E} \mid \forall M \in F, \forall n : o \notin OBS_n^M\}$. Because $|\mathcal{E}_{c'}| \leq |E|$ and E is very small (section 4), a simple breadth first search rapidly finds all the smallest subsets of $\mathcal{E}_{c'}$ that satisfy proposition 17. Finally, $\mathcal{E}_o = \mathcal{E} \setminus \mathcal{E}_c$ is obtained.

Figure 7 contains the full set of symptoms generated for the running example. These symptoms will correctly diagnose the system provided we only enter the states provided in the FMEA. To remove this restriction, the next section will modify the symptom generation by restricting the measurements available based on function state.

Using *all* of these candidate conditional measurements is often over cautious, and may lead to the inability of some symptoms to exonerate faults, particularly when nominal output measurements form part of the symptoms. As a result an initial refinement was made that restricts the measurements available for symptom generation. s is thus redefined as a subset of OBS_n^M to include only abnormal measurements plus triggers associated with the failed functions (defined in equations 7-9).

$$s = \{(RobsAb_n^M\} \cup \{\mathrm{T}(fn)) \,|\, fn \in RfnFa_n^M\} \cap OBS_n^M \qquad (18)$$

This was the measurement selection used for the ASTRAEA project to generate symptoms for a twin engine aircraft fuel system. For that system this simple approach worked well primarily because it was relatively easy to exercise all of the function permutations that could be encountered during operation. For the running example it does not change the symptoms generated, since no nominal measurements are selected by the generation algorithm.

Subsequent experiments for complex systems where operating modes combining multiple functions were restricted in the scenario, together with experience from systems where it is infeasible to exercise all potential function permutations, identified the need to be more selective than the simple approach in equation 18 and is the subject of the next section.

5. Observation selection strategies

This section details the use of functional information to produce generalised symptoms from an FMEA analysis which does not exercise all system states, or even all permutations of possible system functionality. Generalisation enables a set of symptoms with good coverage to be produced.

The symptom generation presented in section 4 in effect generalises symptoms because not all system states are available in the FMEA. However this generalisation is somewhat arbitrary. Symptoms may include measurements that are not causally relevant to a fault, simply because there was no state that provides a counter example. Such symptoms often are not applicable to states outside of those encountered in the FMEA. To relieve this difficulty more steps might be inserted in the scenario, however this typically leads to less simple, equally limited symptoms. Thus, a simple search for symptoms will produce either very limited coverage of operating modes or possibly artifactual symptoms for realistic FMEA scenarios, due to incorrect assumptions on unobserved state. To address these issues there are two possibilities:

- Ensure the necessary states are provided by the FMEA. As noted previously, adding steps to the scenario simply shifts the problem, and ultimately a complete attainable envisionment [10] is required. In addition a great deal of insight is required to determine which additional steps are required. For these reasons this approach is infeasible for most real systems.
- Constrain the search based on states related to structural and behavioural and functional interactions that have been encountered during the FMEA.

The second approach has produced the required results and is based on extracting the association between components and function from the FMEA, and is the focus of section 5.1.

The associations produced allow exclusion of measurement m during the generation of symptoms for a state OBS_n^M for which there is no evidence

Symptoms

Id	Description	Faults
S1	When sw_w.Position is "on" : lmp_w.illuminated is "off"	w3-fracture, w4-fracture
S2	When sw_r.Position is "on" : lmp_r.illuminated is "off"	w1-fracture, w2-fracture, w5-fracture, w6-fracture
S3	When sw_g.Position is "on" : lmp_g.illuminated is "off"	w1-fracture, w2-fracture,w7-fracture, w8-fracture
S4	When sw_r.Position is "on" : w1.flow is "inactive"	w1-fracture, w2-fracture, w5-fracture, w6-fracture
S5	When sw_g.Position is "on" : w1.flow is "inactive"	w1-fracture, w2-fracture,w7-fracture, w8-fracture
S6	When sw_r.Position is "on" : w2.flow is "inactive"	w1-fracture, w2-fracture,w5-fracture, w6-fracture
S7	When sw_g.Position is "on" : w2.flow is "inactive"	w1-fracture, w2-fracture,w7-fracture, w8-fracture
S8	When sw_w.Position is "on" : w3.flow is "inactive"	w3-fracture, w4-fracture
S9	When sw_w.Position is "on" : w4.flow is "inactive"	w3-fracture, w4-fracture
S10	When sw_r.Position is "on" : w5.flow is "inactive"	w1-fracture, w2-fracture,w5-fracture, w6-fracture
S11	When sw_r.Position is "on" : w6.flow is "inactive"	w1-fracture, w2-fracture,w5-fracture, w6-fracture
S12	When sw_g.Position is "on" : w7.flow is "inactive"	w1-fracture, w2-fracture,w7-fracture, w8-fracture
S13	When sw_g.Position is "on" : w8.flow is "inactive"	w1-fracture, w2-fracture,w7-fracture, w8-fracture

Figure 7: Symptoms generated for running example system

in the FMEA that m is related to the affected functions $RfnFa_n^M$ and is the subject of section 5.2. Where components implement several functions there are implications for the states required in the FMEA to ensure generated symptoms are not too specific leading to symptoms that not applicable for function combinations not available in the FMEA. Essentially this means that the measurements associated with multiple functions place additional constraints on the functional states that must exist in the FMEA if they are to be used.

5.1. Avoiding over generalisation

There are two parts to the process of restricting the measurements available for symptom generation based on the associated function states.

- Identify which measurements are behaviourally associated with each function. This is described in section 5.2.
- Select the available measurements for symptom generation based on the failure state of functions for each observation. This is described in sections 5.2.1 and 5.2.2

Clearly measurements referred to in the functional model trigger and effect expressions notated (T(fn) and E(fn)) are associated to the function, however most measurements that might be used for diagnosis are not contained in these expressions. In addition those measurements used in triggers and effects are often not directly available to a diagnostic system, particularly an on-board system, where indirect measurements of the inputs and outputs are required.

The concept of a **function associated measurement**(FAM) is used to indicate a measurement has some behavioural or structural 'causal' relationship to a function based on evidence from the FMEA. The measurements associated with fn are denoted by the relations Ab(fn) and Nom(fn) that provide respectively, the abnormal and nominal measurements associated with a function. Often these will simply be different value measurements from the same sensor however this is not assumed.

5.2. Associating measurements to functions

Measurements are associated with system functions based on the achievement and failure of functions across the entire FMEA. Given a comprehensive set of failures this provides a simulation derived description of which functions each component/measurement affects. This in turn allows an assessment of the relevance of measurements associated with a fault; This technique ensures that irrelevant measurements are not included in symptoms, and hence that symptoms are as general as possible. Conversely the decision as to which measurements are relevant is derived from the entire set of failure and non failure states encountered during the FMEA.

Given a comprehensive component fault list, almost all diagnostically interesting measurements (with the exception of externally controlled triggers) will be affected by at least one of the failure

modes of the system. By associating these abnormal measurements with functions that simultaneously fail we build up a FAM mapping. Even measurements structurally adjacent to external triggers will be affected by faults in the connecting components. For example switch.position $\leftrightarrow$ on is a trigger however the current flow through the switch.contact will be affected by faults such as the contact being stuck or wiring faults in the switch circuit and could be used as diagnostic measurements.

It is normal practice to exercise each function individually[4] during an FMEA and therefore abnormal measurements are available for each isolated function. Computing Ab(fn) from an FMEA using the definition of $RfnAb$ and $RobsAb$ from equation 7 and 10 is a matter of aggregating all of the the abnormal observations that occur during states in the FMEA where single functions fail:

$$\begin{aligned} \mathrm{Ab}(fn) &= \bigcup_{\forall M, \forall n \leq |SCN|} \begin{cases} RobsAb_n^M, & \text{if } c \\ \emptyset, & \text{otherwise} \end{cases} \\ c &= RfnFa_n^M \leftrightarrow \{fn\} \end{aligned} \tag{19}$$

Nom(fn) are the measurements in the nominal state minus the nominal measurements from the failure state.

$$\mathrm{Nom}(fn) = \bigcup_{\forall M, \forall n \leq |SCN|} \begin{cases} OBS_n \backslash (OBS_n^M \backslash RobsAb_n^M), & \text{if } c \\ \emptyset, & \text{otherwise} \end{cases} \tag{20}$$

Generally this produces a set of measurements that agree with engineering expectation of the components used to implement each function. However even if this is not clear, there must be a structural or behavioural relationship between the function and the measurement, as required for measurement selection.

Notice that $m \in \mathrm{Ab}(fn)$ specifies only that m can be affected by a failure of fn in at least one state and not that it will necessarily occur in all states when fn fails. Therefore in general $\mathrm{Ab}(fn) \cap \mathrm{Nom}(fn) \neq \emptyset$. Table 1 contains Ab($fn$) and Ab($fn$) for the running example.

Several variations of measurement selection are possible that trade the simplicity of measurement selection against the power of the symptom and the possibility of spurious symptoms. All variations redefine the measurement selection s in equation 12 to provide a restricted set of measurements.

5.2.1. Selection based on function associated measurements

Using the FAM to provide tighter selection of the abnormal measurements, s, gives better protection against spurious symptom generation, when the final system will exercise function combinations not found in the FMEA scenario:

$$s = \{(\mathrm{Ab}_n(fn) \mid fn \in RfnFa_n^M\} \cup \mathrm{T}(fn)) \cap OBS_n^M \tag{21}$$

Allowing nominal observations in addition to triggers will increase the range of symptoms that can be generated:

$$\begin{aligned} s = \quad & \{(\mathrm{Ab}_n(fn) \cup \mathrm{Nom}_n(fn)) \mid fn \in RfnFa_n^M\} \\ & \cup \mathrm{T}(fn) \cap OBS_n^M \end{aligned} \tag{22}$$

In particular, for systems where a fault only causes artefacts in part of the affected function(s) behaviour, it allows symptoms to be generated that identify the internal inconsistency in the function. For example if one branch of a parallel circuit (providing a single function) fails then symptoms are generated that measure nominal flow in one branch and abnormal in another. It is thus important that the major sub-states of the function are exercised, such as any available control of the branches of a parallel circuit, particularly if 'fully instrumented' systems are being investigated as a prelude to sensor selection.

Fully instrumented systems make large numbers of observations available for symptom generation, and many alternative symptoms can be generated representing alternative combinations of internal measurement inconsistencies. An engineer can then use external information and diagnosability tools [22] to select the most convenient measurements to achieve the required diagnosability, for example.

5.2.2. Selection based on shared function measurements

Measurements selected using equation 21 and 22 can still lead to spurious diagnoses for novel functional combinations that *share components* due to false exoneration. To illustrate the issue, figure 8

[4] It is not always possible to exercise each function individually, however this also makes these operating modes impossible during nominal operation, and therefore the associated measurements can simply be captured for the required groups of functions

Function, fn	Function associated nominal measurements, Nom(fn)
produce_red_light	lmp_r.illuminated:on, w1.flow:active, w2.flow:active, w5.flow:active, w6.flow:active
produce_green_light	w1.flow:active, w2.flow:active, lmp_g.illuminated:on, w7.flow:active, w8.flow:active
produce_white_light	lmp_w.illuminated:on, w3.flow:active, w4.flow:active,
	Function associated Abnormal Measurements, Ab(fn)
produce_red_light	lmp_r.illuminated:off, w1.flow:inactive, w2.flow:inactive, w5.flow:inactive, w6.flow:inactive
produce_green_light	lmp_g.illuminated:off, w1.flow:inactive, w2.flow:inactive, w7.flow:inactive, w8.flow:inactive
produce_white_light	lmp_w.illuminated:off, w3.flow:inactive, w4.flow:inactive

Table 1: FMEA derived function associated measurements for the running example

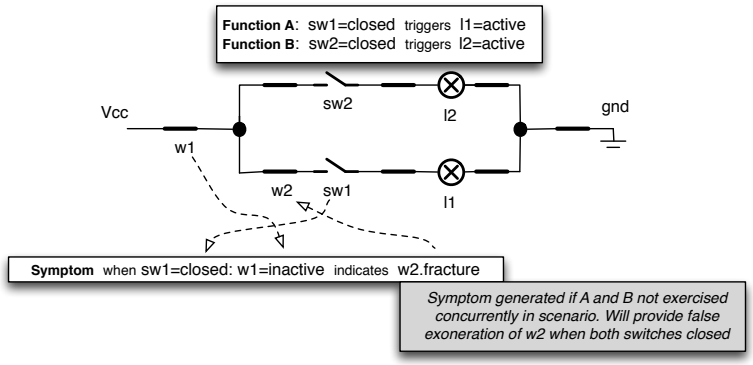

Figure 8: False exoneration caused by a limited FMEA scenario

depicts a simple circuit that implements two functions A and B that share some components including w1. Assume the functions A and B were (for reasons explained in section 5.2) not exercised simultaneously during the FMEA. The symptom shown in the diagram is generated because in all scenario states where the fault exists and sw1=closed then w1 is inactive, and both measurements will clearly be associated with the failed function. This symptom exonerates the fault when both functions are triggered because the symptom is valid (sw1=closed) but w1 is active allowing the symptom to evaluate false.

One solution is to exercise both of the functions simultaneously in the scenario, thereby creating a state where w1 is active when sw1=closed (because current will flow for function B when A fails) forcing the symptom generation to include additional measurements in the symptom, for example the position of switch sw2. As described previously, for some systems it is possible to activate all of the system functions simultaneously in the FMEA scenario, however complex systems often have limitations on the combinations of attainable functions precluding this approach, and in some cases specific combinations of functions may be required. A 'safe' solution is therefore to disallow measurements associated with several functions if any of the functions is In or Un.

If ASh is defined as the abnormal measurements that are not shared with inactive function combinations and given $fn \in RfnFa_n^M$ and $fs \in (RfnIn_n^M \cup RfnUn_n^M)$, the measurement selection in 21 becomes:

$$ASh = \{x \mid x \in \mathrm{Ab}_n(fn) \wedge x \notin \mathrm{Ab}_n(fs)\}$$

$$s = (\{x \in ASh \mid fn \in RfnFa_n^M\} \cup \mathrm{T}(fn)) \cap OBS_n^M \tag{23}$$

Similarly for the nominal measurements NSh, the measurement selection in 22 is modified to:

$$NSh = \{x \mid x \in \mathrm{Nom}_n(fn) \wedge x \notin \mathrm{Nom}_n(fs)\}$$

$$s = (\{x \in (ASh \cup NSh) \mid fn \in RfnFa_n^M\} \cup \mathrm{T}(fn)) \cap OBS_n^M \tag{24}$$

This may prevent some faults being diagnosed, and a reduction in the number of operating states where faults are diagnosable. If a significant difference in the diagnosability of the system exists

between equations 24 and 22 the function combinations causing measurements to be abandoned based on function sharing can be reported and the scenario can then be extended. Alternatively in some situations, for example where functions are mutually exclusive (activated by different positions of a switch), they can be included in a list of allowable unexercised function combinations for shared measurements. There may be scope for determining mutually exclusive functions from the function model, but this has not been investigated because in practice the issue has not arisen.

Neither of the measurement selection mechanisms above make any use of unexpected function achievement to produce symptoms. This is for two reasons.

- Symptoms based on unexpected functions would require the *absence* of trigger measurements in an observation to be included as part of the symptom expression making it necessary to identify the sets of mutually exclusive measurements that form $\neg \mathrm{T}(fn)$ to allow suitable measurements to be selected that represent the trigger absence.
- Since $\mathrm{Nom}(fn)$ are required for $\mathrm{Un}(fn)$, it is necessary to ensure that all abnormal (failed and unexpected) functions present in an observation also exist in their achieved form in the nominal observations. Failing to do this would allow symptoms based on nominal measurements of function combinations that have not been observed in the nominal state.

To complete the running example, we can see from table 1 that w1 and w2 have measurements shared between the red and green functions. The effect of this on symptoms generated using measurement selection in equation 24 is to remove symptoms 4-7 from those in figure 7. In addition no w1 and w2 faults are diagnosed by the remaining symptoms. This is because these components are shared between two functions that were never exercised simultaneously in the scenario. To allow these components to be diagnosed, and also for their measurements to be usable, the scenario is extended to include the state where both red and green functions are activated (the symptom generator can advise which shared measurements are unusable and which function combinations are required to allow them). The functional model also indicates there is a dependency between these functions because they share the sw_Imp.Position on trigger, therefore the state where sw_Imp.Position on, sw_r.Position off, sw_g.Position off is also included in the scenario.

The resulting symptom set in figure 9 provides no spurious diagnoses in any system state. An exhaustive analysis of the evaluation of the symptom set for all of the 16 possible states of the system reveals that these symptoms place the actual fault in the top rank set for each fault as shown in figure 10. The two numbers a/b in each cell of the table indicate that the inserted fault was in the top ranking set of faults (a=*1*) and the how many other faults b, were contained in the top ranking diagnosis. The ranking used is simply a count of the symptoms indicating the fault minus symptoms exonerating the fault. For example, the state representing all switches on (rightmost column) shows 1/2 for the w1.fracture fault the because the diagnosis included this fault in the top ranking faults, of which there were two. The additional information F6 S10 for this entry indicates that there were a total of six positively ranked faults and ten symptoms that were valid (either for fault indication or exoneration).

6. Diagnosis framework

The symptom set can be used to analyse diagnosability and fault isolation for a given set of measurements. The symptoms are qualitative in nature and an engineer must determine thresholds or apply additional techniques for these values if they are to be used for an on board diagnostic system. One application where the symptoms are used as the measurement-fault mapping of a Bayesian network could adjust the symptom weights based on a fuzzy description of the qualitative values for example.

For systems with a well defined set of sensor outputs, the diagnosable components and states can be determined by simple analysis of the symptoms generated. In another application the task is to determine the best n measurements that can diagnose the maximum number of faults. Since it is trivial to generate symptoms based on any set of system variables it is possible to use the symptoms to assist in the selection of potential sensors. One tool to do this was described in [22].

The ASTRAEA target application uses the generated symptoms in a Bayesian network to produce final diagnoses by adding probabilities to the symptom elements;

Symptoms

Id	Description	Faults
S1	When sw_w.Position is "on" : Imp_w.illuminated is "off"	w3-fracture, w4-fracture
S2	When sw_Imp.Position is "on" , sw_r.Position is "on" : Imp_r.illuminated is "off"	w1-fracture, w2-fracture, w5-fracture, w6-fracture
S3	When w1.flow is "active" , sw_r.Position is "on" : Imp_r.illuminated is "off"	w5-fracture, w6-fracture
S4	When w2.flow is "active" , sw_r.Position is "on" : Imp_r.illuminated is "off"	w5-fracture, w6-fracture
S5	When sw_Imp.Position is "on" , sw_g.Position is "on" : Imp_g.illuminated is "off"	w1-fracture, w2-fracture, w7-fracture, w8-fracture
S6	When w1.flow is "active" , sw_g.Position is "on" : Imp_g.illuminated is "off"	w7-fracture, w8-fracture
S7	When w2.flow is "active" , sw_g.Position is "on" : Imp_g.illuminated is "off"	w7-fracture, w8-fracture
S8	When w1.flow is "inactive" , sw_Imp.Position is "on" , sw_r.Position is "on" :	w1-fracture, w2-fracture, w5-fracture, w6-fracture
S9	When w1.flow is "inactive" , sw_Imp.Position is "on" , sw_g.Position is "on" :	w1-fracture, w2-fracture, w7-fracture, w8-fracture
S10	When w2.flow is "inactive" , sw_Imp.Position is "on" , sw_r.Position is "on" :	w1-fracture, w2-fracture, w5-fracture, w6-fracture
S11	When w2.flow is "inactive" , sw_Imp.Position is "on" , sw_g.Position is "on" :	w1-fracture, w2-fracture, w7-fracture, w8-fracture
S12	When sw_w.Position is "on" : w3.flow is "inactive"	w3-fracture, w4-fracture
S13	When sw_w.Position is "on" : w4.flow is "inactive"	w3-fracture, w4-fracture
S14	When sw_Imp.Position is "on" , sw_r.Position is "on" : w5.flow is "inactive"	w1-fracture, w2-fracture, w5-fracture, w6-fracture
S15	When w1.flow is "active" , sw_r.Position is "on" : w5.flow is "inactive"	w5-fracture, w6-fracture
S16	When w2.flow is "active" , sw_r.Position is "on" : w5.flow is "inactive"	w5-fracture, w6-fracture
S17	When sw_Imp.Position is "on" , sw_r.Position is "on" : w6.flow is "inactive"	w1-fracture, w2-fracture, w5-fracture, w6-fracture
S18	When w1.flow is "active" , sw_r.Position is "on" : w6.flow is "inactive"	w5-fracture, w6-fracture
S19	When w2.flow is "active" , sw_r.Position is "on" : w6.flow is "inactive"	w5-fracture, w6-fracture
S20	When sw_Imp.Position is "on" , sw_g.Position is "on" : w7.flow is "inactive"	w1-fracture, w2-fracture, w7-fracture, w8-fracture
S21	When w1.flow is "active" , sw_g.Position is "on" : w7.flow is "inactive"	w7-fracture, w8-fracture
S22	When w2.flow is "active" , sw_g.Position is "on" : w7.flow is "inactive"	w7-fracture, w8-fracture
S23	When sw_Imp.Position is "on" , sw_g.Position is "on" : w8.flow is "inactive"	w1-fracture, w2-fracture, w7-fracture, w8-fracture
S24	When w1.flow is "active" , sw_g.Position is "on" : w8.flow is "inactive"	w7-fracture, w8-fracture
S25	When w2.flow is "active" , sw_g.Position is "on" : w8.flow is "inactive"	w7-fracture, w8-fracture

Figure 9: Enhanced Symptoms generated for running example system

- The component fault **prior** probability describes component reliability.
- The symptom **leak** probability captures the probability that the symptom is observed even though there is actually no fault.
- Fault symptom **conditional** probability captures the probability that when the symptom is observed it is due to the fault indicated rather than any other fault. This can be useful for noisy signals for example.

The probabilities allow for intermittent or unreliable measurements to be used. They also allow the characteristics of the mapping between quantitative measurements and qualitative values (e.g. numerical thresholding) to be taken into account as a certainty measure on the symptom for numerical measurement ranges that cannot be clearly mapped to the qualitative values used.

For the purposes of investigating the diagnosability properties of a system and validating symptoms at the qualitative level, another tool (figure 11) allows the system to be exercised interactively, and any combination of faults inserted. The figure uses the symptoms from figure 9. The system can be exercised in the top section of the tool interface, measurement visibility selected in the center left section. The diagnosis is presented in the lower section, with the valid symptoms that implicate or exonerate faults. The 'I/E' column is shown ticked if the symptom indicates its associated faults and not ticked if it exonerates them. The faults implicated by any symptom can be shown by selecting it (S12 in the figure), and finally a fault ranking is given based on number of satisfied and exonerated faults.

6.1. Multiple faults

The diagnostic system can only explicitly diagnose multiple faults if a multiple fault FMEA is performed. Previous work has described how a multiple fault FMEA can be efficiently performed using this technology [18, 17, 14], and so there is no conceptual barrier to diagnosing multiple faults. Multiple fault FMEA analysis can by run for example by selecting only combinations of faults with higher combined component failure likelihoods. The multiple faults are then considered as a single compound fault within symptoms. If f_{12} represents simultaneous occurrence of two failures $f_1 \in \mathcal{M}$ and $f_2 \in \mathcal{M}$, the symptom set could be modified by

Elements are shown as fault-group rank/equivalent-rank-faults
F indicates total number of positively implicated faults S indicates total number of activated and satisfied symptoms

SETTING:																								
sw_w.Position	**'off'**	**'on'**		'on'		'on'			'on'			'on'			'on'			'on'				'on'	
sw_Imp.Position	**'off'**		**'on'**	'on'				**'on'**	'on'					**'on'**	'on'					**'on'**		'on'	
sw_g.Position	**'off'**				'on'	'on'		**'on'**	'on'							**'on'**	'on'			**'on'**		'on'	
sw_r.Position	**'off'**									'on'	'on'	**'on'**		'on'	**'on'**	'on'				**'on'**		'on'	
Nominal																								
w1.fracture								**1/4 F4 S5**	1/4 F4 S5			**1/4 F4 S5**	1/4 F4 S5							**1/2 F6 S10**	1/2 F6 S10	
w2.fracture								**1/4 F4 S5**	1/4 F4 S5			**1/4 F4 S5**	1/4 F4 S5							**1/2 F6 S10**	1/2 F6 S10	
w3.fracture		**1/2 F2 S3**		1/2 F2 S3		1/2 F2 S3			1/2 F2 S3		1/2 F2 S3		1/2 F2 S3		1/2 F2 S3						1/2 F2 S3	
w4.fracture		**1/2 F2 S3**		1/2 F2 S3		1/2 F2 S3			1/2 F2 S3		1/2 F2 S3		1/2 F2 S3		1/2 F2 S3						1/2 F2 S3	
w5.fracture												**1/4 F4 S5**	1/4 F4 S5							**1/2 F2 S9**	1/2 F2 S9	
w6.fracture												**1/4 F4 S5**	1/4 F4 S5							**1/2 F2 S9**	1/2 F2 S9	
w7.fracture								**1/4 F4 S5**	1/4 F4 S5											**1/2 F2 S9**	1/2 F2 S9	
w8.fracture								**1/4 F4 S5**	1/4 F4 S5											**1/2 F2 S9**	1/2 F2 S9	

Figure 10: Exhaustive validation of symptoms for running example

multiple fault FMEA in one or more of the following ways:

1. A unique symptom, $S_{NEW} = (E, \{f_{12}\})$ is produced that would allow the multiple fault to be isolated.
2. A symptom, $S_n = (E, \{f_3, f_{12}\})$, is produced that is identical to the symptom for a different fault.
3. Symptoms are produced that are the same as some or all of the individual faults. $S_n = (E, \{f_1, f_{12}\})$ or $S_n = (E, \{f_2, f_{12}\})$ or $S_n = (E, \{f_1, f_2, f_{12}\})$.
4. The multiple fault case produces no symptoms.

Without using multiple fault FMEA, we find empirically for the systems tested, the most common effect of multiple faults is that all or some of the individual faults appear in the highest rank of possible faults (case 3 above). For case 1 and 4 above the multiple fault is not diagnosed. If only case 4 occurs then the multiple fault is not diagnosable, this could be two faults that negate each other and provide no overall effect on the system.

Case 2 is the only one where the multiple fault could lead to a spurious diagnosis; f_3 would be diagnosed if f_1 and f_2 occurred simultaneously. Empirically we have found that the most likely situation is a case of fault masking where all the faults are associated to the same function and f_3 masks the multiple fault. Parsimonious principles dictate that the diagnosis f_3 would be a preferred initial diagnosis, and is the result provided by the symptoms produced without multiple faults. Once f_3 is exonerated, by exercising additional faults or including additional measurements, without the multiple fault simulation the fault could not be diagnosed. For example, if two lamps wired in parallel both fail: the effect is the same as the main supply failing, and given that it is impossible to distinguish these faults the single fault is more likely unless the supply can be exonerated.

7. Example systems

We present the results of the symptom generation for two case studies.

The Aircraft Fuel system was a test system used as a technology demonstrator, and had the advantage of a laboratory based physical model that included all of the relevant tanks, pumps and valves including various fault injection mechanisms, allowing a physical validation of the diagnosis. For this system, the sensors were already defined and therefore there was a relatively small number of measurements to be made available for symptom generation. The task was to produce symptoms that indicated faults, rather than perform sensor selection based on likely diagnosability.

The Automotive example was an electrical exterior day time running lighting system (DTRL). We use this example to illustrate the ability to generate diagnostics from an FMEA that did not exercise all function combinations, and indeed did not exercise one entire function, demonstrating that the resulting diagnosis is able to diagnose all faults except those related to the implementation of the disregarded function. This example was also used to illustrate symptom generation based on 'high observability' by allowing the current flow in every wire component to be used to generate symptoms.

For both systems the symptom generation process takes a tiny fraction of the processing time that the FMEA generation takes. The simulation and symptom generation is coded in Java and has not been optimised in any sense. A 2.4Ghz Core 2 laptop completes FMEA generation for either system in around 2-30s per fault for a sensible scenario requiring of the order of an hour in total. The symp-

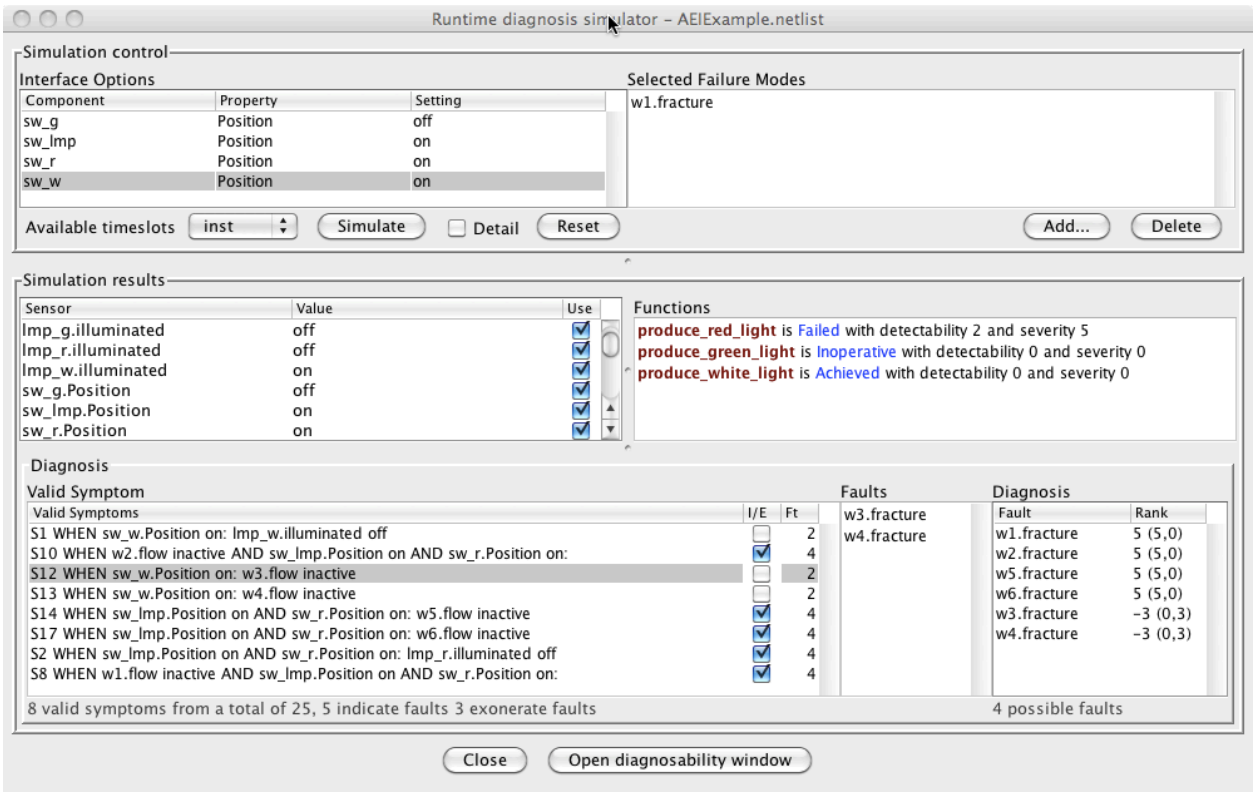

Figure 11: Tool interface used to manually investigate diagnosability of the example system

tom generation takes less than a minute in total for either system. Evaluating the symptom set for a specific set of observations requires evaluation of a small number of simple logical and arithmetic expressions and is extremely fast (milliseconds), requires only a few hundred lines of code, and has minimal memory requirements. The running example system takes around 5s to produce the FMEA and 2s to produce the symptoms, but most of this time is initialising the Java virtual machine.

From an engineers perspective the major inputs to the system are a system schematic (components and connections) and a set of qualitative component models that define the structure and behaviour of the components. Component models may comprise a resistance network connecting component terminals, and state machines that describe higher level behaviour or non linear behaviour aspects. Such models are usually obtained from a library, and the qualitative nature of the modelling allows generalised modelling of component classes. One possible modelling approach for electrical systems [20] has been extended to allow modelling of systems from other domains such as hydraulic, fluid flow and thermodynamic by utilising generalised power and energy flow models using a qualitative approach related to the well known Bond graph theory.

For non switching systems qualitative distinctions may appear too coarse to capture the required behaviour, however this is not necessarily the case. Additional techniques such as exaggeration reasoning [25] may be used to define faults. For example in a fluid flow system a leak in a pipe may not produce a qualitatively significant change in flow, however if we treat the extreme case of the leak as a fracture then the flow in the system will have changed qualitatively. The FMEA (and subsequent symptoms) can therefore report that a potential symptom for the leak may be high or low flow in the relevant parts of the system.

The qualitative modelling and resulting symptoms provide comprehensive analysis of the system covering all qualitatively distinct regions of nominal and failure behaviour. The main requirement for successful FMEA and symptom generation is that the qualitative values utilised in the system simula-

tion description are able to represent the significant system states and failure effects.

The second task for the engineer is to capture the system functions in the functional model outlined in section 3.2. These models identify the triggers and effects that relate to each system function and can usually be defined with a set of relatively simple logical expressions relating system inputs to expected outputs. Where necessary, the functional model can be sophisticated, involving hierarchical functional structures with propositional logical (and temporal logic) expressions to define the required behaviour [2].

A scenario is provided by exercising the system functions, allowing an FMEA is automatically generated and considered by engineers as per usual practices. No further input is required to produce the symptoms which are generated for any abnormal behaviour that has qualitatively significant behavioural deviations. The simulation and FMEA techniques have been widely used for electrical systems with switching behaviour [13] where a qualitative order of magnitude approach [21, 12] allows distinctions between power circuit and signal circuit behaviour to be made, for example. Recently, fluid flow systems have also been successfully modelled, and the energy flow based structural modelling provides an expectation that the methods will be applicable to other domains also.

7.1. Aircraft fuel system technology demonstrator

The system in figure 12 comprised approximately 98 components with failure modes, including 4 tanks, 8 multiway valves and 7 pumps. There were 239 faults considered in the FMEA including full and partial blockages, leaks, valve stuck failures, tank leaks, and pump failures. Measurements included values from several pressure and flow sensors, valve position actuator request and microswitch tell-back values, and tank level measurements. The difference between the number of symptoms using abnormal measurements only (64) and the symptoms generated using both abnormal and nominal measurements (104) is not so marked with this system, due to the relatively small quantity of measurements available. An example of part of the symptom expression set is shown in figure 13. PT refers to pressure transducer measurements, TVL are multiway valves, and CP are pumps. This system required no more than two measurements to form any symptom and each symptom is associated with between 1 and 40 faults (not shown). The colon symbol is used to separate the symptom condition from the main observations. The symptoms containing only a condition are effectively not negatable and cannot exonerate faults, since it is valid and considered satisfied when all of the measurements are observed, and can never be valid and unsatisfied.

Figure 14 shows a breakdown of the number of faults that can be diagnosed in 17 of the main operating states of the system. These are 6 major functions available based on various valve configurations, and it is clear that there is one major operating state for each function that allows the majority of the faults for that function to be diagnosed. The other operating states for each function are where valves have been configured but the pumps are not activated, and as expected many faults cannot be detected in these states. The columns representing the number of faults in each state are subdivided according to the number of other faults that ranked equal top in the diagnosis. Given the limited amount of sensing available for the system, there are inevitably significant numbers of faults that are indistinguishable from the measurable observations; the diagram however only considers individual states, some faults can be isolated by switching states, for example components that contribute to several functions can be isolated this way. Finally the symmetrical structure of the system is revealed by the symptom set; the system has left and right halves and the ability to feed either engine from either wing tank. It is no surprise that the diagnosability of left and right, normal and cross fuel feed is similar.

7.2. Daytime running lights system

This system was supplied by Sumitomo Electrical Wiring Systems Europe Ltd and comprised 166 electrical components, including several multiple contact switches, lamp clusters and splices. 63 component faults were considered for the analysis comprising mostly of wire fractures since these provide good coverage of the circuit topology. The current flow in all wires was made available as a measurement, together with the components that normally produce an input or output such as switches and lamps. The system is designed such that the high beam can be activated if the main switch is in the any position, the sidelights are activated if the main switch is in the sidelights position, and the dip and main beam are both available when the main switch is in the mainlights position dependent on the dip

switch position. The FMEA activated the following function states: off-dipbeam, off-highbeam, sidelights-dipbeam, mainlights-dipbeam. The daytime running function was *not* activated for the purposes of illustration. The number of symptoms generated depends on the measurement selection strategy and is shown in table 2.

The diagnostics were verified by running the simulation for each fault for a set of operational states. These states included the two major operational states that were missing from the original scenario: sidelights-highbeam, mainlights-highbeam. These are highlighted in table 2 which summarises the results of testing the diagnostic system for each of the symptom generation strategies from selection strategies in equations 18-24. The table shows the number of correctly detected faults (inserted fault was in the top or equal top ranking predicted faults), with the number of incorrectly identified faults (higher ranking than the actual fault) in square brackets. For this system the real fault was never worse than second ranking, with between 25 and 2 higher equal ranking predicted faults.

It is clear in column 18 that simply using abnormal measurements and triggers leads to incorrect fault identification when the system enters new states that were not exercised in the FMEA scenario, as expected. Allowing FAM associated with failure of the function significantly reduces the number of symptoms and also reduces the number of mis-identification of faults (column 21). Allowing FAM associated with nominal operation maintains the reduction in mis-diagnoses, however it increases the number of symptoms by allowing abnormal relationships within function activity to be used as symptoms. This will provide more flexibility in the choice of measurements that might be used (since the final diagnostic system will not have the ability to monitor every wire). A tool to display the measurement-symptom-fault relationships visually is described in [22]. The final two columns show the effect of excluding all measurements associated with shared functions that have not been activated simultaneously for the same setup as 21 and 22 respectively. The two faults that were mis-detected due to spurious fault exoneration no longer exist, however the total number of faults that can be diagnosed is also reduced. The diagnosis generator reports that some measurements were unable to be used because the scenario found no instances where the sidelights function and highbeam function were failed or achieved together. This then would provide a direct indication to the engineer that this operating mode could be added to the scenario to improve the symptom set, since the failure behaviour of the system shows that they share components/or behaviour and could therefore interact.

Some of the faults could not be detected at all because part of the circuit functionality was not exercised in the FMEA. This is to be expected for any fault in the FMEA that has the report "No external behaviour or functional effects". There were 15 such faults in the FMEA and therefore the maximum number of faults that could be expected to be detected based on the FMEA information is 48. Figure 15 shows a breakdown of the number of faults that can be diagnosed in the 6 main operating states of the system using measurement selection methods in equation 23 (or 24) and full measurement visibility. State 4 and 6 were not exercised in the scenario but clearly produce good fault detection. The columns are subdivided according to the number faults that ranked equal top with the actual fault in the diagnosis. In this system the maximum number of faults that cannot be distinguished is 9. A selection of these were analysed in detail and are generally wire failures in series circuit elements that could not be isolated further without external intervention. There were no faults in the first operating state because activating the dipbeam function with the ignition off results in no electrical change in the circuit and merely a change to the switch position.

8. Conclusion and future enhancements

The initial concept for this work was to replace the use of manual FMEA as the input to a Bayesian network based diagnostic system with an automated FMEA. It rapidly became clear that the comprehensive nature of an automated FMEA actually makes the identification of general symptoms more time consuming - although the resulting diagnostic system is more powerful. Automation of the generation of the symptoms was an obvious next step, although emulating the selectivity and additional knowledge imparted by an engineer proved to be a challenge, and was finally addressed by the use of the functional model that already plays the role of interpreting detailed behaviour into a more meaningful abstracted description.

An unexpected - and for the ASTRAEA project an exciting possibility - was the ability to generate symptoms based on all potentially measurable quantities in a system. The resulting large set of

	Measurement selection strategy				
	18	21	22	23	24
Symptoms Generated	1014	72	968	52	828
Total diagnosable faults	48	48	48	46	46
Operational State					
dipbeam	0	0	0	0	0
highbeam	13	13	13	11	11
sidelights-dipbeam	31	31	31	29	29
sidelights-highbeam	**13[25]**	**36[2]**	**36[2]**	**36**	**36**
mainlights-dipbeam	17	17	17	15	15
mainlights-highbeam	**20[4]**	**22[2]**	**22[2]**	**22**	**22**

Table 2: Fault detection for DTRL system (63 faults available in the FMEA)

symptoms (1000+ for typical automotive system examples) paves the way for a sensor selection tool that allows an engineer to quickly analyse which measurements may be most useful for a given set of 'diagnosability' criteria. The potential diagnosability of a system can therefore be investigated in broad qualitative terms early in the design, possibly leading to alternative solutions that build diagnosis into preliminary design rather than as a retrofit activity. An early version of this tool is documented in [22].

Currently, symptoms are based on qualitative system states and do not include sequences of states in the description of behaviour that may indicate a fault; however, fault isolation may be carried out by considering symptoms related to multiple system states. Systems that include internal (hidden) state that affects fault behaviour will result in no symptom being generated for affected faults in the relevant operating modes due to a lack of observations able to distinguish the faults. These can then be highlighted to the engineer and if necessary some mechanism for providing the additional state to the diagnostic system can be included.

The technique is applicable to any system for which the automated FMEA can be generated [14, 15, 24]. Applicable systems have behaviours and faults of interest that can be represented qualitatively and these types of systems tend to be topologically complex with component behaviours represented by a relatively small number of linear behaviour regions. This allows the symptoms produced to cover qualitative regions of behaviour. Systems where the behaviour is highly nonlinear at the level of abstraction required or where the measurements required for diagnosis are not qualitatively significant, will not be amenable to the analysis because the system behaviour does not form a reasonable finite state description.

The functional model had not yet been utilised to its full potential in symptom generation. For systems with complex functional dependencies such as warning, fault mitigating, interlocking, recharging functions that may be described in the FIL, as well as functions that share triggers and effects we believe the functional model can provide additional information to ensure the FMEA scenario includes the worst case faults and effects, and also ensure the relevant states are included that allow a comprehensive set of symptoms to be generated. Further investigation is planned in this area.

In summary, this paper demonstrates the generation of a diagnostic system from the results of an automated FMEA. We have deployed software using the algorithms described here on several real systems including an aircraft fuel system that was the main diagnostic case study for one of the ASTRAEA [9] project. The focus of this work was on single component failure modes, however many multiple failure modes are completely or partially diagnosed and possibilities are available to extend the analysis into multiple failure modes if required at the computational expense associated with additional simulation.

8.1. Acknowledgements

This work was supported by Aberystwyth University, the Welsh Assembly Government, BAE Systems and the DTI ASTRAEA Programme. We are also grateful to the anonymous AEI reviewers for their valuable comments.

References

[1] Bell, J., 2006. Interpretation of simulation for model based design analysis of engineered systems. Phd thesis, University of Wales Aberystwyth, email: jpb@aber.ac.uk.

[2] Bell, J., Snooke, N. A., 2004. Describing system functions that depend on intermittent and sequential behavior. In: Proceedings 18th International Workshop on Qualitative Reasoning, QR2004. pp. 51–57.

[3] Bell, J., Snooke, N. A., Price, C. J., Oct 2007. A language for functional interpretation of model based simulation. Advanced Engineering Informatics 21 (4), 398–409.

[4] Cascio, F., Console, L., Guagliumi, M., anda. Panati, M. O., Sottano, S., Theseider-Dupré, D., 1999. Strategies for on-board diagnostics of dynamic automotive systems usingqualitative models. AI Communications.

[5] Chandrasekaran, B., Josephson, J. R., 2000. Function in device representation. Engineering with Computers 16 (3–4), 162–177.

[6] Chantler, M. J., Coghill, G. M., Shen, Q., Leitch, R. R., 1998. Selecting tools and techniques for model based diagnosis. Artificial Intelligence in Engineering 12, 81–98.

[7] Coghill, G. M., 2009. Incremental state based diagnosis. Advanced Engineering Informatics 23, 309–322.

[8] de Kleer, J., 1987. Diagnosing multiple faults. Artificial Intelligence 32 (1), 97–130.

[9] Downes, C., November 2007. ASTRAEA T7 – an architectural outline for system health management on civil UAVs. In: 2nd Autonomous Systems Conference. Institution of Engineering and Technology.

[10] Forbus, K., 1990. The qualitative process engine. In: Weld, D., de Kleer, J. (Eds.), Readings in Qualitative Reasoning about Physical Systems. Morgan Kaufmann, pp. 220–235.

[11] Genesereth, M., 1984. The use of design descriptions in automated diagnosis. Artificial Intelligence 24 (1-3), 411–436.

[12] Lee, M. H., 2000. Many-valued logic and qualitative modelling of electrical circuits. In: Proceedings 14th International Workshop on Qualitative Reasoning, (QR-2000).

[13] Mentor Graphics, July 2012. Capital harness product. URL `http://www.mentor.com/products/electrical-design-software/capital/harness-classic`

[14] Price, C. J., 1998. Function-directed electrical design analysis. Artificial Intelligence in Engineering 12 (4), 445–456.

[15] Price, C. J., Pugh, D. R., Snooke, N. A., Hunt, J. E., Wilson, M. S., 1997. Combining functional and structural reasoning for safety analysis of electrical designs. Knowledge Engineering Review 12 (3), 271–287.

[16] Price, C. J., Snooke, N. A., Lewis, S. D., 2006. A layered approach to automated electrical safety analysis in automotive environments. Computers in Industry 57 (5), 451–461.

[17] Price, C. J., Taylor, N. S., 1997. Multiple fault diagnosis from FMEA. In: Proc. The Ninth Conference on Innovative Applications of Artificial Intelligence (IAAI 97). AAAI, pp. 1052–1057.

[18] Price, C. J., Taylor, N. S., 2002. Automated multiple failure FMEA. Reliability Engineering and System Safety 76 (1), 1–10.

[19] Snooke, N., Price, C., Downes, C., d Aspey, C., April 2008. Automated failure effect analysis for PHM of UAVs. In: International System Safety and Reliability Conference, (ISSRC 2008). Singapore.

[20] Snooke, N. A., 1999. Simulating electrical devices with complex behaviour. AI Communications 12 (1,2), 45–58.

[21] Snooke, N. A., 2007. M^{2}CIRQ: Qualitative fluid flow modelling for aerospace fmea applications. In: Proceedings 21st international workshop on qualitative reasoning. pp. 161–169.

[22] Snooke, N. A., 2009. An automated failure modes and effects analysis based visual matrix approach to sensor selection and diagnosability assessment. In: online proc. Prognostics and Health Management Conference (PHM09). PHM Society, http://www.phmsociety.org/references/proceedings.

[23] Struss, P., 2004. Models of behaviour deviations in model-based systems. In: R.Lopez, Saitta, I. (Eds.), 16^{th} European Conference on Artificial Intelligence. ECAI, IOS Press, pp. 883–887.

[24] Struss, P., Fraracci, A., 2011. FMEA of a braking system - a kingdom for a qualitative valve model. In: 25th International Workshop on Qualitative Reasoning.

[25] Weld, D. S., 1990. Exaggeration. Artificial Intelligence 43 (3), 311 – 368.

[26] Williams, B., Nayak, P., 1996. A model-based approach to reactive self-configuring systems. In: Proceedings of AAAI-96. pp. 971–978.

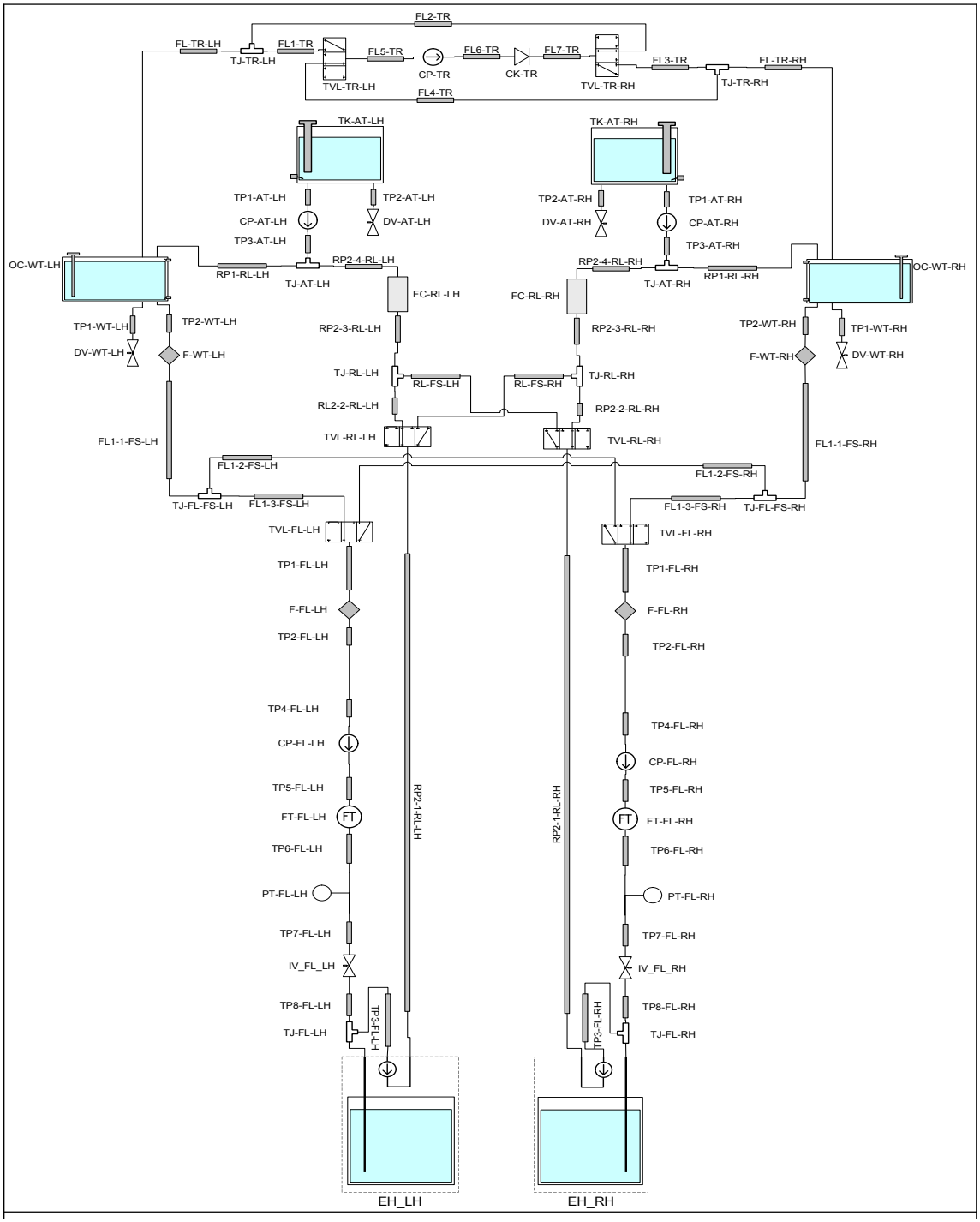

Figure 12: Fuel system schematic

S4	When PT_FL_RH.presssure is "none" , CP_FL_RH.Control is "on" :
S5	When TVL_RL_LH.position is "normal" : TVL_FL_LH.position_tellback is "isolation"
S6	When TVL_FL_LH.position is "normal" : TVL_FL_LH.position_tellback is "isolation"
S7	When CP_FL_LH.Control is "on" : TVL_FL_LH.position_tellback is "isolation"
S8	When TVL_RL_LH.position is "crossover" : TVL_FL_LH.position_tellback is "isolation"
S9	When TVL_FL_LH.position is "crossover" : TVL_FL_LH.position_tellback is "isolation"
S10	When CP_AT_LH.Control is "on" : TK_AT_LH.tank_level is "no level change"
S11	When CP_AT_RH.Control is "on" : TK_AT_RH.tank_level is "no level change"

Figure 13: Example of symptom expressions generated for fuel system

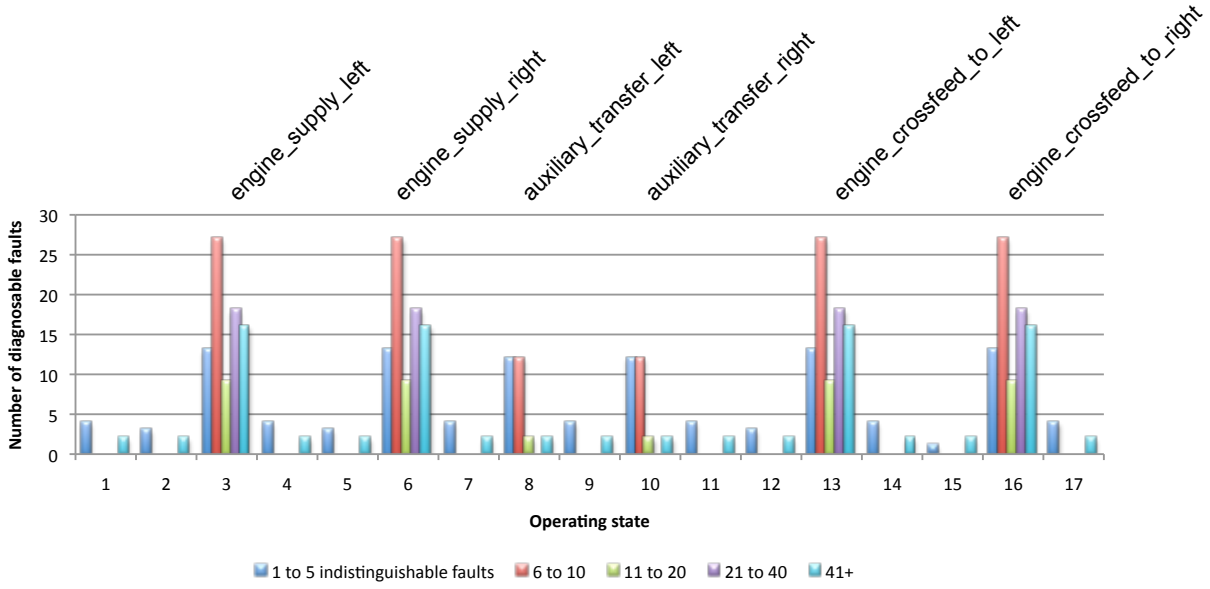

Figure 14: Quantity of faults diagnosable for major operating fuel system states

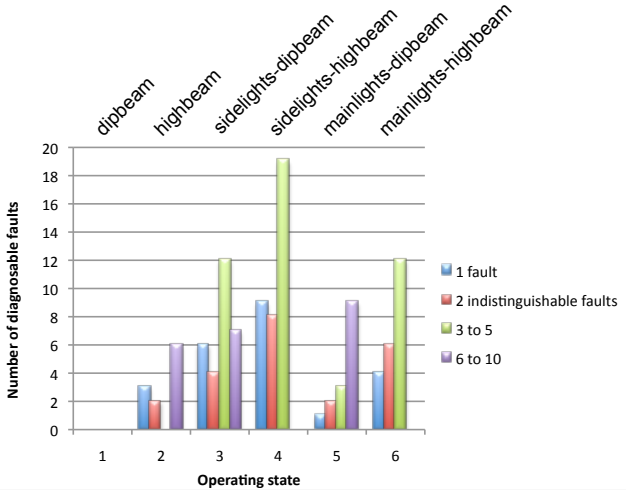

Figure 15: Quantity of faults diagnosable for major DTRL operating states