An Effective Practical Model-Based Detectability Tool

Neal Snooke [1], **Chris Price** [1]

[1] *Aberystwyth University, Department of Computer Science, Ceredigion, SY23 3DB, United Kingdom*
nns{cjp}@aber.ac.uk

ABSTRACT

This paper builds on the ability to produce a comprehensive automated set of component fault–observation relations for vehicle on-board systems using qualitative model based reasoning techniques. Observations are typically expensive (in the broadest sense) and the problem addressed by this paper is how to allow selection of a set of observations that fulfils the detectability/diagnosability requirements of the system. This paper documents a tool that provides an engineer with easy access to information about diagnostic capability via a matrix visualisation technique. The focus of the work was for the fuel system of an Uninhabited Aerial Vehicle (UAV) although the tool has also been used on automotive electrical systems, and is applicable to a wide range of schematic and component based systems.

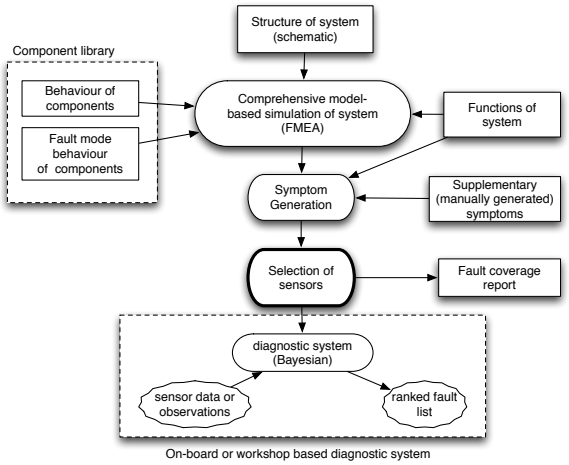

Figure 1: Detectability Process Diagram

1 INTRODUCTION

This paper describes a tool for assisting engineers in deciding what sensors are needed on a system and what observations need to be made in order to be able to detect the occurrence of all possible component failures in that system. The paper also considers the extensions needed to make the detectability tool into a more general diagnosability tool. The detectability tool fits into the following process illustrated in Figure 1.

A system schematic is combined with models of component behaviour and general domain based knowledge to predict the behaviour of the system in the presence of a wide range of component faults in a variety of important system states.

Symptoms are generated in the form of a logical expression involving observations of the system that implicate one or more faults when the expression is satisfied. The algorithm for generating the symptoms uses functional knowledge to allow detailed symptoms to be generated despite incomplete coverage of system states.

The focus of this paper is on a technique to allow the engineer to perform this task, with software tool assistance. The result of the endeavour will be a set of symptoms that can be used directly as a simple symptom based diagnostic system that triggers fault indications when the relevant observations are seen, or alternatively can be used in a more sophisticated Bayesian diagnostic system that includes probabilistic information regarding the reliability of sensors, likelihood of faults, and empirical knowledge of symptoms.

1.1 Why Detectability/Diagnosability is Important

In most systems there are various costs (financial, mass, layout, harness complexity) involved with each sensor, resulting in a need to compromise between diagnostic capability and sensing and therefore a minimal set of the most useful and obtainable observations need to be selected. Due to the complexity of the mapping between sensors, symptoms and faults it is a non trivial task for an engineer to decide on a sensor strategy without tool assistance. Typical issues that require consideration are:

- Which faults are detectable/diagnosable by the sensor strategy?

- Which additional sensors could be included to detect/diagnose additional or critical faults?
- What is the best 'diagnostic value' that can be obtained by adding additional sensors.

Some existing optimisation methods are very specific solutions to an individual system e.g. (Maul *et al.*, 2007; Mushini and Simon, 2005) and do not support schematic and component library based analysis. Other approaches are generic but require significant modelling effort to enable varied additional application specific information to be taken into account (Debouk *et al.*, 1999; Travé-Massuyès *et al.*, 2006). Even when the information required to assess diagnosability can be modelled, the problem has large search spaces and techniques such as genetic algorithms (GA) are often used to find solutions (Spanache *et al.*, 2004; Mushini and Simon, 2005; Maul *et al.*, 2007).

1.2 Why Assistance rather than Automation is the Right Solution

The engineers building the diagnostic system interact with the tool to decide an effective set of sensors for the system. From an academic point of view, this seems like the wrong thing to do. Instinctively, it is felt that the optimal set of sensors can be generated from the model. However, engineer control over sensor selection is necessary for several reasons:

1. Experience shows that in many cases there are simply too many additional application specific considerations that an engineer can resolve, but which would be difficult to provide to a fully automated system. For example, there are spatial constraints associated with adding new sensors for electrical systems, where an engineer may know where it is feasible to add sensors, but without a detailed and 3D spatial model integrated with the electrical circuit description it is impossible for an automated system to decide.
2. A second example is the knowledge of which sensors are required to achieve basic system functionality. Such sensors have a very low cost to any diagnostic system, whereas other sensors are added for diagnostic purposes only and so have a higher diagnostic cost. System engineers will know these things due to their in depth functional and causal understanding of the system architecture, but it is very difficult to extract this information from a system diagram.
3. As a final example, an engineer may know that some parameters are very noisy and should perhaps be avoided (or require additional processing) as inputs to a diagnostic system, for example a fuel level sensor on an aerobatic aircraft.
4. Modelling could be provided for all of the above situations, however the investment in modelling is high for relatively low return, and we take the alternative approach of providing tools that support relatively simple models but allow the engineer to easily make decisions and understand the effects on the potential diagnosability of the system.

Essentially, the system model does not include enough knowledge to generate the best sensor selection solution, and is unlikely to ever have that knowledge. The system described enables the engineer to combine the missing knowledge with an efficient incremental sensor selection method, to evolve an effective solution to the sensor selection problem. Feedback from the sensor selection tool shows the engineer how far they have progressed in ensuring detection coverage for all areas, and indicates the sensors that are most likely to improve fault detectability.

1.3 Structure of the Paper

Section 2 of this paper briefly summarises the way in which model-based reasoning is used to generate symptoms that could detect component failures in a system.

Section 3 explains the graphical matrix representation of detectability in the tool, and how it links sensor readings that can be observed to failures that may occur.

Section 4 gives details of how the tool can calculate for a given set of sensor readings which extra sensor reading(s) to add to improve the failure detectability by the greatest amount.

Section 5 presents a novel way of presenting the present state of the detectability of the system, so that the engineer has the best information on what further sensor readings might be of assistance.

Section 6 extends the work to helping with the task of sensor selection.

Section 7 looks at further work, including the potential of this tool for helping ensure that every potential component failure is individually diagnosable.

2 MODEL BASED GENERATION OF SYMPTOMS

Automated failure mode and effects analysis (FMEA) is a technique that is used to provide a comprehensive and consistent description of the effects of component faults (Price *et al.*, 1997; 2006), and is in use in commercially available electrical deign analysis tools.

More recently, we have extended this research to assist in generating on-board diagnostic systems for unmanned aerial vehicles (Downes, 2007). One of the results from the ASTRAEA work was the ability to generate a set of operating state specific symptoms for monitoring a system on a UAV. This work has recently been protected by BAE systems patent application 1107160.2.

Each symptom is characterized by a tuple of (Ce, Oe, F) where both Ce and Oe are logical expressions and F is a non empty set of component faults that are indicated when the symptom is satisfied.

Ce characterizes the operating state of the system when the symptom is applicable and is termed the *symptom condition expression*. If Ce is false then the symptom is considered invalid and cannot be used. Table 2 gives example symptoms for the UAV case study.

Oe is a set of sensor observations comprising the symptom and is termed the *symptom expression*.

Table 1 shows the possible application of a symptom.

The third row illustrates a 'negatable' symptom able to exonerate faults ($\neg F$) and is the reason for Ce

Table 1: Symptom states

Ce	Oe	Faults indicated
false	false	$\emptyset$ (no fault information)
false	true	$\emptyset$
true	false	$\neg F$ ($\emptyset$ for non negatable symptoms)
true	true	F implicated

expressions. We have observed that allowing negatable symptoms typically leads to fewer symptoms but requires more terms in the expressions than non-negatable symptoms. The ability to exonerate faults when observations are absent is important when the symptoms are used in some forms of on board diagnosis based on for example Bayesian networks.

Based on automotive systems analysed for automated FMEA, we find there are typically hundreds of qualitatively distinct faults (several for each component) and several potential observations associated with each component (Price, 2000). We have found that the number of symptoms generated using our techniques is of the same order as the number of potential component faults, and so for these systems it is feasible to use visual matrices depicting observation-symptom-fault relationships as proposed in the next section.

3 REPRESENTATION OF DETECTABILITY

The mapping from selected observations (switch states and sensor measurements), through available symptoms to observable component faults is illustrated by the matrices in the generic example in Figure 2. This figure is intended only to show the form of the matrices: for a real system there may be hundreds of rows and columns, and it is the visual correlations present in the matrices that provide information to the engineer (zoom/pan is available for larger matrices).

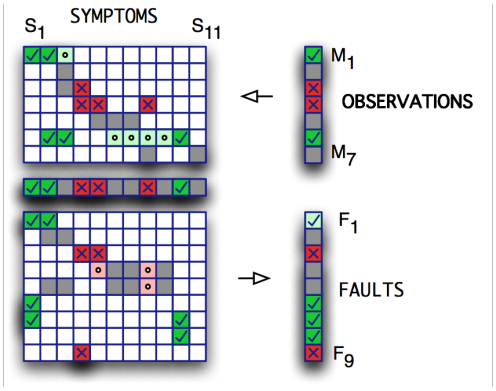

Figure 2: Observation - Fault Matrix

The engineer can affect this matrix by clicking on the Observation boxes, in order to say that the specific observation chosen is available or not available. As they do this, the symptoms also become available or not available, and the component faults become detectable or not detectable.

Meaning of the Observation boxes in Figure 2:

Green Tick: This means that the engineer has chosen that the observation will be available to the diagnostic system.

Red Cross: This means that the engineer has chosen that the observation will not be available to the diagnostic system.

Grey box: This means that the engineer has not yet chosen whether the observation will be available to the diagnostic system.

Meaning of the Fault boxes in Figure 2:

Green Tick: This means that at least one of the set of symptoms that indicate the fault is observable.

Red Cross: This means that none of the set of symptoms that indicate the fault is observable (because the engineer has stated that some of the observations that make up each symptom relevant to this fault are not available).

Grey box: This means that none of the set of symptoms that indicate the fault is yet observable, but at least one set of symptoms has not yet been ruled out by the engineer.

At the start, all of the boxes in the Observation array and the Fault array are grey, as no observations have been included or ruled out. Boxes in the top symptom matrix are grey if an observation is part of a symptom, and white if it is not. So for example, Symptom S1 only includes Observation M1, and so the M1 box in the column for S1 would be grey at the start, and the other boxes would be white. Boxes in the bottom symptom matrix are grey if the presence of that symptom indicates the fault, and white if it does not.

As the engineer includes or excludes observations, the available observations are propagated to infer detectable faults in the following way:

Propagating Green Ticks: When an observation is included by the engineer, then put a pale green dot in the top left matrix for each symptom that has a grey box in the row for that observation. If all observations that comprise a symptom are available, then change all non-white boxes in the column into green ticks (for top and bottom matrix). Where a green tick has been added to the bottom matrix, also put a green tick next to the fault on that row (i.e. that fault is now detectable).

Propagating Red Crosses: When an observation is excluded by the engineer, then put a red cross in the top left matrix for each symptom that has a grey box in the row for that observation. Change all grey boxes in that column in the bottom matrix to pink dots (as that symptom cannot now be detected). Where a pink dot has been added, and all of the grey boxes in the row for that fault are now pink dots, change them to red crosses, and put a red cross next to the fault (because all possible symptoms that could detect it have been explicitly ruled out).

Table 2: Example symptoms

Ce	Oe	F
TVL_RL_LH.position=='isolation'	TVL_FL_LH.tellback=='crossover'	TVL_FL_LH.stuck_crossover
TVL_RL_LH.position=='crossover'∧ CP_FL_LH.control=='on'	OC_WT_RH.tank_level=='higher than expected'	TP4_FL_LH.fracture TP2_FL_LH.fracture TP4_FL_LH.partialblocked
CP_FL_RH.Control=='on'∧ TVL_RL_RH.position=='normal'	FT_FL_RH.flow=='low'	FL1_1_FS_RH.partialblocked TP5_FL_RH.partialblocked

4 RECOMMENDATIONS FOR OBSERVATION SELECTION

There are usually a set of observations that will definitely be available to the diagnostic system, such as changes of operating state. There are also observations that that the engineer knows will be important in the diagnosis of a required set of faults. These observations can be selected, and some faults will then be marked as observable, using the process described in the previous section. At some point, the question will arise of how to choose the next set of observations that diagnose the maximum number of faults.

The problem of finding n additional observations that allow the maximum number of faults to be detected is exponential in the number of additional observations if a brute force search is carried out. Due to the localisation of observation - fault relationships, it is only useful to use small numbers for n, until a new 'block' of elements (observations, symptoms and faults) is identified.

For an exhaustive search, if n is the number of additional observations required and r is the number of unselected observations remaining, there are $\frac{r!}{(n*(r!-n))}$ combinations of observations to consider.

In the early stages, systems tend to have a few critical observations that provide big diagnostic returns and so a relatively small n is adequate to find these, and once a good number of the observations are determined, r becomes small allowing larger n in reasonable time. However, by this stage, symptoms and faults tend to be closely coupled, and so adding an observation only covers a few additional faults, and therefore the next best n observations provides a superset of the faults that can be obtained by the next best $n-1$ observations.

Our experience is that a good strategy is to consider a small combination of extra observations and then investigate why there are alternatives, make a selection (noting any significant effects on the matrices), and then consider subsequent observations associated with the next region of system structure and behaviour.

For a partially selected set of observations for the fuel system of a UAV, Figure 3 shows the next possible pairs of observations that are most effective, along with the faults that those observations monitor. These can be speculatively selected, and their potential effect on the matrix representation will be shown in yellow rather than green, as illustrated in Figure 4. If the engineer is happy with that effect, he can include the observations in the set of accepted observations, and they will be propagated in green as normal.

5 FOCUSING ON THE DETECTABILITY

To gain a much better understanding of the relationships contained within either matrix they can be automatically reformed into an 'approximate diagonal form' which places all the non empty matrix elements as close to an imaginary line from top-left to bottom-right as possible (this is the purpose of the "Order" buttons on the tool interface in Figure 4). The algorithm used is similar to the well known bubble sort applied alternately to row and columns, with the ordering comparison based on the imbalance of the number of non zero cells from the diagonal. Since the matrices are not generally square a true diagonal matrix in the mathematical sense is not possible.

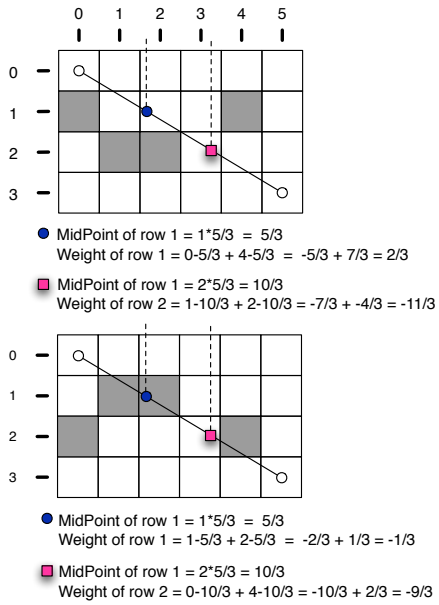

Figure 5: Producing the diagonal matrix

The concept of a row (or column) weight is used to describe the number of cells in either a row or column to either side of the imaginary diagonal line across the matrix. Figure 5 shows an example 6 by 4 matrix. The mid point of rows 1 and 2 are shown by the filled symbols. The weight of each row is calculated as the sum of the distance (as a cell count) of each active cell (shown grey in Figure 5) from the mid point. In the upper matrix of the example row 1 has a weight of $\frac{2}{3}$ and row 2 has a weight of $-\frac{11}{3}$. By extension, the columns can be similarly considered. If the imbalance of two

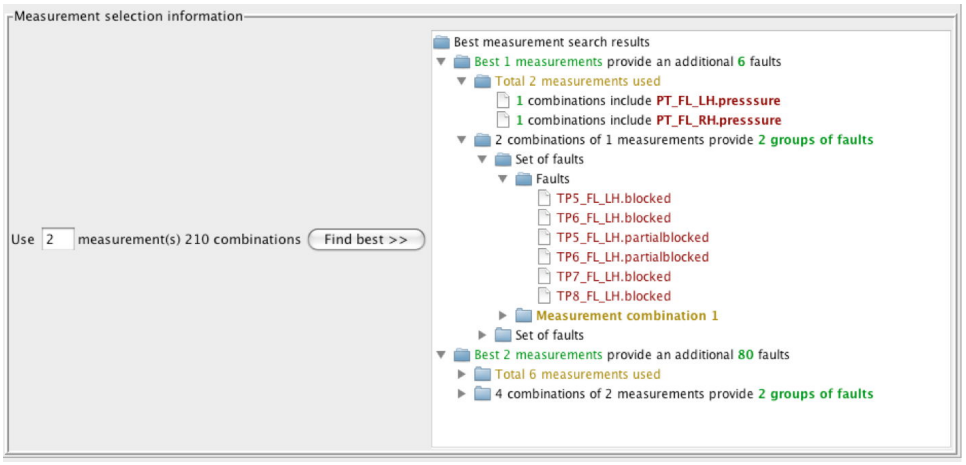

Figure 3: Fuel System - Result of search for two additional observations

rows is defined as the weight of row n−the weight of row $n+1$, then the rows are swapped if the imbalance is greater than zero unless the result of swapping the rows creates a larger imbalance for the rows. In the example the imbalance is $\frac{2}{3} - (-\frac{11}{3}) = \frac{13}{3}$. This is greater than zero and therefore the rows are swapped to produce the matrix shown in the lower part of Figure 5, in which the imbalance is $-\frac{1}{3} - (-\frac{9}{3}) = \frac{8}{3}$. Since $\frac{8}{3}$ is less than $\frac{11}{3}$ the reordered matrix is considered closer to diagonal than the original and the swap is retained. A similar procedure is then carried out between rows 2 and 3, and so on. The overall effect of swaps is to reorder the lists of observations, symptoms, and faults. Each pair of rows are repeatedly considered in the manner of a bubble sort, using the weight measure as the ordering criterion. However, in contrast to a standard sort, the weight of a row changes (and is therefore recalculated) when it is moved. The sort is undertaken alternately on rows and columns.

Once each pair of row and column sorts is completed, the total imbalance of the entire matrix is calculated as the imbalance sum of all rows plus all columns. The alternate sorting of rows and columns continues until no further reduction in the total matrix imbalance can be achieved. Once the chosen matrix is in diagonal form, the unshared axis of the other matrix is sorted to make it as diagonal as possible. At this point the majority of the weight of the matrix is balanced around the diagonal as closely as possible. This has the effect of bringing related observations and symptoms (or symptoms and faults) together on the diagonal and allows the user/engineer further insight to the diagnostic capability of the system by producing visual blocks of colour representing the relationship between groups of observations, symptoms and faults. Disjoint blocks also graphically illustrate parts of the system that are diagnostically separate, for example sets of symptoms and observations that are the only possibility for diagnosing a set of faults for some part of a system.

Each row or column sort is effectively a bubble sort with a worst and average $O(n^2)$ complexity where n is the number of observations, or symptoms, or faults dependent on which dimension is being sorted.

However the matrices have two characteristics that in practice seem to make the average complexity of the whole algorithm not much worse than this. Firstly the matrices are rather sparse and secondly there is a strong relationship between groups of elements on each axis. For example, we find (and would expect) a related set of faults that can be diagnosed by a related set of symptoms using a related set of observations. The algorithm will only need a single sort on one dimension for a matrix that has a perfect simple diagonal form since the order of one axis can be arbitrary and the elements moved onto the diagonal by reordering the other. The more 'imperfect' the final diagonal matrix in the sense of the number of empty elements between the diagonal and any non zero element in the result, the more iterations of the row and column sort sequence could be needed. This is because the solution may require (worst case) a specific ordering of each axis. The sparseness of the matrices combined with the systematic effects of faults and the structure of the system cause the matrices to have a good 'compact' diagonal form, and in fact they will only be useful if this is the case. Therefore only a small number of iterations of the sorting should be required, and this has been observed experimentally. We also observe that the algorithm is converging towards the solution and therefore once the first sort is completed on each axis, subsequent sorts start with most of the elements already in the correct order. The visual effect is that non empty elements 'bubble' along the diagonal until each group of elements has achieved its best order on the diagonal.

In Figure 6 the remaining elements have been diagonalised on the fault symptom matrix and groups of related faults are clearly seen, each block tends to be related to a different system function, due to structural locality. Hovering the mouse over each block and looking at the symptom conditions easily reveals the states of the system involved, for example the block under the mouse pointer is related to the sidelights and the yellow (light coloured) selected symptoms are all

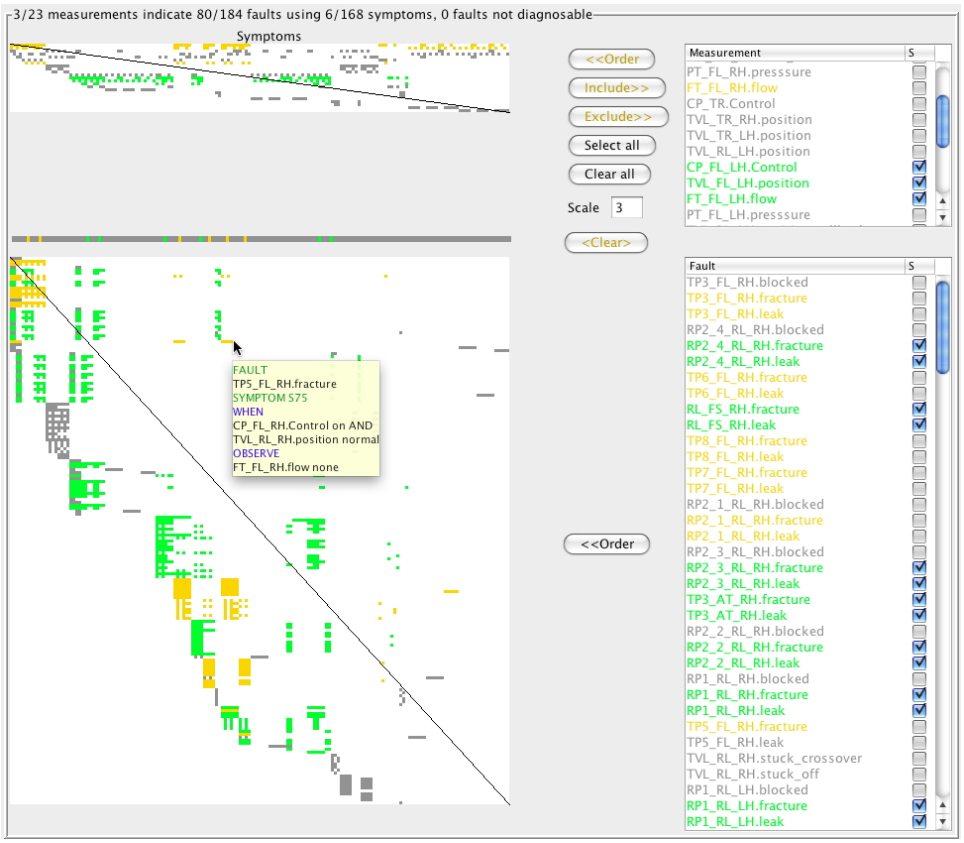

Figure 4: Aircraft fuel system matrix example

related to the dip lights.

The aim is to assist in the selection or rejection of observations, and therefore any elements that are already resolved (red or green) are NOT included in the process and are moved to the bottom or right of the matrix and do not participate in the sorting. This is why the diagonal line does not extend the full size of the center left matrix in Figure 6. It is useful to repeatedly make the matrices diagonal as an interactive activity during the measurement selection process as diagnostic characteristics are discovered.

6 EXTENSION TO COVER SENSOR PLACEMENT

The aircraft fuel system example in the previous sections of this paper had a predefined set of sensors and observable settings. For other systems the task may be to determine which sensors to add to build a diagnostic system. We concentrate on sensors that measure system parameters within the domain of the simulation, so for example in an electrical network, rising temperatures as a fault symptom could not be produced as a symptom unless the simulation were to include a thermal model. For systems that include diagnosis specific sensors (e.g. vibration sensors) from other domains, hand crafted or externally generated symptoms can easily be added to the symptom set and included in the overall detectability analysis, if required.

It is easy to allow the diagnostic generator to have access to any system (simulation) parameter, and as an example we present an automotive daylight running lights system (DTRL) allowing the current in every wire in the system as a possible sensor input. Perhaps unsurprisingly, many symptoms are generated based on the function output observations (lamps) and the inputs that are the triggers for the functionality that will cause activity at the observation point. The matrices show which observations are diagnostically equivalent for various sets of faults, for example the vertical 'stripe' pattern in the Figure 7 fault-symptom matrix. Figure 7 also demonstrates critical input as a long horizontal bar in the center of the measurement matrix (lighting switch position), without which most faults cannot be diagnosed. The bar is (green) light coloured because it is clear it must be selected for the majority of the symptoms to be usable.

The lower right of the Figure also demonstrates a situation where three equivalent alternative observations may be used. The number plate lamps have been excluded because they are not directly observable by a sensor, leaving a choice between W16 and W27. W27 was chosen and this makes 6 symptoms redundant (red), although there is no effect on the number of faults that can be diagnosed.

Following the process until all faults are accounted for results in the statistics in Table 3. Many systems exhibit this law of diminishing returns as each extra sensor pinpoints fewer faults.

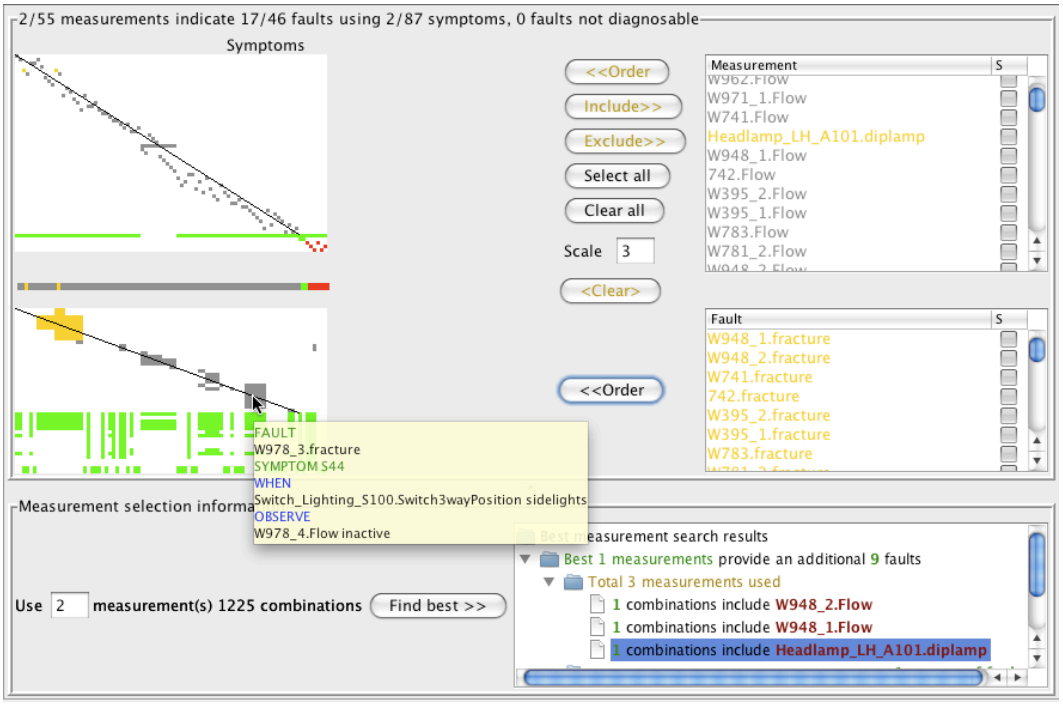

Figure 6: Diagonalisation Example

Table 3: DTRL sensor selection

Observations Selected (55 total)	Faults Detectable (46 total)	Symptoms Used (87 total)
2	17	2
3	19	4
4	28	6
5	35	8
6	38	10
8	42	11
9	43	13
10	44	15
11	45	16
12	46	18

7 FURTHER WORK AND CONCLUSIONS

7.1 Documentation of Engineering Decisions

The detectability tool described here works with engineers enabling them to enhance its recommendations with their knowledge of the characteristics of the system. They can decide to exclude observations because they know that in practice those specific observations are not reliable. They can include extra sensors because they are the fastest or most efficient way of detecting problems. At the moment, such decisions are only recorded as an observation being ”in” or ”out”. It would be good to have the option of recording the rationale behind such decisions, at least as an option for the users.

7.2 Diagnosability

Presently, many symptoms implicate a number of component faults. For example, in the fuel system of the UAV that we studied, a loss of flow in one of the fuel lines under normal running might imply a possible leak in a number of pipes or tanks, or a problem with one of several pumps. It is precisely this quality that makes the detectability tool so effective. Selecting the best two or three extra observations often covers a significant number of faults for precisely this reason.

It may be possible to adapt the tool from ensuring that all faults can be detected, and instead consider whether all significant faults can be uniquely identified from the available sensors. The focus of such work is likely to be either making decisions about continued safe operation, or repair. If the focus was safe continued operation, then for the fuel system it would be necessary to partition component faults into ones that would demand an engine shutdown, and ones that could be compensated. If the focus was repair, then the fuel system would need to be partitioned by least replaceable unit (LRU), and the system could search for sensor combinations that uniquely distinguish between LRUs.

In either case, it should be possible to adapt the work described here to give appropriate support to engineers concerned with guaranteeing diagnosability.

7.3 Conclusions

The matrix visualisation technique described in this paper provides engineers with valuable feedback on the extent to which all possible component faults in a system are detectable by a specific set of observations. A detectability tool based on this technique works in-

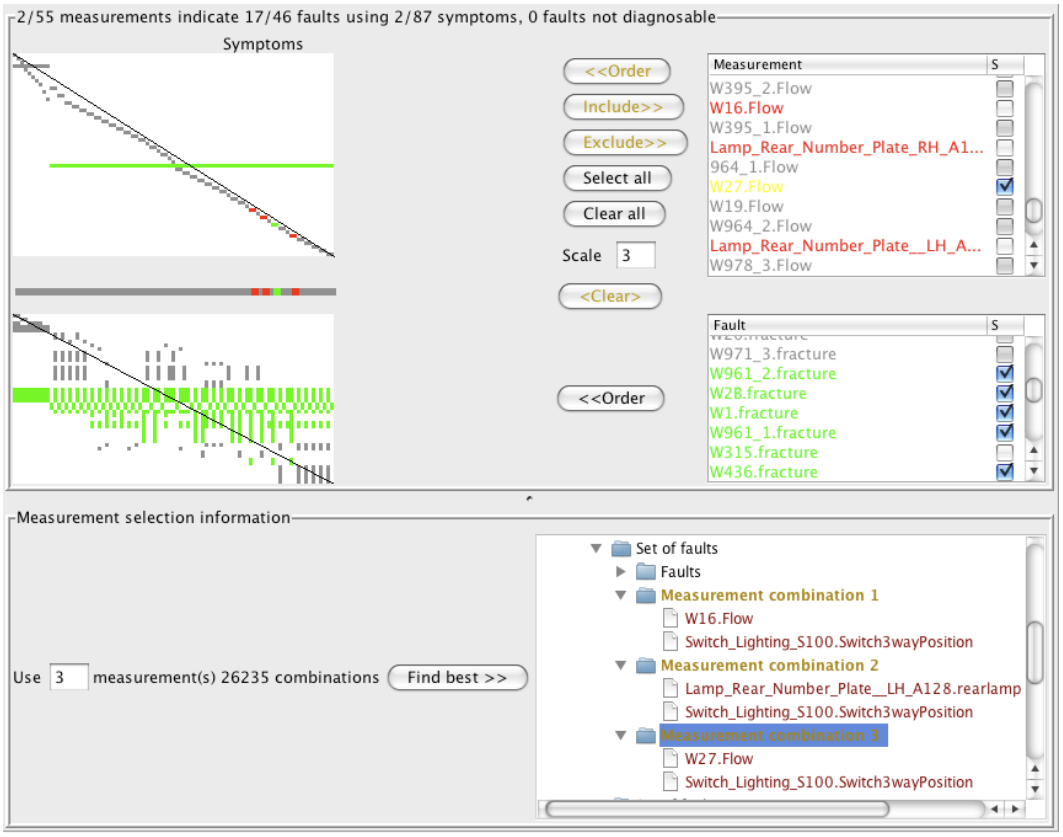

Figure 7: Instrumented DTRL system

teractively with the engineer to select a set of observations that cover all fault possibilities. By assuming that all values in the system might be observable, the tool can also be used by the engineer to decide on a sensor strategy for the system.

ACKNOWLEDGEMENTS

Aberystwyth University's work on the ASTRAEA project was funded by the Welsh Assembly Government, by BAE Systems and by Flight Refuelling Limited. The ASTRAEA project was co-funded by the Technology Strategy Board's Collaborative Research and Development programme, following an open competition. This work is protected by BAE Systems patent applications (0910145.2 and 1107160.2).

REFERENCES

(Debouk *et al.*, 1999) R. Debouk, S. Lafortune, and D. Teneketzis. On an optimization problem in sensor selection for failure diagnosis. In *Procs 38th IEEE Conf. on Decision and Control*, pages 4990–4995. U. Mich, 1999.

(Downes, 2007) C. Downes. Astraea T7: an architectural outline for system health management on civil UAVs. In *Procs 2nd Autonomous Systems Conference, IET, November*, 2007.

(Maul *et al.*, 2007) W. A. Maul, G. Kopasakis, L. M. Santi, T. S. Sowers, and A. Chicatelli. Sensor selection and optimization for health assessment of aerospace systems. Technical Report NASA/TM—2007-214822, http://gltrs.grc.nasa.gov/, 2007.

(Mushini and Simon, 2005) R. Mushini and D. Simon. On optimization of sensor selection for aircraft gas turbine engines. In *18th Int. Conf. on Systems Engineering*, pages 9–14. ISBN: 0-7695-2359-5, August 2005.

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

(Price *et al.*, 2006) C. J. Price, N. A. Snooke, and S. D. Lewis. A layered approach to automated electrical safety analysis in automotive environments. *Computers in Industry*, 57(5):451–461, 2006.

(Price, 2000) C. J. Price. AutoSteve: automated electrical design analysis. In *Procs ECAI-2000*, pages 721–725, 2000.

(Spanache *et al.*, 2004) S. Spanache, T. Escobet, and L. Travé-Massuyès. Sensor placement optimisation using genetic algorithms. In *Procs DX04*, pages 179–183, 2004.

(Travé-Massuyès *et al.*, 2006) L. Travé-Massuyès, T. Escobet, and X. Olive. Diagnosability analysis based on component-supported analytical redundancy relations. *IEEE SMC Part A*, 36(6):1146–1160, 2006.