Automated Failure Effect Analysis for PHM of UAV

Neal Snooke and Chris Price — Department of Computer Science, Aberystwyth University, nns{cjp}@aber.ac.uk
Clive Downes — BAE SYSTEMS, Autonomous Systems & Future Capability(Air), clive.downes@baesystems.com
Carole Aspey — Cobham's Flight Refuelling Ltd., Wimborne, carole.aspey@cobham.com

Keywords: FMEA, diagnostics, safety case

Abstract

The ASTRAEA project is a £32M UK initiative to develop the safety case for unmanned aerial vehicles flying in commercial airspace. It is addressing both the issue of what needs to be covered by such a safety case, and how such a safety case can be constructed efficiently. One of the key areas within the remit of ASTRAEA is that of generating diagnostics capable of correctly identifying the causes of all possible failures of the vehicle.

This paper describes how model-based simulation can be employed to automatically generate the system-level effects of all possible failures on systems within the aircraft. The results of the simulation can be used in several ways. They can be used to produce a system-level FMEA for aircraft systems. They can be used to identify the sensors necessary to discriminate remotely between different failures on the aircraft. Once a set of sensors have been chosen for placement on the vehicle, the simulation results can also be used to generate diagnostic and prognostic software for deployment on the vehicle. Using the automated safety analysis software developed on the ASTRAEA project is more efficient than doing the same work without the software, and also provides a guaranteed level of performance.

1. Introduction

This paper describes the use of qualitative models of aircraft systems to automate the generation of diagnostics and prognostics for unmanned aerial vehicles (UAVs). This work is part of a much wider initiative known as ASTRAEA, to develop the technologies needed to safely operate UAVs in civilian airspace.

ASTRAEA is a £32 million initiative to research and prove the necessary technologies to enable the safe, routine use of UAVs. It is funded by the UK Department of Trade and Industry, by several of the UK regional development agencies, and by major aerospace companies such as BAE Systems, Thales UK, Rolls Royce, EADS, QinetiQ and Flight Refuelling Limited. Successful development of relevant technologies, and the ability to make convincing safety cases for the operation of UAVs in civilian airspace would be of great benefit, opening up potential uses for UAVs in logistics, environmental monitoring, and many other areas.

ASTRAEA covers eight different areas of technology needed for routine deployment of UAVs:

- Ground Operations and Human Interaction.
- Communications & Air Traffic Control.
- UAV Handling.
- Routing.
- Collision Avoidance.
- Multiple Air Vehicle Integration.
- Prognostics & Health Management.
- Decision Modelling.

The Prognostics and Health Management (PHM) area is responsible for developing the technology and systems needed to enable UAVs to monitor their own state. Based on that monitoring, the PHM systems need to make assessments of mission readiness that can be utilised by higher level planning in the Decision Modelling area. The major goal is that the PHM systems should be capable of replicating the fault detection, assessment, and decision making ability of human pilots.

The PHM work carried out by Aberystwyth University within ASTRAEA concerns the use of computer models of aircraft systems to reason about the potential causes of any possible failure of representative aircraft systems, and to decide what aspects of the system need to be monitored in order to be sure that any serious operational discrepancies will be detected.

This paper describes the kind of modelling technology being used in this work, and shows how it can be used for failure analysis, for diagnosability analysis, and for production of online diagnostics and prognostics.

2. Underpinning research in model-based reasoning

The Advanced Reasoning Group at Aberystwyth University has been developing model-based systems capable of performing failure modes and effects analysis (FMEA) for more than a decade [1,2]. Earlier versions of this work resulted in AutoSteve [3], a commercial tool for FMEA of automotive electrical systems that has been taken up by a major electrical computer aided design tool vendor.

The main ideas of the work are that a description of the overall behaviour of a system can be constructed by knowing the structure of a system and the behaviour of each of its components. In order to determine the behaviour of the system when a failure occurs, it is only necessary to replace the component which has failed with a version of the component that reproduces the faulty behaviour, and the model of the system will then reproduce the effects of the failure for the whole system.

Such analysis is often best done at the qualitative level [4] rather than through detailed numerical simulation, as the precise data needed for the numerical simulation is not available. Hence, techniques such as Monte Carlo methods need to be used to give the coverage of possible outcomes that naturally fall from a qualitative simulation.

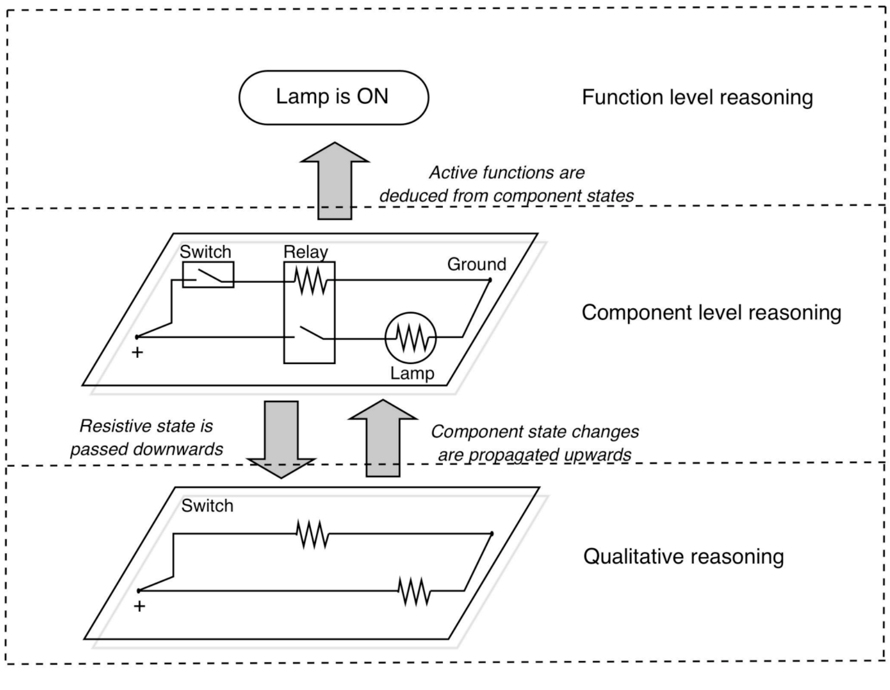

Lamp is ON
Function level reasoning
Active functions are deduced from component states
Switch
Relay
Ground
Lamp
+
Component level reasoning
Resistive state is passed downwards
Component state changes are propagated upwards
Switch
+
Qualitative reasoning

Figure 1: Levels of electrical simulation

Figure 1 shows the different levels of reasoning needed to perform model-based simulation of an electrical system. The system description is usually provided at the component level - the system is described in terms of its components and their states and connections. This can be mapped onto the resistive state of all components, and the electrical state of the overall system can be determined qualitatively. This may result in changes in component state (e.g. if current is now flowing through a relay, then it may close, changing the state of the relay component, necessitating further qualitative simulation. Eventually, the state of the system will either stabilise (once relays have opened or closed, and devices have been powered) or oscillate. The results of this interchange between the component level reasoning and the qualitative electrical reasoning can be abstracted to the functional level, and returned in terms of which functions of the system have occurred.

Figure 1 illustrates the relationship between the three levels for a very simple system, but these techniques work for electrical/electronic systems with several thousand components, and produce useful FMEA results that pinpoint significant potential failures early in the design process.

3. Modelling more complex systems

Within ASTRAEA, the fuel system has been identified as an important aspect of UAV operation, and demands more advanced reasoning than the electrical systems that could be dealt with by the previous AutoSteve software. It includes all of the electrical aspects that AutoSteve could deal with, but also requires the capability to reason about fluid flow.

It is only the lowest level of figure 1 that needs to be extended to deal with the more complex simulation demanded by mixed electrical/fluid systems. Previously, the qualitative reasoning was based on just three levels of resistance (zero, load, infinite), and this has been extended to enable reasoning about different levels of resistance (e.g. low, medium, high) [5].

Despite the analogies between current flow and fluid flow, further extensions to the original qualitative reasoning are needed in order to be able to simulate a fuel system effectively. The original qualitative reasoning assumed a single power source (the car battery). As multiple pump configurations are common in fluid systems, the reasoning was extended to be able to deal with multiple "power" sources within a network. Also, in order to be able to reason about the effect of leaks that allow the ingress of air or escape of the fluid, it was necessary to have an explicit representation of the substance being propagated through the system. The qualitative reasoning has been extended to deal with these needs [6], and is now capable of reasoning about mixed electrical/fluid systems.

On the ASTRAEA project, this technology is being used to simulate the electrical and fluid components of a complex fuel system, with multiple tanks, and load balancing in supplying a twin engine aircraft. The techniques discussed in this paper scale up to such realistic systems, but for commercial and clarity reasons, the system analysis that can be automated using the simulation will be illustrated in this paper with a simpler example system as depicted in figure 2. The simpler system essentially connects a fuel tank and engine with a pump, two valves, pressure and flow sensors.

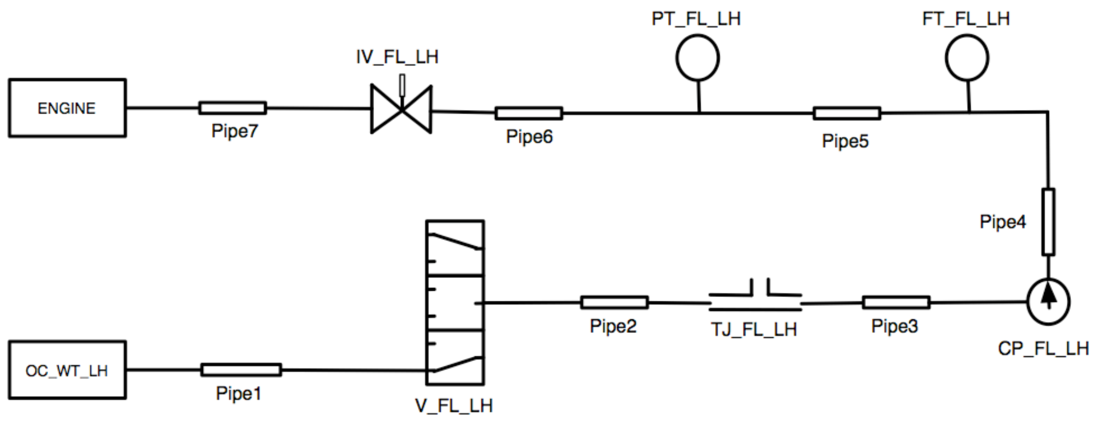


Figure 2: Simple example system

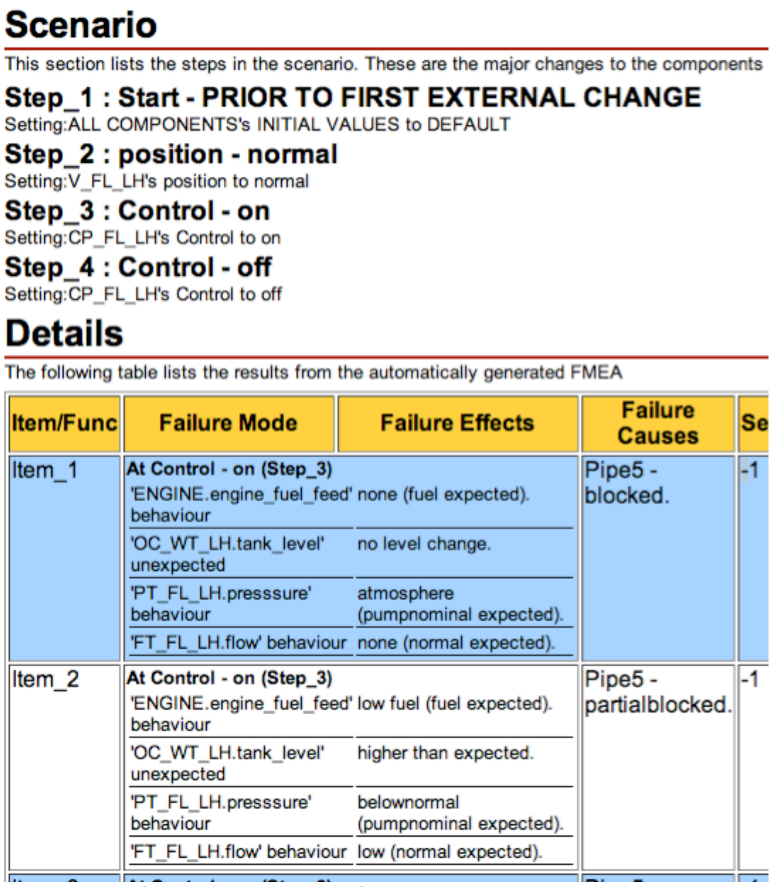
4. Generating FMEA results

An FMEA report can be automatically generated by comparing the results from a simulation of nominal system behaviour with the results of a simulation of a version of the system for each possible component fault. The differences between the two simulations comprise the effect of the component failure (e.g. a function occurs at some point in the simulation of nominal system behaviour, but fails to occur in the same state when there is a specific component failure). More details of this technique are given in [2].

The system needs to be simulated in its different operating modes in order to identify the possible effects of a failure. That means that it is necessary to decide how the system should be exercised during simulation in order to cover all operating modes. Normally an engineer will provide a set of inputs that cause the system to enter all of its operating modes and configurations. For example, in the full aircraft fuel system, we may change valve positions to encounter normal fuel feed, fuel transfer operations, and fuel cross feed configurations. The set of inputs chosen is referred to as the scenario. On occasion an engineer may decide to only deal with a subset of system operation, and this will limit the effects that will be observed to those observable during the selected operations.

The failure behaviours for each component are contained in a library of component models, and usually represent a modification of the nominal component structure or behaviour model. For example a blocked pipe will simply change the resistance of the pipe to ‘infinite’. Of course these faults are created for each *type* of component and are then automatically inserted for each instance of the component present in a system. Once a component library exists, most of the components and failure modes are immediately available because minor parametric changes to components are insignificant to a qualitative analysis.

The FMEA generation system operates in the following way. The system is simulated for the chosen scenario with no failures present, and all observable values are recorded for each step of the simulation. The system model is reconfigured for each component failure. Each step of a failure simulation is compared with the non failure simulation and the differences are recorded.



Scenario

This section lists the steps in the scenario. These are the major changes to the components

Step_1 : Start - PRIOR TO FIRST EXTERNAL CHANGE
Setting:ALL COMPONENTS's INITIAL VALUES to DEFAULT

Step_2 : position - normal
Setting:V_FL_LH's position to normal

Step_3 : Control - on
Setting:CP_FL_LH's Control to on

Step_4 : Control - off
Setting:CP_FL_LH's Control to off

Details

The following table lists the results from the automatically generated FMEA

Item/Func	Failure Mode	Failure Effects	Failure Causes	Se
Item_1	**At Control - on (Step_3)**		Pipe5 - blocked.	-1
	'ENGINE.engine_fuel_feed' behaviour	none (fuel expected).		
	'OC_WT_LH.tank_level' unexpected	no level change.		
	'PT_FL_LH.presssure' behaviour	atmosphere (pumpnominal expected).		
	'FT_FL_LH.flow' behaviour	none (normal expected).		
Item_2	**At Control - on (Step_3)**		Pipe5 - partialblocked.	-1
	'ENGINE.engine_fuel_feed' behaviour	low fuel (fuel expected).		
	'OC_WT_LH.tank_level' unexpected	higher than expected.		
	'PT_FL_LH.presssure' behaviour	belownormal (pumpnominal expected).		
	'FT_FL_LH.flow' behaviour	low (normal expected).		

Figure 3: Fragment from a generated FMEA

The automatically generated FMEA provides a description of the observable effects of each fault. These effects are consistent, providing all potential effects in qualitative terms. Figure 3 shows a fragment of an FMEA generated from the simple example system shown in figure 2, comprising a fuel tank, pump, pressure and flow transducer feeding an engine. Due to the qualitative nature of the analysis, the results are naturally presented as qualitative differences (such as value *higher than expected*).

The FMEA results can ordered by significance, and would be inspected by the engineers to identify how significant failures can be mitigated, either by decreasing the effect of the failure (e.g. by providing backup systems) or improving the detection of the failure (e.g. by adding sensors) or reducing the occurrence of the failure (e.g. by using more robust components) [7].

The FMEA report provides a consistent report that covers all component failures of specific types, and this consistency and coverage provides a basis for generating diagnostics that will be explored in the next section.

5. Generating on-board diagnostics for the UAV

Diagnostics on the UAV are controlled by an online diagnostic system being developed by BAE Systems. The basis of the system is a set of fault-symptom pairs with weighted links between each symptom and each fault.

For convenience the symptoms are categorised into three types:

- **Single value** for example, *pressure transducer value = out of range low.* In this situation a symptom is a simple value that does not occur during normal operation and directly indicates one or more faults.
- **Multiple value** for example, *level = empty AND low level switch = on*. Two values may be linked to indicate an inconsistency in system operation caused by a fault. Typically, this is because two values measure physically related properties and should be consistent in the absence of a fault.
- **Conditional** *IF (pump = on), pressure monitor = high* or *IF (Valve SOV4 commanded OPEN), Open Sensor is not responding OPEN.* Conditional symptoms may be used when a value is only indicative of a fault in specific operating modes or configuration. For example a low pressure reading might only be significant when a pumping operation in progress. If no pumping is carried out then the symptom does not apply (pressure will be expected to be low). The condition specifies if the observation is valid. For the diagnostic net (discussed in following paragraph) the distinction between multiple value and conditional symptoms is important because the symptom observations are not entered to the net for an invalid symptom.

The FMEA report provides differences of observable values, e.g. when PIPE5 is blocked, PT_FL_LH pressure is below normal when normal was expected. This type of information can be used to produce the kind of fault-symptom information needed by the online diagnostic system. The symptoms should be as simple as possible while still providing a definite indication of the associated fault.

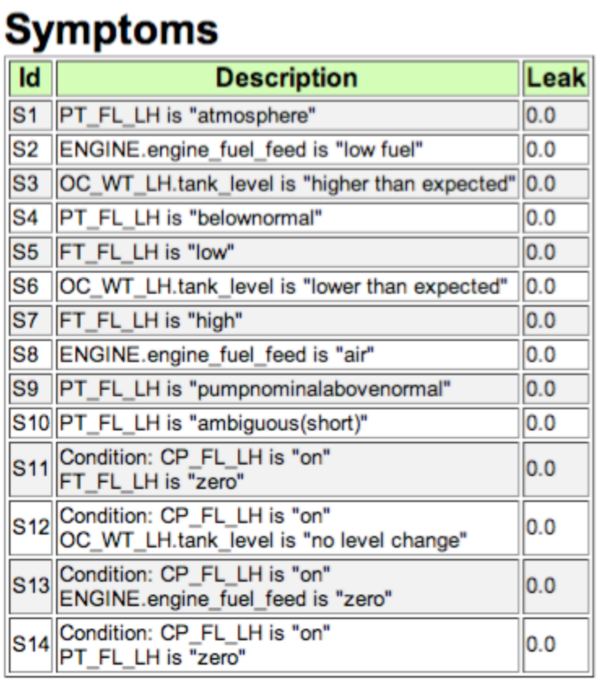
The FMEA produces output in the form of the difference between nominal and failure operation at each step of a scenario, exercising the system. An example may be `After switch X closed and then switch Y closed, Obs 1=L when Obs 1=H expected'. The FMEA information does not directly provide a symptom because it is usually not known explicitly what is expected. The symptoms must therefore be based only on identification of abnormal observations. The FMEA item above is used to produce a symptom of the form {(X, closed), (Y, closed), (Obs 1, L)}.

The symptoms are produced from the FMEA information using the following steps:

- **Collect symptoms.** All observable sets of values are collected for each observable state (scenario step) in the simulation. Potential symptoms are generated as observation sets that that do not appear in the nominal operation. Initially single value, multiple value, and conditional symptoms were generated separately. This is unnecessary however since the simpler symptoms are located as part of the simplification steps below. The observables that are used are; outputs, inputs (exogenous variables that are only read), and scenario steps (operating configurations).

- **Shorten symptoms.** To determine which observations are actually relevant to a symptom, each observation is removed in turn from the symptom and it is retained if it still represents an abnormal set of observations. Shorter symptoms are added to the symptom list. The process is applied repeated for all successfully generated shorted symptoms, until single observation symptoms are found, or observations no longer form a valid symptom.
- **Simplify Symptoms.** Symptoms with more terms than a shorter one *and* cannot distinguish more faults are removed. A shorter symptom must include a superset (or the same set) of the faults indicated by the longer symptom to allow the longer symptom to be removed from consideration.

Single and multiple value symptoms are produced when sets of outputs are found to produce a fault indication. Conditional symptoms are generated when inputs and/or operating configurations are required to form a symptom. Figure 4 shows an example set of symptoms generated from the simple example system shown in figure 2. Most of these symptoms are single abnormal values, however notice that several conditional symptoms have been generated (S11-S14) to allow values that occur when the system is inactive to become symptoms when the operating conditions change. For more complex systems, the combinations of values and conditions required becomes greater and provides symptoms that detect unusual effects, and is particularly useful when a system has many operating modes with symptoms specific to each.

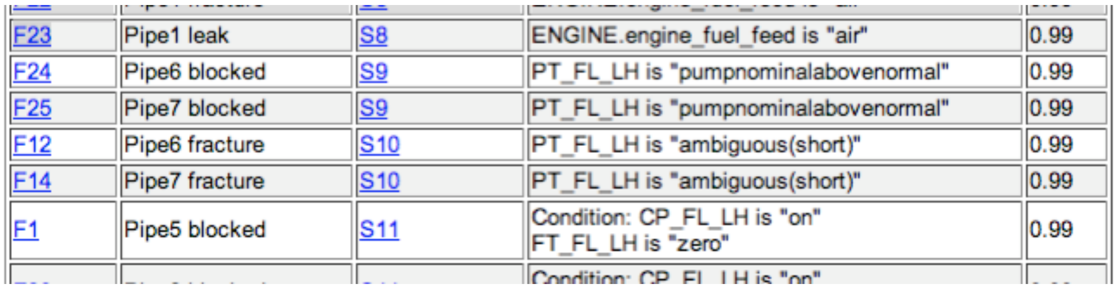


Symptoms

Id	Description	Leak
S1	PT_FL_LH is "atmosphere"	0.0
S2	ENGINE.engine_fuel_feed is "low fuel"	0.0
S3	OC_WT_LH.tank_level is "higher than expected"	0.0
S4	PT_FL_LH is "belownormal"	0.0
S5	FT_FL_LH is "low"	0.0
S6	OC_WT_LH.tank_level is "lower than expected"	0.0
S7	FT_FL_LH is "high"	0.0
S8	ENGINE.engine_fuel_feed is "air"	0.0
S9	PT_FL_LH is "pumpnominalabovenormal"	0.0
S10	PT_FL_LH is "ambiguous(short)"	0.0
S11	Condition: CP_FL_LH is "on" FT_FL_LH is "zero"	0.0
S12	Condition: CP_FL_LH is "on" OC_WT_LH.tank_level is "no level change"	0.0
S13	Condition: CP_FL_LH is "on" ENGINE.engine_fuel_feed is "zero"	0.0
S14	Condition: CP_FL_LH is "on" PT_FL_LH is "zero"	0.0

Figure 4: Generated symptoms example

Once the set of symptoms has been generated, the FMEA is used to produce a matrix of fault symptom pairs. An example fragment is shown in figure 5, showing Fault number, fault name, symptom number, symptom name, confidence value.



F23	Pipe1 leak	S8	ENGINE.engine_fuel_feed is "air"	0.99
F24	Pipe6 blocked	S9	PT_FL_LH is "pumpnominalabovenormal"	0.99
F25	Pipe7 blocked	S9	PT_FL_LH is "pumpnominalabovenormal"	0.99
F12	Pipe6 fracture	S10	PT_FL_LH is "ambiguous(short)"	0.99
F14	Pipe7 fracture	S10	PT_FL_LH is "ambiguous(short)"	0.99
F1	Pipe5 blocked	S11	Condition: CP_FL_LH is "on" FT_FL_LH is "zero"	0.99

Figure 5: Fault symptom matrix

The fault-symptom pairs are qualitative, but it is necessary for the online monitoring system to decide the value at which a specific observation triggers a symptom. For example, at what level is a pressure value to be considered 'low'? Some of the symptoms generated by the FMEA may need additional computation as part of the signal conditioning process. For example the observation 'tank fuel level lower than expected' may require prediction of the expected level based on engine throttle demand over a period of time these measures may be considered as virtual sensors that provide computed values. Both of these issues require very detailed knowledge of the system, the sensor characteristics, and may even need to be determined empirically or calibrated for each specific instance of a system.

Other than thresholds and virtual sensors, three basic numerical measures are used by the network:

- **Symptom leak** - probability that the symptom will be observed even though there are no faults. By default this is set to 0 for the auto FMEA symptoms to indicate the symptom will not be observed unless a relevant fault exists. It may be decided that certain observations are more likely to be spurious than others due too sensor or system characteristics.
- **Prior** - probability that the fault has occurred prior to any symptoms. This is the fault occurrence number in FMEA (or 0.01default)
- **Fault** - Probability of the fault, given the symptom. By default this is set to 0.99 for the auto FMEA symptoms to indicate the symptom will be observed if the fault exists. Other values may be used for some symptom or fault categories because the effects predicted by the FMEA are less or more significant than can be determined qualitatively. For example leaks may not always produce a *measurable* pressure drop, even though theoretically any leak would produce some drop in pressure.

When one or more symptoms are observed, these values are propagated through the weighted network to produce an ordered list of possible faults. A confidence value is associated with each predicted fault allowing decisions to be made within the higher level control elements of the system.

6. Diagnosability analysis

Ideally, any system fault could be diagnosed to the individual component that has failed, however, during design, there is a trade-off between the amount of sensing possible for a system, and the diagnostic capability. It is useful to be able to investigate the relationship between the number and placement of sensors and the resultant ability to detect faults and subsequently isolate faults to a specific component or line replaceable unit (LRU). One of the benefits of model-based simulation is that we can easily gain access to many system parameters and analyse which might provide the required diagnosability at a given cost. Alternatively we might wish to analyse how many sensors would be required to be able to provide a given level of diagnosis. On the ASTRAEA project, we have performed two types of analysis, the first provides an ordered list of sensors that can provide the maximum number of diagnosed faults. The second provides an ordered list that prioritised by the ability to isolate a fault.

Symptom ranking is carried out using a recursive procedure starting with no included symptoms or detected faults, and consisting of the following steps:

- From the remaining symptoms that have not been considered, find the symptom(s) that provide the maximum number of additional faults. i.e. the number of faults detected by the symptom that are not already located by previously considered symptoms. These are termed the 'next best symptoms'
- For each of the 'next best symptoms' add it to the considered symptom list, add any failures it detects to the faults detected list.
- If the symptom has not already been found and included in the overall results, then include it in the results in the correct place (according to the number of symptoms), and carry out the procedure again excluding the current symptom from the available symptoms.

This will generate sets of symptoms for each number of observations that can diagnose the most faults. Where several symptoms each diagnose the same number of faults, they are all outputted. For the example circuit in figure 1, we get the result:

```
ORDERED SYMPTOM INFORMATION

1 combinations of 1 SYMPTOMS  indicate 11 FAILURES (1 partitions)
1 combinations of 2 SYMPTOMS  indicate 18 FAILURES (2 partitions)
1 combinations of 3 SYMPTOMS  indicate 24 FAILURES (3 partitions)
1 combinations of 4 SYMPTOMS  indicate 28 FAILURES (5 partitions)
8 combinations of 5 SYMPTOMS  indicate 28 FAILURES (6-7 partitions)
28 combinations of 6 SYMPTOMS  indicate 28 FAILURES (7-9 partitions)
56 combinations of 7 SYMPTOMS  indicate 28 FAILURES (8-10 partitions)
70 combinations of 8 SYMPTOMS  indicate 28 FAILURES  (8-11 partitions)
56 combinations of 9 SYMPTOMS  indicate 28 FAILURES  (9-11 partitions)
28 combinations of 10 SYMPTOMS  indicate 28 FAILURES (10-12 partitions)
8 combinations of 11 SYMPTOMS  indicate 28 FAILURES (11-12 partitions)
```

The first of these provides:

```
ENGINE.engine_fuel_feed = none AND CP_FL_LH.Control = on
INDICATES FAULTS:
Pipe5.blocked; Pipe5.fracture; Pipe2.blocked; Pipe3.blocked; Pipe1.blocked; Pipe6.blocked;
Pipe6.fracture; Pipe7.blocked; Pipe7.fracture; Pipe4.blocked; Pipe4.fracture
```

The second entry considers the best pair of sensors:

```
ENGINE.engine_fuel_feed = none AND CP_FL_LH.Control = on
INDICATES FAULTS:
Pipe5.blocked; Pipe5.fracture; Pipe2.blocked; Pipe3.blocked; Pipe1.blocked; Pipe6.blocked;
Pipe6.fracture; Pipe7.blocked; Pipe7.fracture; Pipe4.blocked; Pipe4.fracture

ENGINE.engine_fuel_feed = low fuel
OC_WT_LH.tank_level = higher than expected
FT_FL_LH.flow = low
INDICATES FAULTS:
Pipe5.partialblocked; Pipe2.partialblocked; Pipe3.partialblocked; Pipe1.partialblocked;
Pipe6.partialblocked; Pipe7.partialblocked; Pipe4.partialblocked
```

In this case, there are three symptoms that all indicate the same set of failures and we could choose any one of these. Sometimes there may be several sets of (non equivalent) symptoms able to discriminate the same number of faults. In the example above, this occurs for 5 symptoms, where 8 different combinations of symptoms indicate 28 faults (though not necessarily the same 28 faults). They may have different fault isolating capability dividing the faults into 6 or 7 partitions provided by different combinations of the chosen symptom set. These partitions each contain a set of one or more faults that are indistinguishable using the selected set of observations used by the selected symptoms. From the above table, we see that 4 symptoms are adequate to identify all 28 faults present in the FMEA of the system.

```
ENGINE.engine_fuel_feed = none AND CP_FL_LH.Control = on
INDICATES FAULTS: Pipe5.blocked; Pipe2.blocked; Pipe3.blocked; Pipe1.blocked;
Pipe6.blocked; Pipe7.blocked; Pipe4.blocked

ENGINE.engine_fuel_feed = low fuel
OC_WT_LH.tank_level = higher than expected
FT_FL_LH.flow = low
INDICATES FAULTS: Pipe5.partialblocked; Pipe2.partialblocked; Pipe3.partialblocked;
Pipe1.partialblocked; Pipe6.partialblocked; Pipe7.partialblocked; Pipe4.partialblocked

ENGINE.engine_fuel_feed = air
INDICATES FAULTS:
Pipe2.fracture; Pipe2.leak; Pipe3.fracture; Pipe3.leak; Pipe1.fracture; Pipe1.leak

ENGINE.engine_fuel_feed = none AND CP_FL_LH.Control = on
OC_WT_LH.tank_level = lower than expected
```

```
INDICATES FAULTS: Pipe5.fracture; Pipe6.fracture; Pipe7.fracture; Pipe4.fracture

OC_WT_LH.tank_level = lower than expected
INDICATES FAULTS: Pipe5.leak; Pipe6.leak; Pipe7.leak; Pipe4.leak
```

This analysis demonstrates that the fuel feed detected by the engine is the most important available diagnostic indicator, followed by an incorrect fuel level in the tank. The flow and pressure sensors are not even necessary to detect a full set of faults. However we could decide that the engine fuel feed is not a feasible observation, and by removing it and running the analysis again we find that the flow meter becomes an important sensor.

It is also possible to maximise the fault isolation capability using a modified version of the symptom ranking algorithm that maximises the number of fault partitions instead of simply the number of faults. We find that 7 symptoms is the fewest that can identify all 28 faults and divide them into 10 separate sets of (indistinguishable) faults. We also find that the best fault isolation that is possible is 12 partitions of faults using various combinations of the following 12 symptoms:

```
(Pipe2.partialblocked Pipe3.partialblocked Pipe1.partialblocked Pipe6.partialblocked
Pipe7.partialblocked) (Pipe5.partialblocked Pipe4.partialblocked) (Pipe4.leak) (Pipe5.leak
Pipe6.leak Pipe7.leak) (Pipe2.leak Pipe3.leak Pipe1.leak) (Pipe2.fracture Pipe3.fracture
Pipe1.fracture) (Pipe5.blocked Pipe4.blocked) (Pipe6.blocked Pipe7.blocked) (Pipe6.fracture
Pipe7.fracture) (Pipe5.fracture) (Pipe2.blocked Pipe3.blocked Pipe1.blocked) (Pipe4.fracture)
```

This in fact includes all the symptoms in figure 4 because S2, S3, and S5 turn out to be identical as mentioned above.

The above paragraphs assume that all faults are equally important in terms of diagnosis. In practice, there are several additional considerations. Often there is a limit to the granularity required by fault isolation due to the presence of LRU's. There is no requirement to be able to isolate a fault beyond a single LRU.

Observations may also fall into categories including: Basic system sensing that must be available for nominal operation; Groups of additional observations that only make sense to provide as single units (sensors). In the future we will be possible to include these additional factors in the diagnostic generation be further structuring of the symptoms, observations and measures used to select symptom permutations.

This diagnosability goes beyond the diagnostic work described in section 5. That work is able to generate diagnostics for existing sensor placements. The diagnosability work provides the opportunity to decide where sensors should be placed for the most efficient and effective diagnostics.

7. Conclusions

The ASTRAEA project is concerned with researching the methods and technologies needed to be able to routinely fly UAVs safely in commercial airspace. The work described in this paper contributes to this goal in the following ways:

- It automates the generation of failure effects for an FMEA report, providing consistent results for a complete set of component failure modes. It can be guaranteed that results are produced for all component failure modes that are modelled.
- The generated FMEA results can be arranged as failure-symptom pairs with the same consistency, and can be linked to specific symptoms that are observable by an on-line system.
- The failure-symptom pairs have been integrated into a larger diagnostic system, where other failure-symptom pairs will have been produced by other methods.
- Further analysis can indicate the most effective points within the system being diagnosed to place sensors, assisting in the decision of where sensors should be placed when designing the system.

Acknowledgement

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

References

[1] C. J. Price. Function Directed Electrical Design Analysis, Artificial Intelligence in Engineering, 12(4), 445–456, 1998.

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

[3] C. J. Price, AutoSteve: Automated Electrical Design Analysis, in Prestigious Applications of Artificial Intelligence (PAIS-2000), Berlin, August 2000, in Proceedings ECAI-2000, 721-725.

[4] M. H. Lee. Qualitative circuit models in failure analysis reasoning. Artificial Intelligence 111:239–276. 1999.

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

[6] N. Snooke. M2CIRQ: Qualitative fluid flow modelling for aerospace FMEA applications. In Proceedings 21st International Workshop on Qualitative Reasoning, (QR-2007), 161-169, 2007.

[7] J. B. Bowles, & , R. D. Bonnell. Failure Mode Effects and Criticality Analysis: (What it is and How to use it), in Topics in Reliability & Maintainability & Statistics, Ann. Reliability and Maintainability Symp., from 1993-1998, 32 p. (revised each year)