PROPRIETARY RIGHTS STATEMENT

THIS DOCUMENT CONTAINS INFORMATION, WHICH IS PROPRIETARY TO THE CRYSTAL CONSORTIUM. NEITHER THIS DOCUMENT NOR THE INFORMATION CONTAINED HEREIN SHALL BE USED, DUPLICATED OR COMMUNICATED BY ANY MEANS TO ANY THIRD PARTY, IN WHOLE OR IN PARTS, EXCEPT WITH THE PRIOR WRITTEN CONSENT OF THE CRYSTAL CONSORTIUM THIS RESTRICTION LEGEND SHALL NOT BE ALTERED OR OBLITERATED ON OR FROM THIS DOCUMENT. THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S SEVENTH FRAMEWORK PROGRAM (FP7/2007-2013) FOR CRYSTAL – CRITICAL SYSTEM ENGINEERING ACCELERATION JOINT UNDERTAKING UNDER GRANT AGREEMENT N° 332830 AND FROM SPECIFIC NATIONAL PROGRAMS AND / OR FUNDING AUTHORITIES.

CRitical S**YST**em Engineering **A**cce**L**eration

Airbus Fuel Management Risk Analysis Use Case Description Report –V1

WP 2.1.c: Fuel Management Risk Analysis

D211.010

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	**Fuel Management Risk Analysis**
Deliverable No.	D211.010
Dissemination Level	CO
Nature	R
Document Version	V01.00
Date	2014-03-11
Contact	Dr. Xiaodong Wu
Organization	Airbus Operations Ltd.
Phone	44 1179360243
E-Mail	Xiaodong.wu@airbus.com

AUTHORS TABLE

Name	Company	E-Mail
Xiaodong WU	Airbus Operations Ltd	Xiaodong.wu@airbus.com
Christopher SLACK	Airbus Operations Ltd	Christopher.Slack@Airbus.com
Iiria-Romina I RODRIGUEZ-BETANCOURT	Airbus Operations Ltd	iiria-romina.i.rodriguez-betancourt@airbus.com
Chris PAPADOPOULOS	Airbus Operations Ltd	Chris.Papadopoulos@Airbus.com

REVIEW TABLE

Version	Date	Reviewer
V01.00	2014-03-13	Jean-Luc Johnson
V01.00	2014-03-13	Andreas Mitschke

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
V01.00	2014-03-11	Initial version	

CONTENT

Content of Tables

Content of Figures

Content of Appendix

No table of contents entries found.

1 Introduction

1.1 Role of deliverable

This document will describe Airbus Fuel Management Risk Analysis use case. It provides an overview of civil aircraft fuel system, fuel system management risk analysis process, methods and tools.

The CRYSTAL Fuel Management risk analysis use case work package (WP2.01C) has the following major purposes:

- Define of the overall use case, including a detailed description of the underlying development processes and the set of involved process activities and engineering methods
- Provide input to WP601 (IOS Development) required to derive specific IOS-related requirements
- Provide input to WP602 (Platform Builder) required to derive adequate meta models
- Establish the technology baseline with respect to the use-case, and the expected progress beyond (existing functionalities vs. functionalities that are expected to be developed in CRYSTAL)

1.2 Relationship to other CRYSTAL Documents

The Fuel Management risk analysis use case is supporting the PRA use case led by Airbus France, the more detailed information about the PRA use case is available in document D210.010

1.3 Structure of this document

In this document, we describe the Fuel system and associated architecture; focus on Fuel Quantity Management System. The Safety analysis for the impact of Uncontained Engine Rotor Failure (UERF), one of most critical Particular Risk Analysis is illustrated. Then we describe fuel function modelling and simulation process and safety model-based analysis process, the associated tools chain to be developed in the frame of CRYSTAL. The engineering methodology is described as well. The more detailed information will be written in the next version of report.

First version of the use-case definitions is describing the associated technology bricks and the meta-model of the platform builder.

2 Use Case Description (1) – Fuel System

The following section describes the system that will then be used to create the models used for safety assessment. Section 7 describes the actual model that represents this system.

2.1 System Definition

The primarily purpose of the fuel system is to ensure the required fuel feed supply to the engines. In addition to the Engine Feed function, other systems functions are needed to ensure a suitable fuel system management, including fuel quantity measurement and fuel distribution. The following sub-paragraphs offer a brief description of the most commonly Fuel System functions provided for civil aircraft.

2.1.1 Engine and Auxiliary Power Unit (APU) Feed

The Engine Feed is commonly achieved by the operation of engine feed pumps. There is normally a main pump and a standby pump for each engine, so the system can operate the standby pump when the main one is failed. Each engine normally has a dedicated fuel supply valve, which can be commanded closed to isolate the engine from the Fuel System. Engine feed is normally designed in a way each engine can be fed from any fuel tank (fuel redirection achieved by crossfeed valves), this feature reduces the possibility of engine fuel starvation and allows fuel leakage isolation.

APU Feed line is typically connected to the Engine Feed lines. It also has a dedicated fuel supply valve, which isolates the APU from the fuel supply. Normally, an APU Pump is provided to provide pressurized fuel to the APU when the engines are not operating (on ground).

2.1.2 Fuel Transfer

Fuel Transfer function is to redistribute fuel between the aircraft fuel tanks. Normally, the fuel is transferred from the centre and Aft tanks to the engine feed tanks for engine feed (fuel supply is usually ensured by collector cells). Fuel transfer can be also performed to modify the aircraft's Centre of Gravity (CG) position. Depending of the design, a number of transfer pumps and transfer valves can be provided in the fuel tanks for fuel redistribution.

2.1.3 Refuel / Defuel

The Refuel/Defuel Sub-System controls the flow of fuel into or out of the aircraft via the refuel Coupling. Usually, an aircraft can be refuelled in automatic or manual modes, and the fuel can be supplied to the fuel tanks via tank inlet valves, allowing the selection of tank fuel supply.

2.1.4 Fuel Jettison

Fuel Jettison functions allows the system to jettison large amounts of fuel overboard to reduce aircraft weight. This function avoids overweight landings, preventing potential landing gear and /or structural damage.

2.1.5 Fuel Tank Vent

Fuel Tank Vent system connects the fuel tanks with the outside air. The aim of the system is to prevent high pressure differentials to be developed within the fuel tanks during the different aircraft flight phases. The interface with the outside air is commonly achieved via a NACA duct installed in the surge tanks (vent tanks which also acts as a reservoir in case fuel can enter the vent system), allocated at the outermost position of the fuel tank arrangement.

2.1.6 Fuel Quantity and Management System

The Fuel Quantity and Management System is in charge of fuel control and monitoring. This sub-system is usually managed by a control computer (two redundant computers are normally provided to ensure redundancy, one of them as a standby), which acquires data from different gauges and sensors located in the fuel tanks. The FQMS also provides valves and pumps commands for the system to perform its intended functions, as well as system indications and warnings for interaction with the flight crew.

2.2 System Architecture

2.2.1 Fuel System Configuration

The Fuel System storages the fuel in a series of tanks allocated in the wings, horizontal stabilizer and/or fuselage. The fuel is redistributed between the tanks to ensure engine feed and other functions as lateral and longitudinal CG position modification.

In-tank equipment as sensor and fuel probes are provided for fuel quantity management and monitoring. The data is acquired and sent to the control computer, which provides control commands to in-tank valves and pumps to perform engine feed, fuel transfer, jettison or any other required function.

The following picture represents a simplified typical civil aircraft Fuel System layout and general Electrical System schematic:

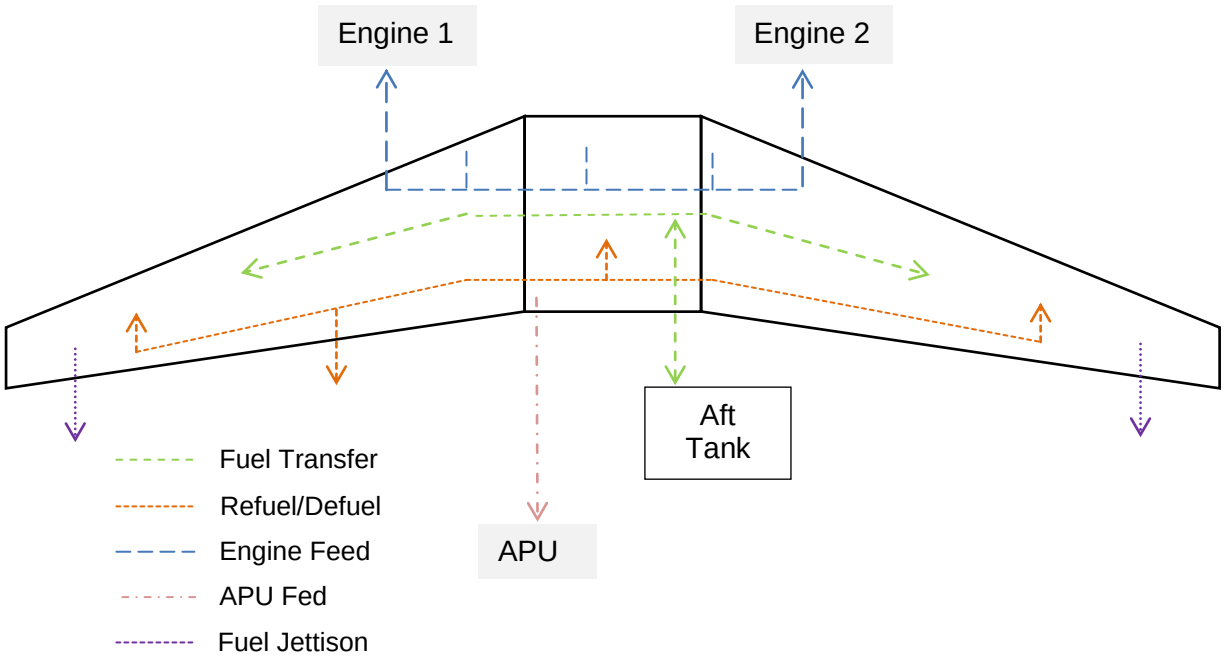


Figure 2-1: Simplified Typical Fuel System Layout

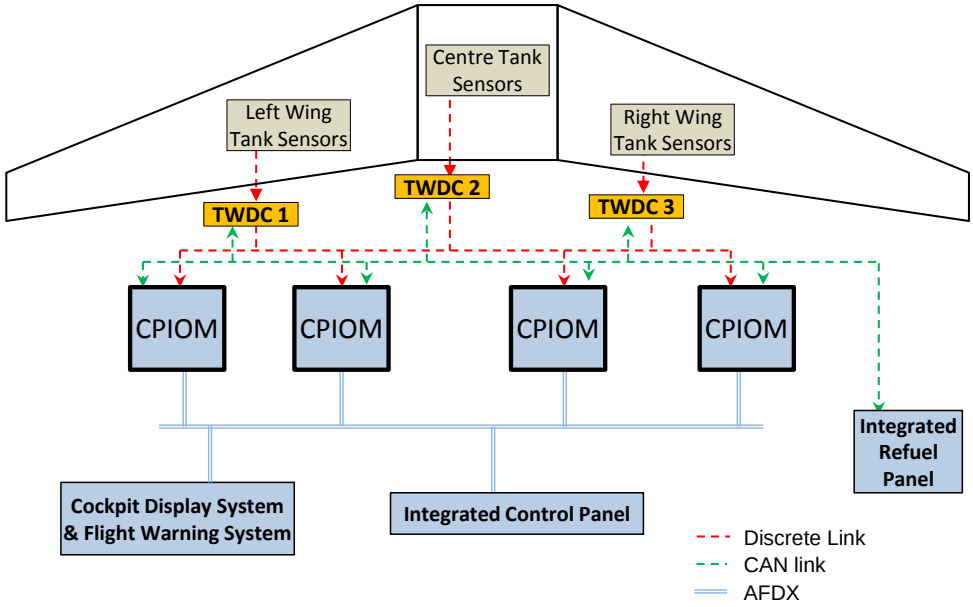


Figure 2-2 Fuel System Electrical Network Schematic

2.2.2 Fuel System Components

The following chart provides a summary of the most typical Fuel System components used on civil aircraft:

Component Type	Description
Electronic Equipment	
Control Computer	Typically consists in two segregated and independent computers (one of them is in control while the second one is stand-by, available in case of failure). They are provided to manage the fuel system, and to provide alerts and indications to the flight crew.
Fuel Tank Data Concentrator	Equipment intended to collect data from in-tank components (Probes, sensors, valves…) in order to be transmitted to the control computer.
Electrically actuated valves	
Transfer Valves	Controls fuel flow in the transfer gallery to re-distribute fuel between tanks.
Crossfeed Valves	Allows either the engines and APU can be fed from any fuel tank.
LP Valves	Stops fuel flow to the engines from fuel system when required
Tank Inlet Valves	Controls fuel flow into the tanks.
APU Valve	Stops fuel flow to the APU when required
Refuel Valve	Controls fuel flow between the fuel gallery ground refuelling / defueling Equipment.
Jettison Valves	Allows discharge of fuel from all tanks overboard to reduce the fuel load and hence the aircraft weight
Probes and sensors	
Fuel Probes	Provided for fuel quantity measurement.
Temperature Sensors	Measures fuel temperature.

Component Type	Description
Fuel Characteristics Sensors	Measures fuel properties as density, permittivity and temperature.
Fuel Pumps	
Transfer Pumps	Pump fuel from one tank to the fuel transfer gallery.
Engine Feed Pumps	Pumps fuel from the engine feed tanks (collector cell) to the engine feed gallery.
APU Pump	Pumps fuel to the APU feed line.
Mechanical & fluid actuated equipment	
Jet Pumps	Provided for fuel and/or water scavenge in the fuel tanks.
Non-return Valves	Ensures fuel flow in only one direction, provide the means to prevent fuel path backwards.
Surge Relief Valves	Provided to minimize surge pressure produced when a shut-off valve closes.
Thermal Relief Valves	Provided to limit the fuel gallery pressure generated from thermal expansion of fuel in a closed section.
Air Release Valves	Allows air to escape from the fuel gallery to prevent air being fed to the engines or APU.
Water Drain Valves	Typically installed at the low points of the tanks, allows the water to be removed by manual operation of the valve.

2.3 Control and Indication

2.3.1 Instrumental Control Panel (ICP)

The Instrumental Control Panel is provided to allow the flight crew to control functions as refuel, defuel, fuel transfer, crossfeed, Engine fuel isolation, APU feed and jettison. These control functions are facilitated using switches, pushbuttons and/or FAULT lamps on the ICP.

The following picture represents the typical layout shown on Fuel System ICP (image represents fuel controls preliminary design):

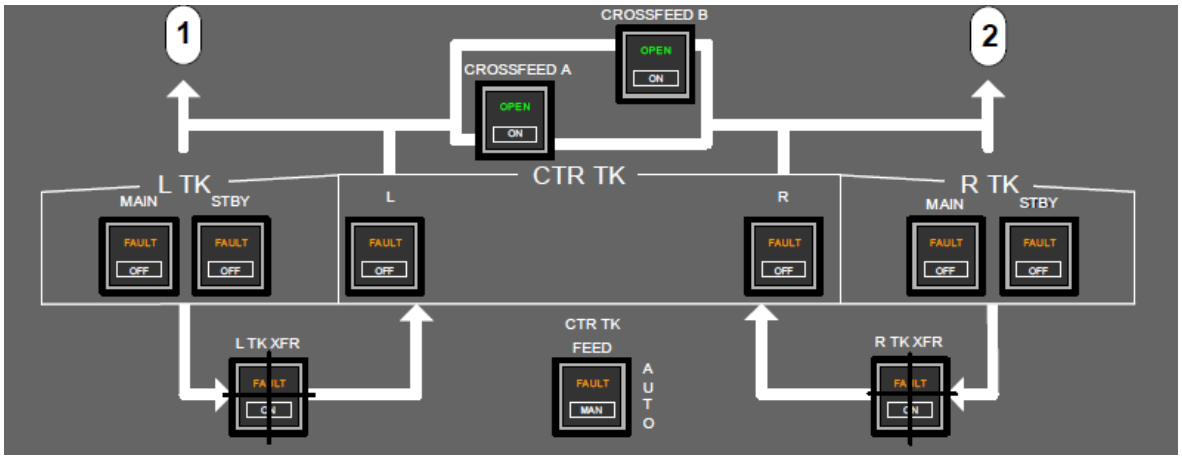


Figure 2-3: Integrated Control Panel (ICP)

2.3.2 Common Data System

It provides fuel system information to flight crew such as: valves and pumps status, fuel temperature and fuel quantity. Additional information (processed by the control computer) as aircraft gross weight and centre of gravity is typically can be also provided.

The following picture represents the typical layout shown on Fuel System data page. This image represents typical ECAM (Electronic Centralized Aircraft Monitoring) FUEL page preliminary design).

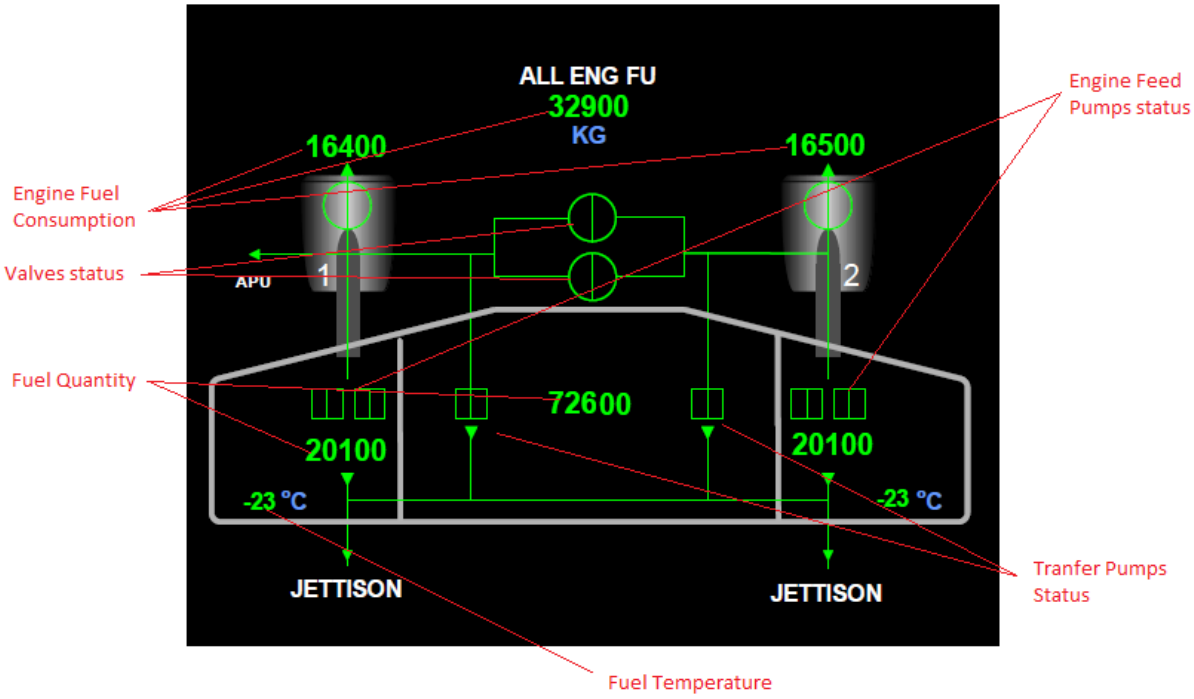


Figure 2-4: ECAM Fuel Display Page

2.3.3 Flight Warning System

FWS shows centralized alerts (audible and visual) in response to the data provided by fuel system, displaying information that requires flight crew action or awareness. The FWS alerts are commonly displayed in the ECAM, and their classification depends on the criticality of the failure associated to the alert. FWS also provides the required crew action for failure management, displaying the detailed procedure.

2.4 System Interfaces

The typical fuel system interfaces (highly integrated fuel systems) are listed in the chart below:

Interfacing System	Interface Description
Integrated Control Panels (ATA 31)	Integrated Control Panels allow commands to be directed to the Fuel System
Flight Warning System (ATA 31)	The Flight Warning System monitors the Fuel System for failures and provides alerts to the flight crew
Control and Display System (ATA 31)	The Fuel System provides data to the Control and Display System
Electrical System (ATA 24)	The electrical system provides power for all fuel equipment that require electrical power
Flight Management System (ATA 22)	Data transmission as Aircraft Gross Weight and Centre of Gravity
Landing Gear Extension and Retraction System (ATA 32)	Used to determine Ground/Flight Status for fuel measurement and management purposes
Aircraft Data Communication Network (ATA 42)	Provides data transmission and reception between aircraft systems
Power plant Control System (ATA 70)	Provides the Fuel System with data to determine Engine Fuel Consumption and status
Auxiliary Power Unit (ATA 49)	Auxiliary Power System provides data to the Fuel System to facilitate control of the APL LP Valve and the APU Pump
Inert Gas Generation System (ATA 47)	Inert Gas Generation System supplies Nitrogen Enriched Air (NEA) to the fuel tanks to reduce the fuel tank ullage flammability exposure
Central Maintenance System (ATA 45)	Central Maintenance System provides facilities for reporting Fuel System faults for maintenance purposes
Digital Flight Data Recording System (ATA 31)	Acquires critical parameters from the Fuel System for investigative purposes
Fuselage and wings (ATAs 53 and 57)	Have a mechanical interface with the Fuel System as equipment is mechanically attached to the fuselage and wing structure.

2.5 Fuel System Sub-systems

A typical Fuel System performs the following functions:

- Supply fuel to the Engines.
- Supply fuel to the APU.
- Control Tank Pressures.
- Manage fuel distribution, including refuel, containment, distribution, defuel and
- Jettison.
- Indicate fuel state, including quantity and temperature.
- Provide indication and support for maintenance activities.
- Provide Gross Weight and Centre of Gravity data.

In the same way, there is a list of fuel sub-systems, intended to carry out the functions the Fuel System is designed to perform:

- Fuel Containment
- Venting
- Refuel, Defuel and Ground Transfer
- Engine and APU Feed
- Fuel Quantity Management System
- Fuel Scavenge
- Fuel Jettison

The following table provides a list of the equipment involved in each of the Fuel System main functions:

Fuel System Function	Related Sub-system/s	Involved equipment
Supply Fuel to the Engines	• Engine and APU Feed • FQMS	• Engine Feed Pumps • LP Valves. • Crossfeed Valves • Thermal Relief Valves • Air Release Valves • Pressure Holding Valves • Clack Valves (collector cells). • Non Return Valves
Supply Fuel to the APU	• Engine and APU Feed	• Engine Feed Pumps • APU Pump • APU LP Valve • APU Isolation Valve • APU Drain and Vent Valve
Control Tank Pressures	• Venting	• Vent Line Fuel Drain Valves. • Overpressure Protectors. • NACA Inlets/Outlets.
Manage fuel distribution, (including refuel, containment, distribution, defuel and jettison)	• Refuel, Defuel and Ground Transfer • FQMS • Fuel Containment • Water Scavenge • Fuel scavenge • Jettison	• Transfer Valves • Crossfeed Valves • Tank Inlet Valves • APU Valve • Refuel Valve • Jettison Valves • Transfer Pumps • Jet Pumps
Indicate fuel state, (including quantity and temperature)	• FQMS	• Fuel Probes • Temperature Sensors • Fuel Characteristics Sensors
Provide Gross Weight and Centre of Gravity data.		
Provide indication and support for maintenance activities.	• FQMS	All equipment except the ones involved Fuel Containment and Venting

3 Use Case Description (2) - Uncontained Engine Rotor Failure (UERF)

3.1 Certification Requirements

UERF is one of the Particular Risk Analysis (PRA) considered during the aircraft Common Cause Analysis. The analysis is intended to define design precautions to be taken in order to minimise the hazards in the event of UERF. The analysis must show compliance with the following certification requirements:

> CS/FAR 25§903(d)(1)
> *"Design precautions must be taken to minimise the hazards to the aeroplane in the event of an uncontained engine rotor failure …"*

The associated Acceptable Mean of Compliance AMC 20-128A sets forth a method of compliance with the requirements of § 23.901(f), 23.903(b)(1), 25.901(d) and 25.903(d)(1) of the EASA regulations pertaining to design precautions taken to minimize the hazards to an airplane in the event of UERF.

3.2 Analysis Approach and General Assumptions

There are two approaches followed during UERF analysis:

- Qualitative approach: it has to be demonstrated that practical design precautions have been taken to minimize the consequences on the A/C following an UERF event.
- Quantitative approach: quantification of the residual risk once all practical design precautions have been taken.

To analyse an UERF model, general assumptions should be made:

- Only one UERF event and one trajectory are considered all along the flight.
- The rotor fragment is supposed to have a straight trajectory before and after any structure/system perforation.
- The probability of release of debris within the maximum spread angle is uniformly distributed over all directions.
- No combination with other PRA should be considered.
- Additional conditions associated with the Master Minimum Equipment List (MMEL) must not be considered in combination with the UERF event.
- An UERF event is not considered detectable. The UERF could not be visible from cockpit or cabin (However, the flight crew could suspect an UERF event has occurred).
- UERF trajectory analysis is performed on A/C in JIG position.

Other more specific assumptions can be made as applicable for each system. Whenever possible, the general design precautions principles that should be considered for essential systems are:

- Installation of critical systems out of the burst area.
- Installation behind heavy primary structure.
- Segregation of redundant circuits.
- Use of specially segregated electrical routes

3.3 UERF Model

The AMC 20-128A provides the following proposal to show compliance with certification requirements. The following chart provides guidance to delimitate risk areas and to specify in which cases a quantitative analysis is required:

Basic Model			Alternative Model*		
Fragment	Spread angle	Quantitative Objectives	Fragment	Spread angle	Quantitative Objectives
Single 1/3rd Disc	+/- 3°	1/20*	Single 1/3rd Disc	+/- 5°	1/20
Intermediate	+/- 5°	1/40			
Fan Blade	+/- 15°	No quantitative analysis	Fan Blade	+/- 15°	No quantitative analysis
Small Fragments	+/-15°	No quantitative analysis	Small Fragments	+/-15°	No quantitative analysis

*Alternative model is the one typically considered by Airbus.
**Only one in twenty trajectories is Catastrophic.

The figures below represent an example of UERF area definition or 'risk areas':

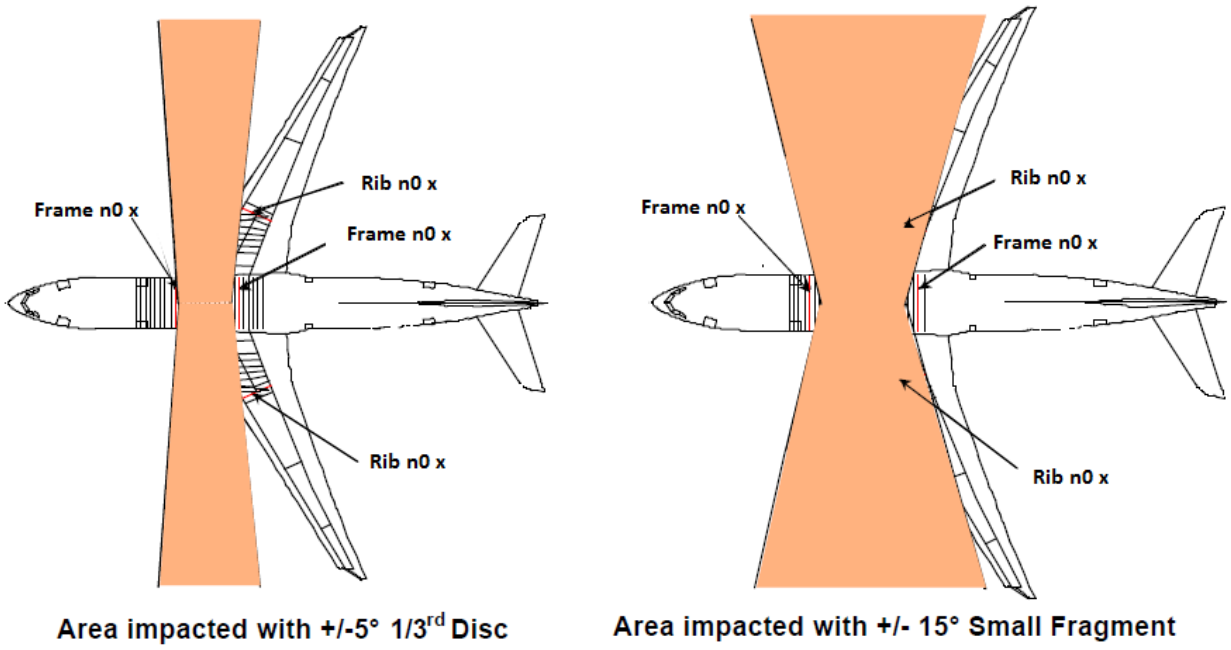


Figure 3-1: Example UERF Zones

3.4 UERF Analysis

3.4.1 Damage Component Status

There is a generic failure 'status' to be considered following an UERF according to the type of debris. Systems components are considered unserviceable if their envelope has been touched. In case of an

engine being impacted, the nacelle structure may be regarded as engine envelope, unless damage is not likely to be hazardous (AC20-128a §Appendix 4.a. (6)).

The following chart lists the generic failure modes considered after UERF. Additional component failed 'status' which is considered realistic and relevant for the analysis must be added as applicable.

Type of component	Failure Mode to be considered	
	1/3rd Disc	Small Fragment
Equipment (computer, power source…)	Lost	1st equipment impacted after penetration of significant structure is considered lost.
Wiring	Cut open	1st equipment impacted after penetration of significant structure is considered lost.
Mechanical part	Broken or jammed	Structural assessment to be performed.
Pipe	Clean cut	Clean cut of 1st pipe impacted after penetration of significant structure.
Structure	Clean cut	Structural assessment to be performed.

The combination of failures after an UERF event shall be analysed against system Failure Conditions list. This will allow identifying the design precautions needed to avoid such catastrophic scenarios as far as practical, and to assess the remaining risk.

3.4.2 System Failure Conditions

The UERF qualitative analysis is focused into determine if a given fragment trajectory within the risk area could lead into a Catastrophic Failure Condition. The system Failure Conditions are determined by means of the Functional Hazard Analysis (FHA), performed as part of the System Safety Assessment (SSA).

Each Failure Condition represents a given scenario (related to a functional failure) and is categorized considering the safety effects at aircraft level. A Catastrophic Failure Condition is considered to lead to multiples fatalities and potential total loss of the aircraft.

One of the critical functions within Fuel System is: 'To Supply Fuel to the Engines' (see section 2.5). The Catastrophic Failure Condition derived from this function is: 'Total loss of fuel supply to the engines'. For this failure condition to occur, a given number of failure(s) have to be present in the system equipment.

Fuel System Function	Related Sub-system/s	Involved equipment
Supply Fuel to the Engines	• Engine and APU Feed • FQMS	• Engine Feed Pumps • LP Valves. • Crossfeed Valves • Thermal Relief Valves • Air Release Valves • Pressure Holding Valves • Clack Valves (collector cells). • Non Return Valves

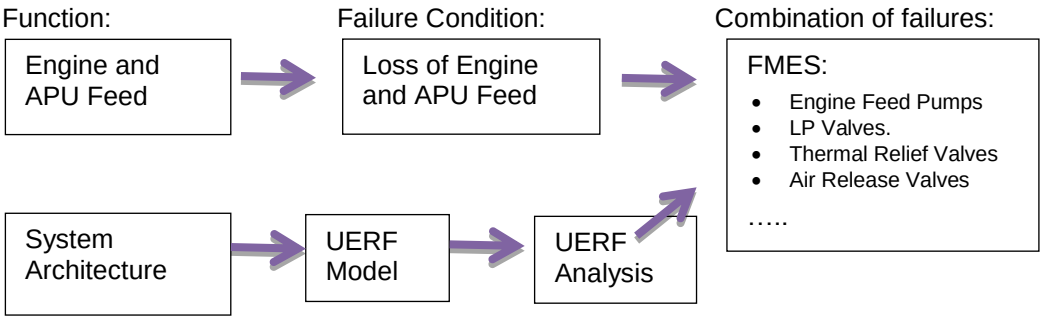
Additionally, System Failure conditions consider contribution from failures of electrical wiring (power distribution), installation (pipes, fittings…) and external events (ice, crosswind.). Failures from interfacing systems are also addressed as contributions to the Failure Condition when applicable.

3.4.3 UERF Analysis and Results

The UERF analysis is performed in order to ensure that the Fuel System design meets the required safety precautions required to ensure that in case of UERF, the system will not develop into a catastrophic Failure Condition.

To identify the impact at system level, the UERF model provides a list of impacted items (electrical routes, pipes, equipment) per trajectory; this is called the 'hit list'. The list of affected items allows recreating the effect of a given trajectory into the safety model (i.e. RAMSES) in order to determine if the combination of failures will develop into a catastrophic failure condition.

Considering the system function Engine and APU Feed (see sub-chapter 2.1.1), it is assumed that the combination of failures affecting a given number of components (see sub-chapter 2.2.2) could develop into the functional failure: 'Loss of engine and APU feed'. The list of affected components from the UERF model allows comparing the existing dysfunctional model with the effect of the UERF event at system level.



4 Model Based Safety Analysis Approach Evolution

4.1 The Classical Safety Approach

Any system is defined by system designers on the basis of data (system specification, RAMS objectives…) provided by the aircraft manufacturer or the systems design authority. The system designers elaborate architectures[1] which are evaluated in order to assess the likelihood of compliance with safety requirements.

Assessing the safety of systems functions entails analysing its architecture with consideration of its interfaces focusing on:

- Design principles,
- Required resources for the nominal behaviour of each system component,
- Possible failure modes and functional mechanisms (monitoring, reconfigurations…) elaborated to limit/control their effects,
- Dependencies between system components.

In order to perform a safety assessment, the safety analyst has to understand the function/system behaviour, especially in the presence of failure modes and functional mechanisms linked to failure events. From function/system definition documents and discussion with function/system designer, the safety analyst creates his own understanding of system behaviour; this allows him to perform a safety assessment using classical methods (fault trees, dependence diagrams, Markov chains).

This safety assessment approach is illustrated on the following figure.

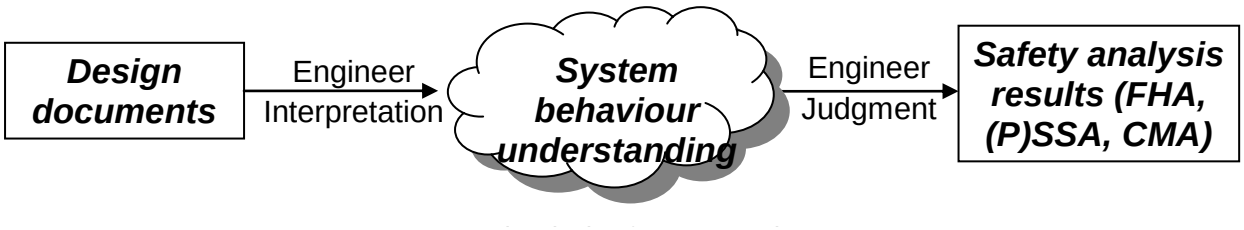


Figure 4-1: Classical safety approach

Understanding of system behaviour may be formalized by means of fault tree or dependency diagram and/or textual specification. Verification of safety analysis is then performed by reading/analysing safety representations (FT, DD, MA) using a graphical representation and textual specification if available.

4.2 Motivations Resulting From Recent Evolutions

The classical safety approach has been developed over a long period of years in order to ease the safety analysis of a given design philosophy (systems independence, strong segregations…) linked to limited technological means, and a given work organization. Natural evolution of technological means and work organization revealed important limitations to the static analyses types (FT, DD, MA) associated with this classical safety approach. These limitations highlighted the need for an improved safety assessment method.

[1] Several preliminary high level architectures, presenting candidate design principles, are evaluated during trade-offs enabling the selection of a preferred architecture that is further detailed up to implementation.

4.2.1 Technological Evolution Impacting Safety

Technological evolution of function/system design permits the development of more and more functions using common resources (e.g. Modular avionics) and/or sharing information from other systems (e.g. weight on wheels, air data information, navigation information, radio altimeter information). This increased sharing of common resources leads to an increase in the complexity of required system reconfiguration, functional interactions, interactions between systems and man machine interfaces. This increased complexity induces:

- an increased difficulty in the performance of safety analyses and thus risk of overlooking safety issues,
- a need for multi-system analyses, effective means of verifying that transverse functions satisfy aircraft level requirements.

Design engineer safety-assessment skills are usually limited to the understanding of Fault Trees and Dependency Diagrams. In addition, such "static" analyses tools are often poorly suited to describing a system's dynamic behaviour.

Those two considerations induce:

- Difficulties for design engineers to verify that safety analysts have correctly understood the operation of complex system behaviour.
- Increased risks of potential misunderstanding.

These difficulties will increasingly become more apparent to independent verification authorities, such as Airworthiness Authorities and various other certification and verification experts.

In response to the difficulties described above, any **new safety assessment method** should provide a new and adapted means of communication:

- **easily readable and understandable** (without need for an in-depth safety knowledge),
- **merging dysfunctional and functional behaviour** into a **single dynamic description,**
- **allowing structured break-down of the system**, in order to more easily, master its complexity,
- **focusing verification of the correct understanding of system behaviour**, as well as on Failure Condition analysis results.

4.2.2 Evolution of Development Process and Industrial Practice Impacting Safety

Evolution of the development process and industrial practice during the last decades concern the interaction between design activities and safety assessment. Part of this evolution concerns the multiplication of actors linked to the dysfunctional behaviour of a common resource, and the economic constraints associated with industrial practice.

Better integration of safety in the development phase/process

Safety considerations have been progressively brought forward in development milestones. The need to consider safety as a primary design requirement has progressed significantly over the years, and long-gone are the days when primary design requirements were limited to cost, weight etc. No-where is this more-so the case than in the aviation industry. Safety assessments are now applied during early phases of architecture trade-off. This permits the detection of any design weaknesses as early as possible and helps avoid late and costly modification of architecture and wiring.

A new safety assessment method should facilitate the performance of continuous/progressive safety assessment during trade-off phases on preliminary architectures, and then be easily enriched to support certification analyses.

Organization evolution

In addition to safety, it is envisaged that other assessment types (reliability, maintainability) shall also use modelling and simulation. These other assessment types also consider dysfunctional behaviour in their analyses. Formalizing both functional and dysfunctional behaviour in a sequentially-dynamic model, abstracted at the safety level and with appropriate granularity, provides a common support for several disciplines. This would:

- **avoid the need for multiple verification work** by system designers,
- **guarantee consistency of understanding between the various disciplines**,
- **enrich the means of compliance** during potential discussions with certification authorities

Recent changes in industrial organization have seen aircraft manufacturers delegating increased design responsibilities to suppliers, whilst they retain a global integration, management and validation role. The resulting increase in supplier/customer interfaces brings a definite need for an improved means of communications, and it is considered that sequentially-dynamic models, offering a formalized understanding of functionality would represent a relevant/appropriate communication means between customer/supplier.

Economical constraints

Time-to-market reduction objectives have resulted in a series of methodology improvements.
With the classical safety analysis techniques (FT/DD/MA), each FC has to be individually analysed, with each FT/DD being built separately, and validated by the system design authority. Any architecture refinement or modification necessitates the need to check if the analysis model remains correct.

A major improvement in the safety methodology would be to **reduce verification of system/function understanding to a single support (model) common to** several FC analyses. The new safety methodology should **allow tools to support verification of understanding.**

4.3 Introduction to Model Based Safety Assessment Approach

Considering the previously mentioned difficulties, a new safety methodology has been developed in order to:

- Overcome safety-analyses issues related to increased complexity of functions and systems, by profiting from computer based methods:
  - Verification activity focused on accurate understanding of behaviour through the use of simulation, compared to just reading fault tree diagrams
- Provide a dynamic safety system description (to ease the verification of the analysis by the design engineer, Airworthiness Authorities or any expert).

This approach is called Model-Based Safety Assessment (MBSA).

The MBSA approach relies on the following:

- A formal language allowing safety system dynamics description,
- Formal tools (based on mathematical notion) allowing:-
  - Verification of the dynamic system behaviour by simulation (potentially graphical)
  - Support of the safety assessment by automatic MCS generation capability.

4.3.1 MBSA Process Overview

Once safety requirements have been specified the MBSA approach considers four main aspects in the performance of the system safety-assessment:

- Specification
- Implementation
- Validation
- Assessment

Obviously the MBSA approach is based on a correct & complete understanding of system behaviour and a subsequent abstraction from a safety point of view, resulting in a useful safety-oriented model of the system. The following aspects of the system being analysed have to be clearly identified:

- the system architecture,
- system component behaviour, including behaviour in the presence of individual failures modes

One of the safety abstraction operations consists in the identification of system components that are similar (i.e. same Inputs/Outputs, same Failure Modes, same internal behaviour…). This permits focus on a reduced set of items. Next, the system failure modes of each item are defined and their influence is formalized. Finally system components dependencies are described.

Safety requirements, such as Observer of a specific Failure Conditions, have to be formalized: this requires decomposing each FC as a logical formula based on expected system component outputs/status. The result is a System Safety Specification[2].

Once the system architecture and the behaviour of its components have been specified, they have to be written in a formal language and captured in a safety oriented modelling tool (i.e. a modelling tool whose functions support safety assessment). In order to enrich behaviour-description, such a modelling tool should enable a graphical representation of the model.

Since it is mainly focused on the dysfunctional aspects, the resulting model is referred to as a "Failure Propagation Model" (FPM).

In order to guarantee the relevance/correctness of generated simulation results, the model has to be consistent with the behaviour expected by system designers and described in the system safety specification. Therefore an independent validation has to be performed (i.e. performed by someone not Involved in the model construction). Note that simulation and analysis of the subsequent results are the main means by which the behaviour of the FPM can be assessed. The result is a validated FPM.

Finally, the system safety assessment is performed through analysis of the Observer results (cut sets) obtained from the FPM. The qualitative and quantitative safety results are automatically generated by the FPM (thanks to modelling tool functions). Those results are analysed in order to verify compliance with initial requirements and verify the system safety.

4.3.2 MBSA Summarized

MBSA is, in effect, a safety view of the system behaviour, formalized and implemented in a formal language in order to create a formal safety model called "Failure Propagation Model" (FPM). This model is a propagation-dynamic description: the system propagation-dynamics are integrated and highlighted thanks to dedicated formal tools.

[2] The System Safety Specification may represent the safety description contributing to the description chapter in SSA and may also be the main input to the modelling activity.

The following figure summarizes the MBSA approach, highlighted in the dashed red box on the lower part, as an alternative to the "classical" approach, highlighted in the dashed blue box on the upper part.

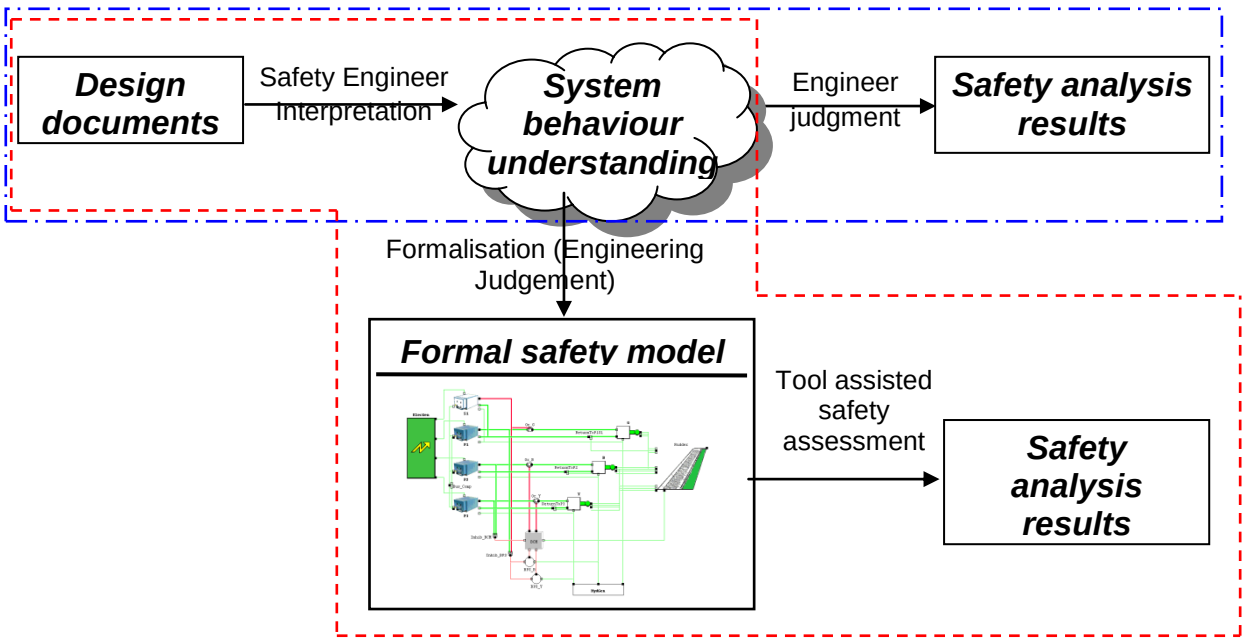


Figure 4-2: MBSA and classical safety approach

4.3.3 Formal Language Usage Motivations

Natural language allows any system to be described; the resulting description may only be understood by an analyst sharing the same natural language with the person that wrote the initial description. Formal languages have precise and finite semantics that are “understood” by appropriate software tools. When an analyst writes a specification in an appropriate formal language, tools are able to interpret it and perform certain deductions without any human intervention. This quality permits system behaviour to be described in formal software languages such that computers are able to effectively interpret the language and subsequently, perform simulations of system behaviour, under various configurations (e.g. fail state). Therefore system may be implemented in formal tools, taking benefit of constantly increasing computer computation power. Formal languages are well adapted to provide exhaustive exploration services on large combinatorial problems.

As mentioned before, the MBSA approach is based on the formalization of a system’s behaviour in a formal tool. When using the MBSA approach, an analyst manipulates 3 kinds of objects:

- Individual formalized items, textually described in a formal language,
- System architecture, graphically composed from predefined items,
- Safety results, automatically computed for a given system architecture.

These are the only objects visible to the analyst, but two more are actually elaborated in order to be able to generate safety results from a graphical representation of a system’s architecture:

- The formal tool comprises item descriptions of system components in order to elaborate the formal system description: this step uses code-generation which is automatically called when simulation is invoked,
- The system behaviour is deduced from the formal system description producing a state machine, also called Interfaced Transition System (ITS). The state machine is composed of states (one initial/nominal state and several non-nominal states) and transitions labelled by

failures or reconfiguration between two states; the formalization of a FC allows the formal tool to define if the FC is reached or not at each state

The following figure presents the main objects of the MBSA approach and summarizes the main steps leading to the production of safety results. Blue arrows symbolize steps that are visible to the analyst while white arrows represent hidden steps, automatically performed by the tool on request of safety results generation.

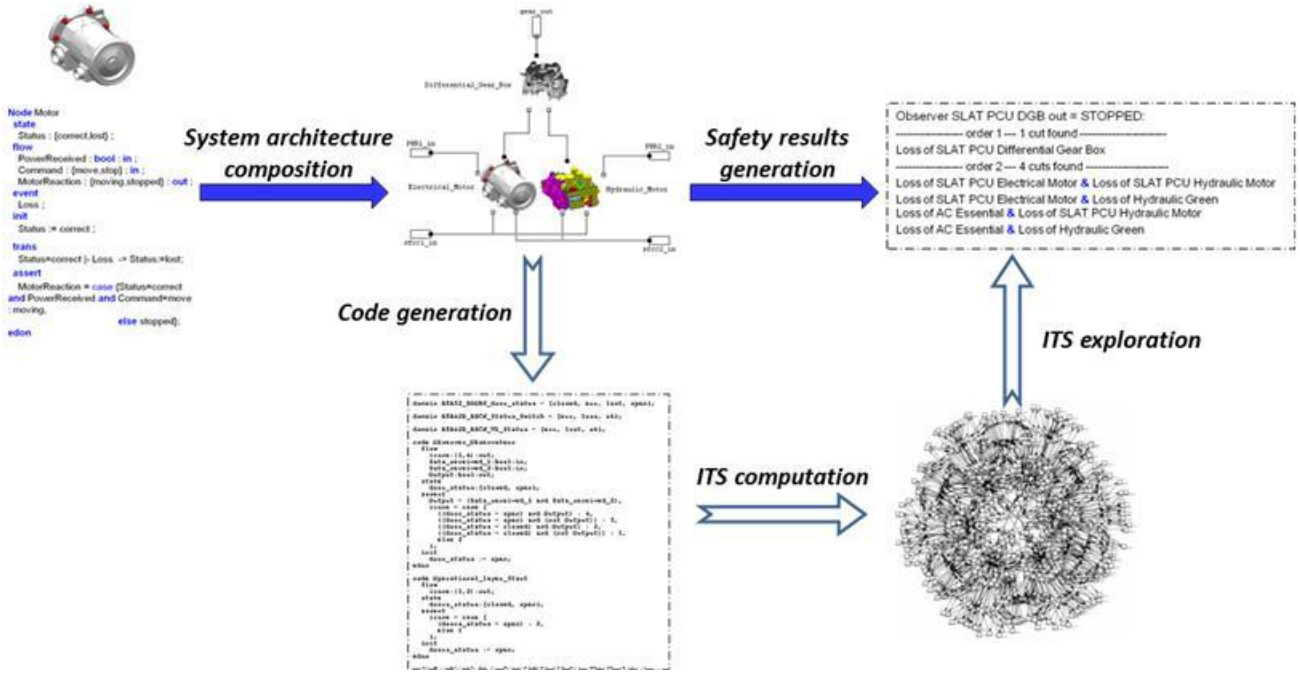


Figure 4-3: MBSA objects and steps

A major difficulty in the review of fault trees lies in the fact that part of the review heavily relies on reading through the huge number of failure combinations: The MBSA approach exploits the benefits of simulation in order to analyse much more complex state machines, representative of industrial systems. An assessment previously dependent to the engineering judgment in the classical approach becomes exhaustive in the MBSA approach. A consequence is that the confidence of the safety results is linked to the confidence in the model correctness and completeness: the validation step is of high importance.

4.4 Applications

The model based safety approach can be beneficial if applied at all stages of aircraft development, from Upstream architectural validation, down to aircraft level functions analysis, through the classical (Preliminary) System Safety Assessment, Linking in Common Cause Analyses and finally to supporting Continued Airworthiness Investigations as well as feeding back knowledge gained to update the SSA.

For the scope of the case study that will be applied in the CRYSTAL project only the Function PSSA and the Linking to a Particular Risk Analysis aspect of Common Cause Analysis will be developed to evaluate the capabilities developed within CRYSTAL.

5 Description of the model based safety methodology

The following flow chart shows all the different steps that are taken to arrive at a model that can be used for an analysis and then to use this model to generate the various results to support a PSSA.

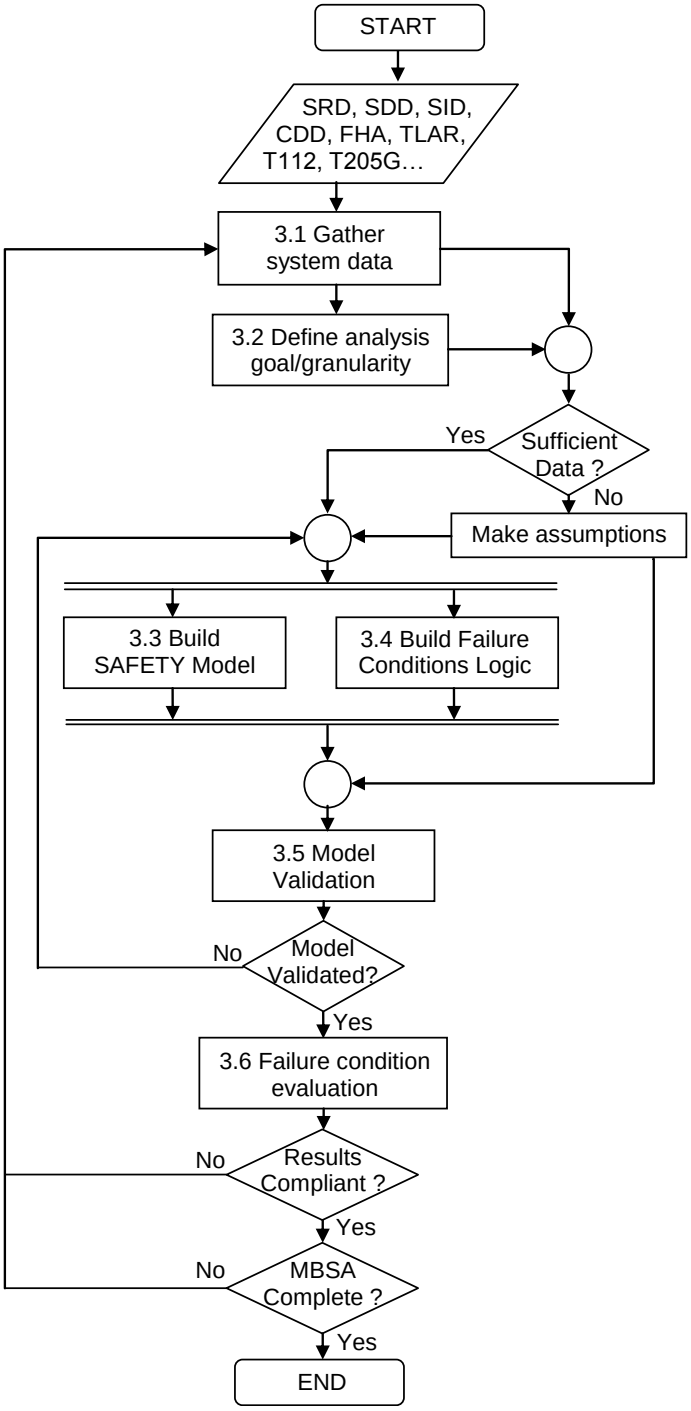


Figure 5-1: PSSA Modelling Flow Chart

5.1 Gather System Data

The same inputs are required to perform model-based safety assessment or document-based safety assessment. This first step remains common with the current safety process. It consists of gathering all available system data useful for safety to feed the process. This information may be found in design or safety documents.

It includes:

- **System architecture**: it describes system architecture (from SDD or SRD), i.e. relations between components or functions in the system and hierarchical organization inside it. Architecture allows highlighting redundancies and functional chains.
- **System behaviour**: it describes the internal way of working of the system. It may be described textually or using block diagrams from design modelling tools in the SDD. It rather may be extracted directly from those tools. Behaviour allows particularly highlighting functional reconfigurations.
- **System interfaces**: it describes the functions exchanged with other systems and the external systems/system components in interface with the studied system (from SID). Failures of these dependent systems may affect the nominal behaviour of the studied system.
- **Failure modes**: it describes the different failures which can affect the system components, their causes and their effects. This information is described in FMEA or FMES depending on the expected granularity level.
- **Safety requirements**: These requirements may come from aircraft level (TLAR/FHA(via T112)), from other systems (T205G) or from airworthiness constraints (CDD) (FCC's, DSF's, PRA's)
- **Failure conditions**: it describes the feared events to be studied and assessed. Failure conditions are defined in Functional Hazard Analysis performed at aircraft or system level. (FC's)
- **Safety attributes**: it describes useful system parameters for safety assessment. These attributes cover elements that have not been described by the previous elements. It is for example zonal information, technology, for common cause analysis. It can be also risk or maintenance times used for quantification.

5.2 Define the Goal and the Granularity of the Analysis

After having collected all the safety-relevant data, the safety analyst shall define the goal and the granularity level of his model-based analysis. Depending on the amount of information and the requirements, this level may evolve and force to define several assumptions, such as crew procedure is performed, failure confirmation time is reasonably short, etc.

The granularity is mainly driven by the analysis type (cf Applications). Each analysis has not the same goal. Common information on each analysis type and requirements associated to it are defined in program documents. They describe high-level guidelines on these steps. However, this is not sufficient and shall be adapted for each model.

During the first step, the safety analyst has identified several safety requirements and the main architectural and behavioural principles of the system. He is then able to define the model perimeter with all the involved equipments and interfaces and their safety-related attached information, as well as the observers needed to be implemented to cover the required failure conditions identified in the FHA. This perimeter defines a first needed granularity level entirely based on the goal to reach.

The safety analyst shall then crosscheck this perimeter with available information. In case of missing elements, it shall take assumptions either to reduce the perimeter or to complete them, by using generic elements, reference to in service experience or requirements (Through engineering

judgment). In any case, these assumptions shall be well documented to establish clearly what are the perimeter and the goal of the analysis and its domain of validity.

5.3 Build the Safety Model

After the previous steps, the safety analyst has all the needed to build his model. This operation is decomposed in three main phases, failure modes selection, architecture implementation and low level behaviour definition. It is supported by a MBSA tool offering the possibility to build the model textually or graphically. The following process represents a complete model creation. However, model elements may be stored in a library in order to be reused after considering the applicability of each part of the logic and events that are contained within the model. If they already exist, the analyst can integrate them only, instead of rewriting the complete model.

- **Architecture implementation:** The architecture defines the model structure. It contains the nodes basic structure (states, events, I/O) and the connection between them. The analyst is able to extract information about equipment and interfaces from design documents in order to build the different model elements. In a second step, nodes are linked together following the functional chains identified in the architecture. If the tool offers a graphical user interface, the analyst is able to reproduce the design model organization, its hierarchy and the nodes disposal. It may be difficult to have a faithful representation but it is important to keep similarities. It should reduce the risk of misunderstanding between system designers and safety analysts. Moreover, functional architecture and physical architecture may be defined separately, allowing starting safety assessment without having complete design but only a functional definition.
- **Failure modes selection:** With the failures modes available in the FMEA/FMES, the analyst is able to identify the different states of the system components and the events affecting them. The number of states is influenced by the expected granularity level as failure modes may be more or less refined, depending on the analysis type. Failure modes are also dependent of the studied failure conditions. For a given failure condition, some equipment may not be implied directly in the observed functional chain. It may acceptable not to integrate these failure modes in the model. In that case, it shall be documented as an assumption. However, model shall be exhaustive in order to be able to cover failures that may have an indirect effect on the observed system.
- **Low-level behaviour definition:** Model is only complete when system components internal behaviour has been implemented. Internal behaviour is expressed through transitions representing failure effects and assertions representing a transfer function between inputs, internal states and outputs. Dysfunctional behaviour will be mainly described but it is also interesting to implement part of functional behaviour, such as reconfigurations or functional states affected by failure modes. This step is the most critical one as safety analyst shall describe behaviour with the right level of abstraction needed for safety. Design documents contain generally more information than required with the risk to build a model too detailed and hardly maintainable. As a consequence, it may lead to numerous assumptions on the implemented behaviour.

At the end, the safety model becomes a failure propagation model representing the effects and the propagation of failures on the systems components. This model is not expected to be an exhaustive representation of the real system as it is not mandatory for safety assessment to implement every detail. The safety analyst shall explain why he can reduce the model perimeter by skipping non-relevant design information. These decisions are compiled under assumptions delimiting the model domain of validity. They shall be properly documented too in order to know under which perimeter results are considered valid.

5.4 Build the Failure Condition Logic

Failure conditions are modelled with an observer. An observer is a specific model component implementing a logical expression qualifying the feared event. The textual expression from the FHA shall be translated into this logical expression by combining model variables (states, outputs…). This observer has no interaction in the model. The model behaviour is consequently not disturbed by its presence.

5.5 Validation of the Safety Model and Failure Condition Logic

Model validation is not obvious. Safety assessment results are only valid if we are sure it has been performed on a model representative of the reality. It is not possible to take an assumption considering model may describe the real system behaviour. It shall be demonstrated relying on the MBSA language formalism. This formalism doesn't let place to interpretation. It has only a single meaning and it has to be in accordance with the real system one.

Safety analysts and designers are involved in this process, as in the document-based process. However, they can rely on a common formalism to ensure they have the same understanding of the system behaviour. The validation protocol shall take advantage of formal methods and not only be limited to a common review. MBSA tools offer a simulation feature, able to play scenarios and display results visually. Simulation is not sufficient to cover a complete test set. It shall be supported by a formal method to define this test set and used as verification mean. In case of inconsistencies, designers and analysts are able to check the model state and find whether it is a modelling error or a design error. Designers are involved to define the test case specification based on the real system behaviour and to check results. This definition is done according to state automatons. Designers catch the system real behaviour in automatons that can be compared with automatons generated from the model. Highlighted differences are then reviewed by both actors to define where errors are.

This step shall be associated with a validation plan covering configuration management, means of compliance and assumptions validation & verification. It is important to define clearly what shall be validated and how. It concerns especially assumptions that may impair the model validation results. Some of them may reveal not suitable to represent the real system, leading to inconsistent behaviour. They shall be validated and verified in order to ensure MBSA results validity. Configuration management helps to trace modifications done to models and validated them by non-regression tests with the previous validated issue.

This validation protocol concerns also failure conditions observers that shall be reviewed to ensure tool will generate correct results for each FC.

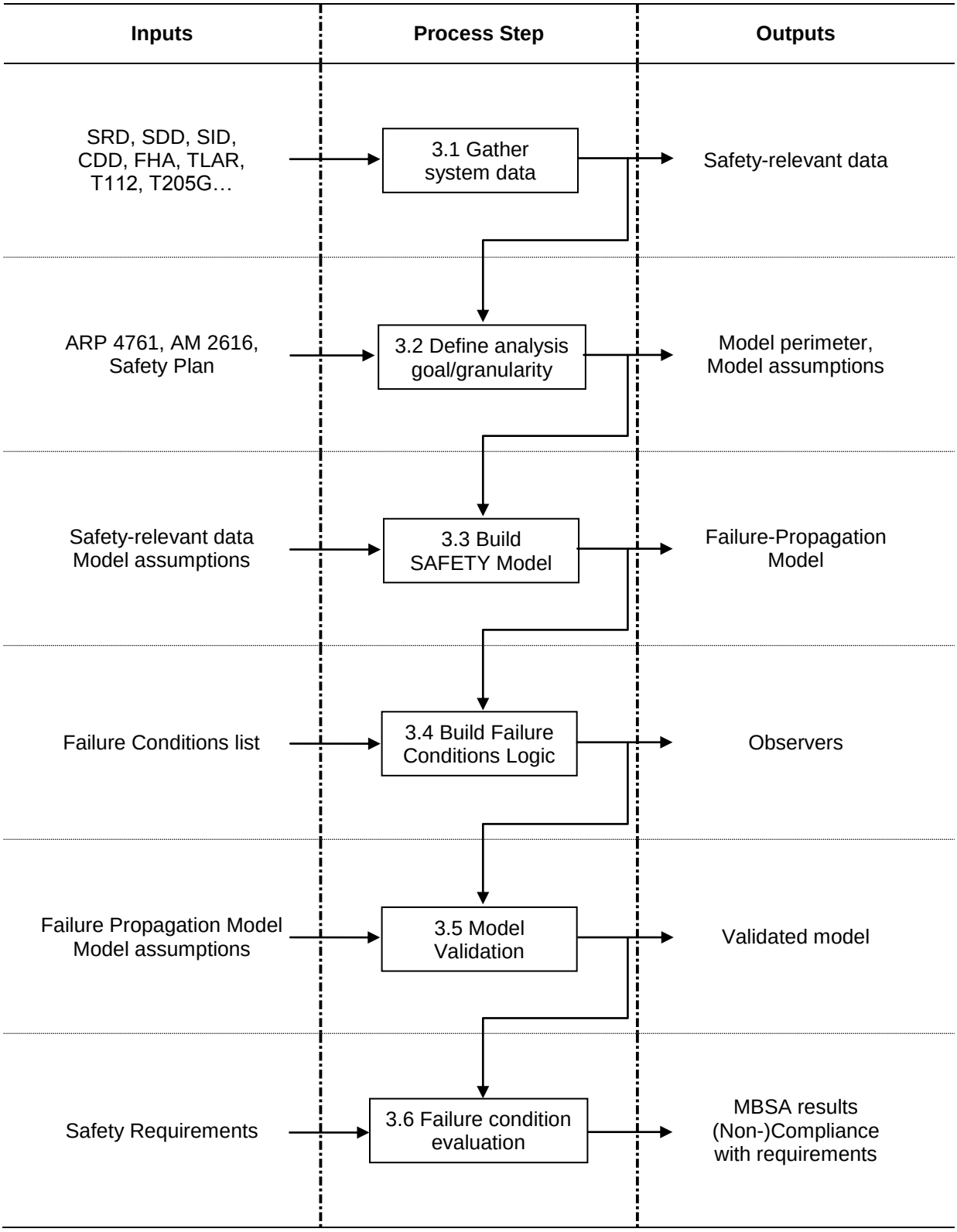
5.6 Failure Condition Evaluation and Analysis

Failure conditions are modelled by observers. Observers are nodes combining several variables in a logical expression. This expression becomes true when system state corresponds to the failure condition. In the current document-based methodology, each failure condition is assessed in a fault tree or a dependence diagram. These graphical constructions combine elementary failures to give an overview of the different combinations, also called cut sets, leading to the feared event. With MBSA tool, this step is reproduced by exploring the model state space and deducing the entire set of combinations for which the observer becomes true, the minimal cut sets. For the moment, results are mainly textual as there is no competitive tool able to reproduce a manually built fault tree.

These cut sets shall be post-processed to be exploited for safety assessment. With attributes and failure rates attached to events, qualitative analysis, quantitative analysis or common cause analysis can be performed. Depending on tool capabilities, limitations are existing for generating cut sets with a high order (order >4). The order represents the number of elementary failures contributing to a combination. Due to combinatorial explosion, the state space size increases exponentially with the requested order. These cut sets have generally an insignificant contribution to the failure condition

but it is not always the case. For those exceptions, it is possible to deduce them by simulation, playing a scenario manually identified to check if it contributes to the failure condition.

The loop is closed by completing safety documents and reporting safety assessment results to design office under the form of requirements or S/R parameters. If some parameters or requirements are still open, the model may be modified to assess them. In that case, the process restarts at step 1 to adapt the model.


Inputs
Process Step
Outputs
SRD, SDD, SID, CDD, FHA, TLAR, T112, T205G…
3.1 Gather system data
Safety-relevant data
ARP 4761, AM 2616, Safety Plan
3.2 Define analysis goal/granularity
Model perimeter, Model assumptions
Safety-relevant data
Model assumptions
3.3 Build SAFETY Model
Failure-Propagation Model
Failure Conditions list
3.4 Build Failure Conditions Logic
Observers
Failure Propagation Model
Model assumptions
3.5 Model Validation
Validated model
Safety Requirements
3.6 Failure condition evaluation
MBSA results
(Non-)Compliance with requirements

6 Tools and methods

The primary dataflow between tool chain bricks is sketched here and the potential IOS architecture to be developed is also identified in Figure 6-1.

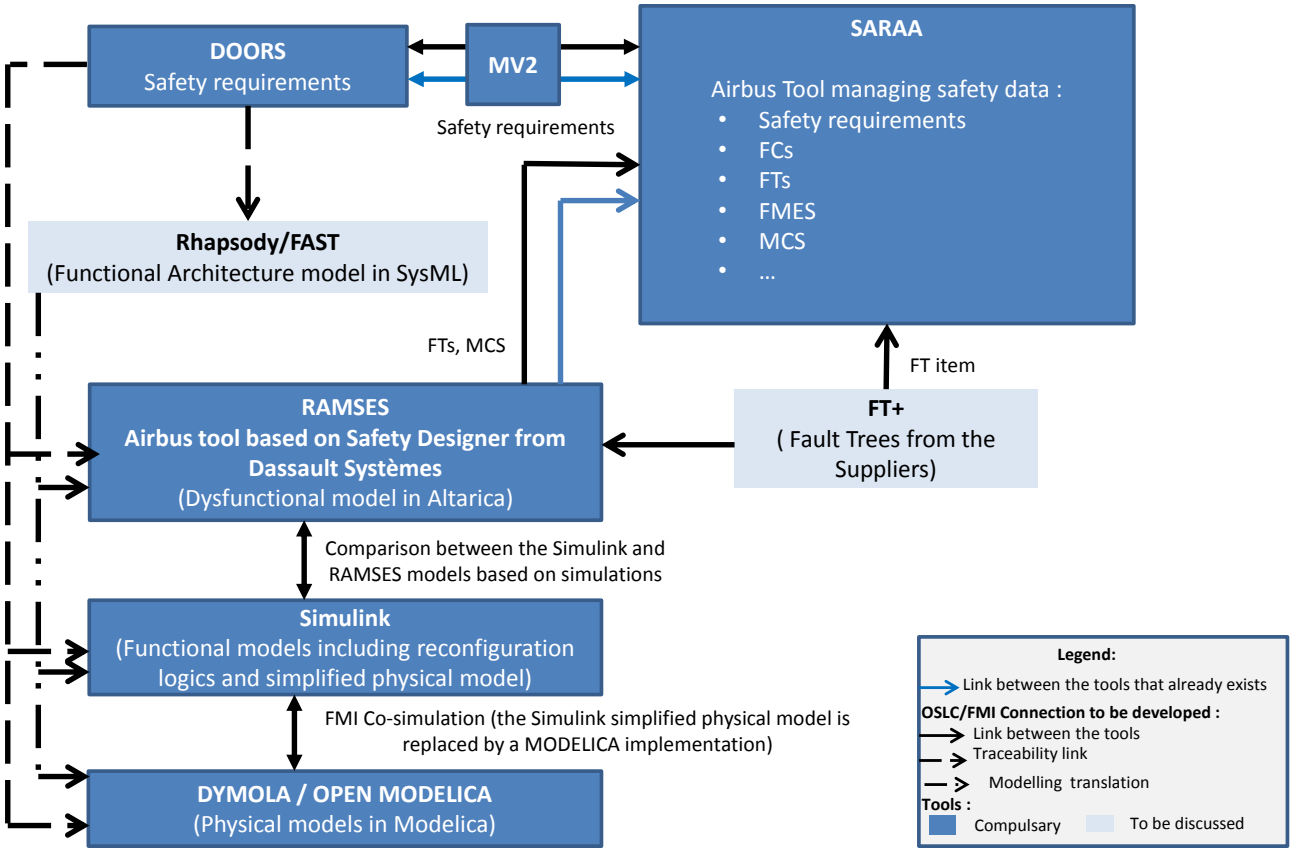


Figure 6-1: Primary Fuel Management Risk Analysis Tool chain, Dataflow and IOS

The functional and behavioural requirements as constrained by reliability and safety properties will be captured, analysed and managed within DOORS, SARAA, RAMSES and The Reuse company tools (if time and resources permitting) . Matlab/Simulink, Stateflow, SCADE Display and Dymola/Modelica will support the modelling and simulation of physical and control processes; Simulink will be used for continuous control laws modelling and refinement and possible future integration into SCADE Suite to exploit the SCADE FaultTree Analysis Manager. The IOS architecture will be targeted using IBM JAZZ platform to have the impact analysis on traceability features for the following Tool chain: DOORS, Rhapsody, Simulink, Dymola/Open Modelica. The simulations and co-simulations will be targeted via FMI platform.

6.1 Tools description

6.1.1 DOORS

DOORS is a well-known commercial tool sold by IBM whose aim is to support requirements based engineering activities. The DOORS offers the following features:

- requirements capture,
- traceability by linking requirements to design items, test plans, test cases and other requirements,
- requirements management in a centralized location for better team collaboration.

At Airbus, since the A380 programme, DOORS is used to manage aircraft requirements, whatever their types.

6.1.2 SARAA

SARAA (Safety And Reliability Analysis for Aircraft) is an Airbus tool supporting safety and reliability analysis for new aircraft designs in accordance with the standards agreed with the certification authorities (DGAC, FAA). The tool covers the development and documentation of Functional Hazard Analysis (FHA), Preliminary System Safety Analysis (PSSA), System Safety Analysis (SSA) and Common Mode Analysis (CMA). This includes both system level development of the safety case and aircraft level analysis and synthesis.

The tool organises safety analysis according to Aircraft and ATA chapter. The primary view is of a series of chapters in Microsoft Word supported by an information database. Most of the safety information is entered through a forms-based editor supported by navigation and browsing capabilities.

The reliability model includes calculation of dependency diagrams and fault-trees. This is accessed using graphic editors linked to the information model in the rest of the tool. Fault trees can be imported from the FaultTree+ tool (version 11.2) as well as entered through the graphic editor.

SARAA is a daily tool for safety teams.

6.1.3 SRMV2

It is self-contained V&V tool for safety requirements; it is also acting as interface to DOORS. The T205G and T105G databases (held in the SMRV2 tool) can allocate requirements to system requirement documents (including Specifications, SRDs, SIRDs, SWRDs). These are then copied into the folders within the DOORS.

6.1.4 RAMSES

RAMSES is an Airbus tool relying on Safety Designer from Dassault Systèmes. RAMSES is an Integrated Development Environment for the development and the analysis of safety models of systems, based on the AltaRica formal language. With RAMSES, one can create models and libraries of reusable components, observe the propagation of faults by raising events in a dedicated step-by-step simulator, and perform several calculations to assess the modelled systems.

The main RAMSES capabilities are:

- Graphical model editor: Edit AltaRica models through drag & drop, tables or text editor; organize models in libraries for future re-use.

- Step-by-step simulator: Simulate the propagation of faults on AltaRica models, by specifying initial configurations and raising events at will.
- Compiler to fault trees: Automatically generate fault trees from AltaRica models, specifying top events and initial configurations.
- Generate the minimal cut-sets of the recreated scenarios
- Generate the Minimal Sequence Set
- Compute the general and cut-sets quantification
- FMEA assistant: Automatically generate drafts of FMEA.
- Report generator: Generate reports in the DocBook, RTF, XML file formats.

RAMSES is not used operationally yet, but is a key enabler for safety R&T projects.

6.1.5 SIMULINK

Simulink® is a graphical environment for multidomain simulation and Model-Based Design produced by The Mathworks. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink provides a set of predefined blocks that enables the design engineer to create detailed block diagrams of the system. Tools for hierarchical modelling, data management, and subsystem customization allow the engineer to represent even the most complex system concisely and accurately.

The simulation can be run interactively from the Simulink Editor or systematically from the MATLAB command line. The following simulation run-time modes are available:

- Normal (the default), which interpretively simulates your model
- Accelerator, which increases simulation performance by creating and executing compiled target code but still provides the flexibility to change model parameters during simulation
- Rapid Accelerator, which can simulate models faster than Accelerator mode by creating an executable that can run outside Simulink on a second processing core

After running a simulation, the simulation results can be analysed in MATLAB and/or Simulink. The full set of powerful Matlab functions is available to perform detailed analysis of the results of simulation.

A typical Simulink diagram models the flow of data (signals) from the inputs through a series of blocks to the outputs.

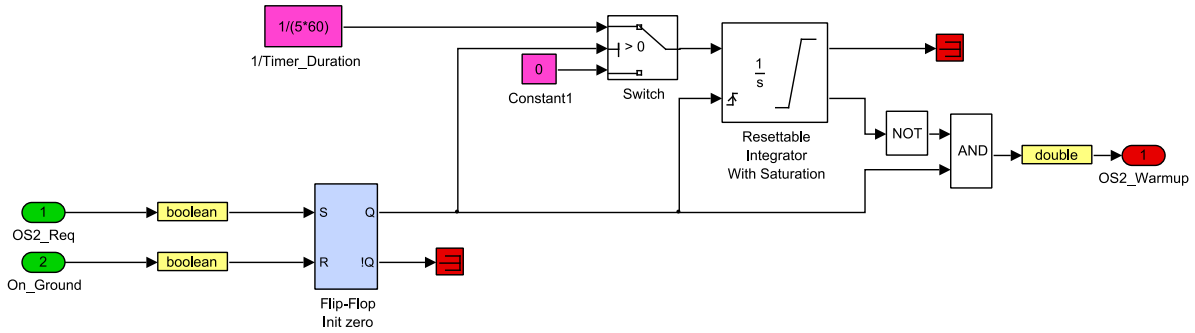


Figure 6-2: Example Simulink Block Diagram

At each simulation step (be it continuous or discrete-time simulation), the outputs of each block are computed based on the inputs at the previous time step. For some blocks which have internal states (e.g. the integrator block), an extra “minor-time step” calculation is performed to allow the blocks to update their internal states.

When a continuous time simulation is performed, the simulation time advances based on the system dynamics. When the inputs and/or internal states of the blocks are changing very only gradually then the time-step can be quite large, but when the states are changing rapidly the time-step is reduced so that the simulation can capture the system dynamics.

During a discrete-time simulation, simulation time advances at a fixed rate. Care must be taken, therefore, to ensure that any important dynamics (such as zero-crossing or other discontinuities) are not missed.

6.1.6 STATEFLOW

Stateflow® is an environment for modelling and simulating combinatorial and sequential decision logic based on state machines and flow charts. Stateflow lets you combine graphical and tabular representations, including state transition diagrams, flow charts, state transition tables, and truth tables, to model how your system reacts to events, time-based conditions, and external input signals.

You can model the different components in your system as states that execute exclusively or in parallel. Stateflow lets you manage the complexity of your design by organizing state diagram objects, functions, and components hierarchically.

In Stateflow you can represent combinatorial logic graphically with flow charts and in tabular format with truth tables. Designing logic involves defining conditions to be checked and subsequent actions to be performed.

With Stateflow you can design logic for supervisory control, task scheduling, and fault management applications. Stateflow includes state diagram animation and static and run-time checks for testing design consistency and completeness before implementation.

The key features are:

- Modeling environment, graphical components, and simulation engine for modeling and simulating complex logic
- Deterministic execution semantics with hierarchy, parallelism, temporal operators, and events
- State diagrams, state transition tables, and state transition matrices representing finite state machines
- Flow charts, MATLAB functions, and truth tables for representing algorithms
- State diagram animation, state activity logging, data logging, and integrated debugging for analysing the design and detecting run-time errors
- Static and run-time checks for transition conflicts, cyclic problems, state inconsistencies, data-range violations, and overflow conditions
- Mealy and Moore notations for finite-state machines

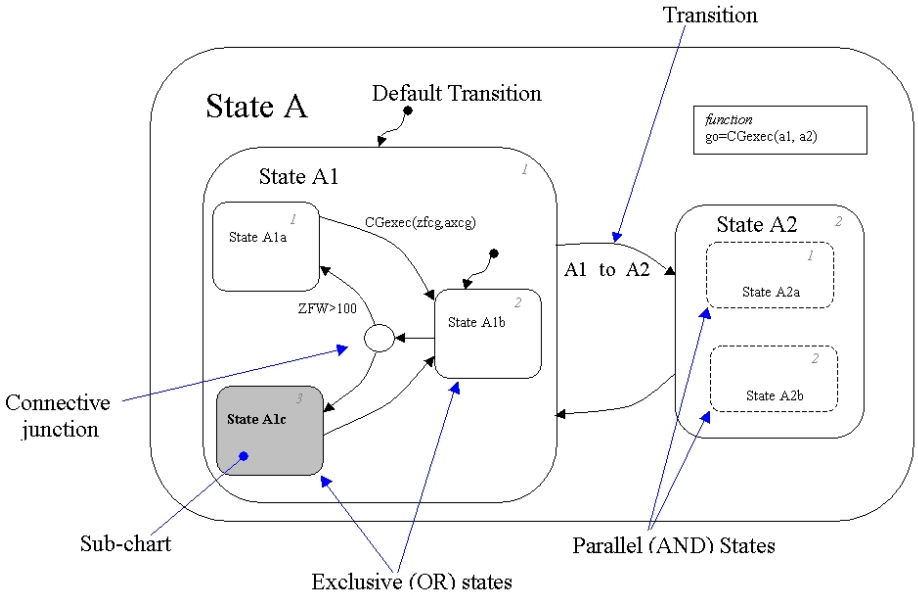


Figure 6-3: Example Stateflow State-Transition Diagram

An overview of the standard notation is shown Figure 6-3. The terms used in this notation are defined as follows:

State - They are presented as rectangles with rounded corners. States are labelled with a name and may contain a number of action statements each separated or terminated by a semicolon.

Sub-charts - A state chart embedded within a higher level state chart is known as a sub-chart. Sub-charts are shown as a labelled state with a shaded grey background and may contain any of the elements and notation that a higher level chart may contain, e.g. states, transitions, actions. Sub-charts allow a complex chart to be reduced to a set of simpler, hierarchically organised diagrams. This makes the overall system easier to understand. In Figure 6-3 state A1c is a sub-chart. Sub-charts are used extensively within this document.

Child Objects - any object that sits within a higher level object, and hence can only execute if the parent is active. In Figure 6-3 state A1, state A2 and the function go are all child objects of state A.

Transition - the occurrence of an event causes a transition from one state to another (denoted by the syntax ***[condition]***).

Transition Action - the action carried out during a transition from one state to another (denoted by the syntax/expression). Note that a transition action executes only if the entire transition path is valid (when the origin state is exited, immediately before the destination state is activated).

Default Transition - represented by the symbol below shows the default sub-state on entry into a given state chart. For example, in Figure 6-3 on entry into State A, sub-state A1 will be the initial default state.

Exclusive (OR) State - represented by states with solid borders. These states are mutually exclusive. Only one state can be active at a time.

Parallel (AND) State - represented by states with dashed borders. All states at the same level can be active at the same time. The activity within parallel states is independent.

Connective junctions or decision points - denoted as a circle connecting two or more transitions. The transition taken out of the junction, and therefore the next state to be actioned, is dependent on the evaluation of each state's transition label. Where a transition out of a junction has no associated function then that transition maybe considered to be the 'else' clause of an 'if-else-endif' construct, so for example, in Figure 6-3 the connective junction will enter state 'A1a' if ZFW>100 and state 'A1c' if ZFW <=100.

Hierarchy is formed when a state contains second-level states. The parent is referred to as 'super-state' while the lower level states are called sub-states. There is no limitation on how many levels of hierarchy can be specified.

Child objects including sub-states can only execute or activate if the parent state is active. When a super-state becomes inactive, the active sub-state has to exit as well.

Note, that 'high level' transitions have priority over 'low level' transitions, so, for example, in Figure 6-3 the transitions (and states) within state A1 are pre-empted if transition 'A1_to_A2' occurs.

Transition testing always starts from the highest level active super-state and moves inward to its active sub-states, therefore 'High level' transitions have priority over 'low level' transitions.

For example, if state A1b is active and transition 'A1 to A2' becomes true, the following sequence occurs; Exit of state A1b, Exit of state A1, Entry of state A2. Parallel states A2a and A2b are both activated before transition 'A2 to A1'.

Function state - They are presented as rectangles with square corners. These states may be used to define complex arithmetic functions. The rectangle is the formal definition of the function and includes the word 'function', the function's name and a list of its required parameters. The function's algorithm is defined 'underneath' the formal definition; in this respect it is the same as a sub-chart. Functions defined in this way are typically used on transitions, for example the CGexec function on Figure 6-3

6.1.7 DYMOLA

Dymola is a design, modelling, and simulation solution for complex systems, based on the Modelica language. Dymola enables the definition and optimization of dynamic behaviour and complex interactions thanks to a simple and practical model creation interface, using a symbolic digital solver for complex models.

The tool is sold by Dassault Systèmes. It was assessed by Airbus but it is not operationally used.

6.1.8 MODELICA

Modelica® is a non-proprietary, object-oriented, equation based language to conveniently model complex physical systems containing, e.g., mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents.

Models are described by differential, algebraic, and discrete equations, and can be built using various graphical editor environments;

- Icons represent physical components. (electrical resistance, mechanical device, pump, ...)
- A connection line represents the actual physical coupling (wire, fluid flow, heat flow, ...)
- A component consists of connected sub-components (= hierarchical structure) and/or is described by equations.
- By symbolic algorithms, the high level Modelica description is transformed into a set of explicit differential equations:

The standard capabilities of Modelica can be enhanced by the use of additional libraries that are available both free and commercially.

Modelica is a language produced and maintained by The Modelica Association (http://www.modelica.org). There are several tools, both free and commercial that support this language – and make it easier to build models via graphical building techniques.

Commercial Tools include:

- LMS AMESim
- MapleSim by MapleSoft
- SimulationX by ITI GmbH
- Dymola by Dassault Systemes

Free Modelica Simulation Environments include:

- JModelica.org (Modelica Simulation and Analysis Suite)
- Modelicac (Modelica Compiler)
- OpenModelica (developed by the Open Source Modelica Consortium – OSMC – to produce a complete Modelica modeling, compilation and simulation environment).

An example OpenModelica physical model is given in the following screenshot. It is produced by the OpenModelica Connection Editor (OMEdit)

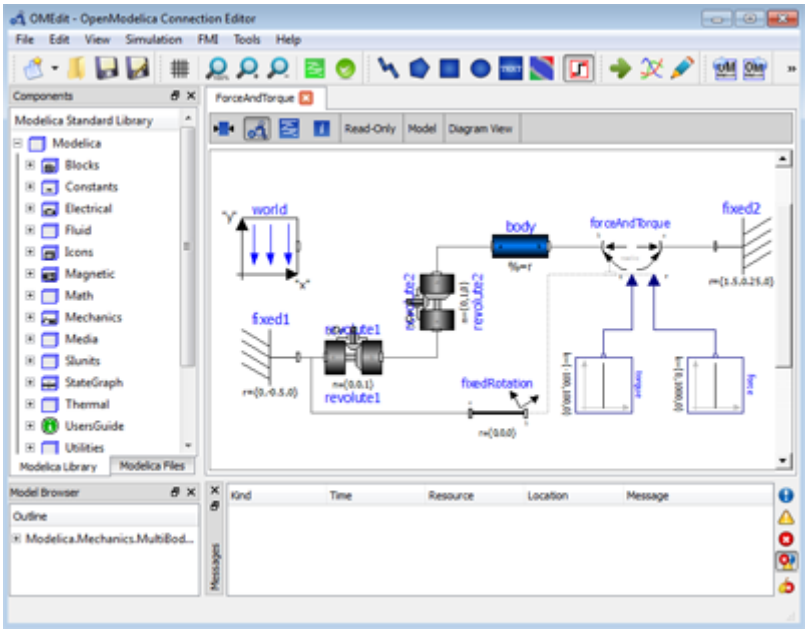


Figure 6-4: OpenModelica Screenshot

It is the intention of CRYSTAL project to focus the physical modelling development using the open source OpenModelica toolset. It is the considered opinion of the authors that the available commercial solutions provide their own “enhancements” and non-standard additions to the Modelica language which will make the developed models less portable.

6.1.9 SCADE

SCADE (by ANSYS) covers the full development cycle of critical embedded software from specifications to the generation of correct by-construction production code in C and Ada.
It supports both data flow and control logic type of applications.
It is the only commercial automatic code generation tool qualified to the strictest level of the civilian avionics standard RTCA DO-178C, Level A.

SCADE is used to implement the detailed design of Airbus critical avionics systems (e.g. flight Control system and flight warning system).

7 Modelling and Simulation

7.1 RAMSES tool for Model Based Safety Analysis

Reliability Availability Maintainability and Safety Environment for Simulation (RAMSES) is a tool based on the AltaRica Data-Flow formal language. It was developed to create graphical models of highly integrated systems, and to use those models to perform model base safety analysis.

7.1.1 Fuel System Model in RAMSES

The Fuel System model includes the ATA28 (Fuel System) functional block and those systems in the boundary. The interfaces represented are electronic and electrical interfaces. The systems included in the model are:

- Cockpit and Display System (ATA31A)
- Integrated Control Panel (ATA31C)
- Avionics Data Communication Network (ATA42B)
- Electrical Power Distribution Centre (ATA24A)
- Flight Warning System (ATA31B)

The following picture represents the Fuel System model layout in RAMSES:

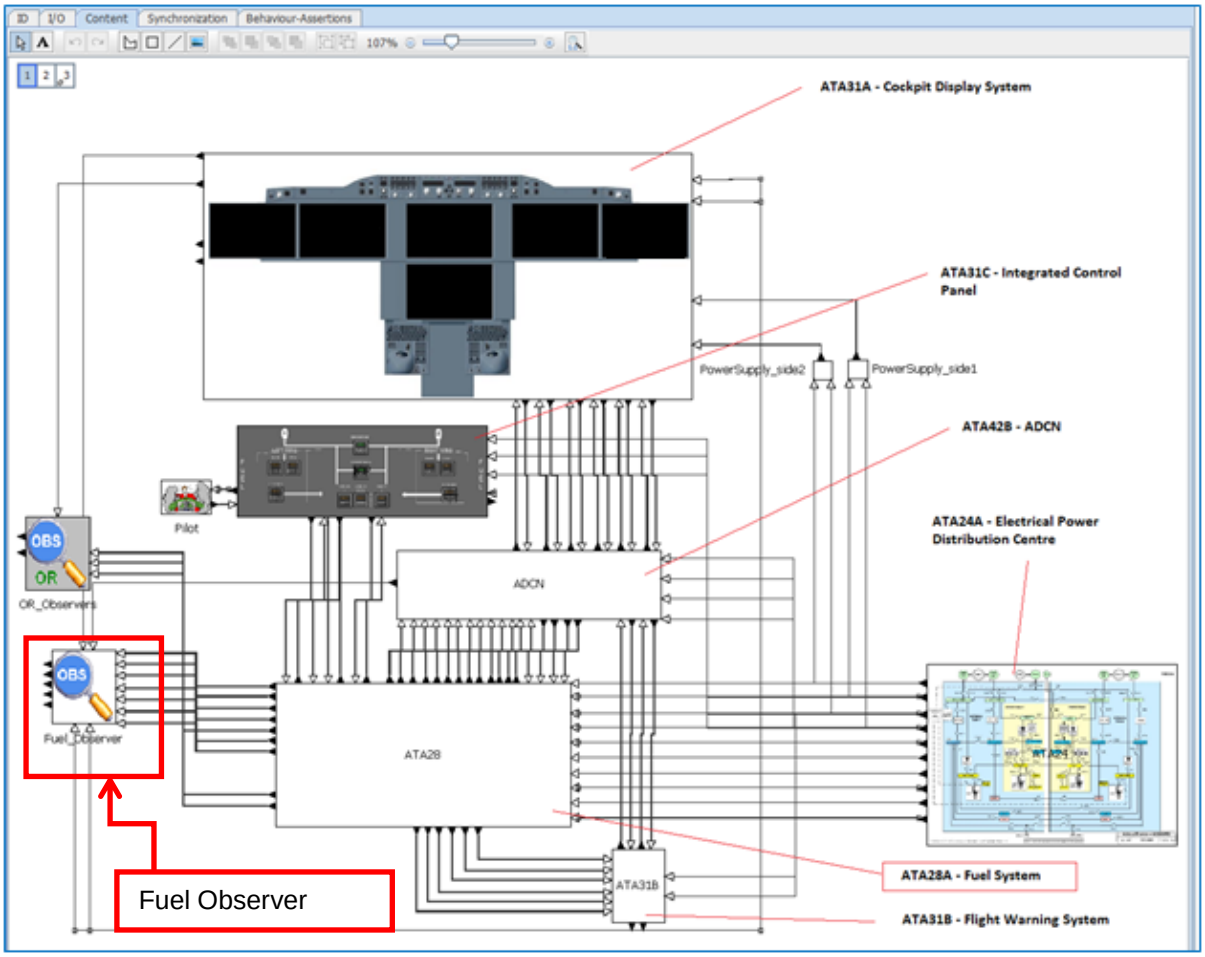


Figure 7-1: Fuel System model layout in RAMSES

7.1.2 Observers

As in the traditional safety analysis, the system needs to be shown to be compliant to the safety objectives that are set against the various failure conditions that may occur. This generally definition of a failure conditions is captured in the top part of a fault tree, with the lower parts of the fault tree representing the systems architecture. For MBSA, the architecture is built into the model and so only the top part of the fault tree that represent the FC need to be constructed. In the MBSA approach this is referred to as the Observer. The following diagram shows the list of FC's that have been defined for this study and shows how the top part of the Fault tree logic can be captured. So for FC01 = total loss of provision of fuel to both engines, basically the equation for the FC is

not(info_FC01^fuel_Supply_eng1) and not(info_FC01^fuel_Supply_eng2)

The definition of Observers, or FC's can be as simple as that. It can be constructed as a composite node in the same way that any other part of the model is constructed. It can get more complicated when defining FC's that are only applicable during certain flight conditions such as, during take-off, during landing, in Cruise, etc.

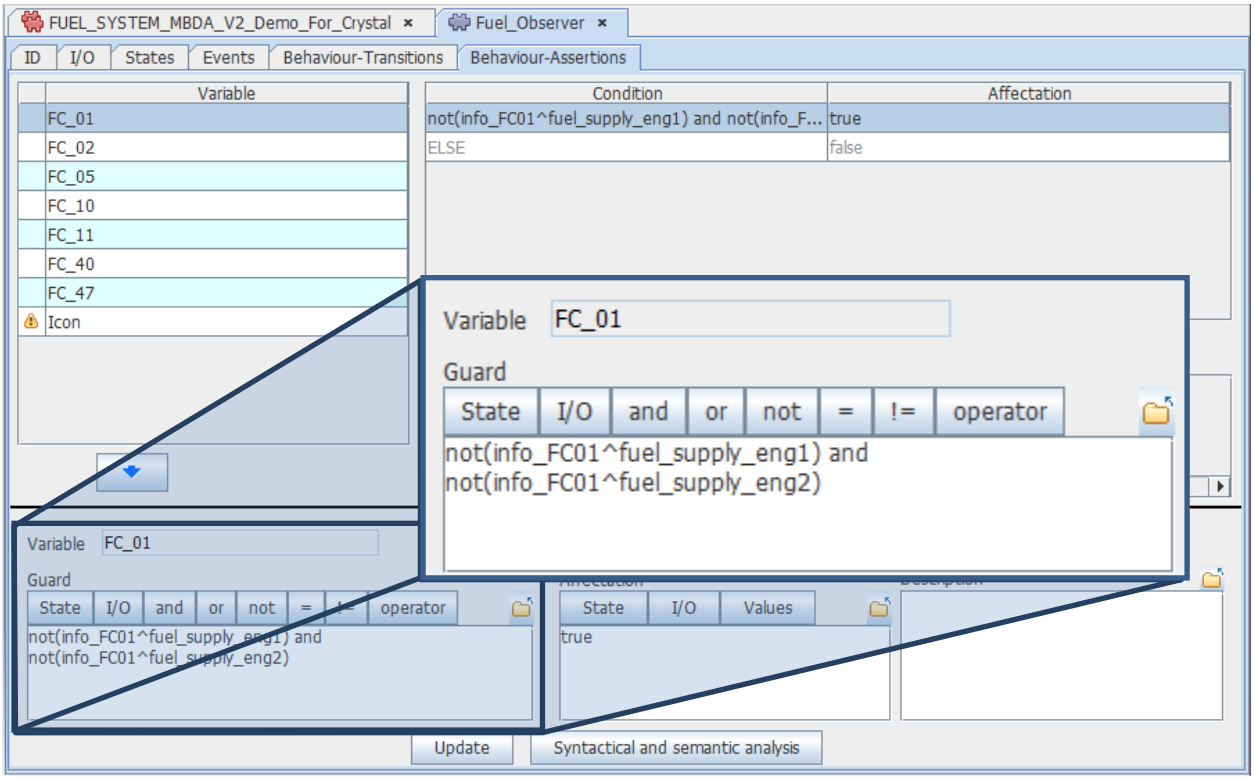


Figure 7-2: List of FC's defined in the fuel systems FLM and the definition of FC01

7.1.3 ATA 28 (Fuel System) Functional Block

The ATA28 functional block includes all the equipment within Fuel System and their interactions (inputs & outputs). The model includes the electrical power supply, discrete signals and data exchange within the FQMS. As a summary, the items included in the model can be listed as follows:

- TWDCs data exchange (including discrete independent high level signals to FWS*).
- CPIOM data exchange (including AFDX inputs from ATAs 70A, 32A, 34A and 27B) for fuel management purposes.
- CPIOM command inputs (Valves and Pumps).
- Integrated Refuel Panel interface.
- Equipment power supplies.

The following picture shows the general ATA28 block internal arrangement:

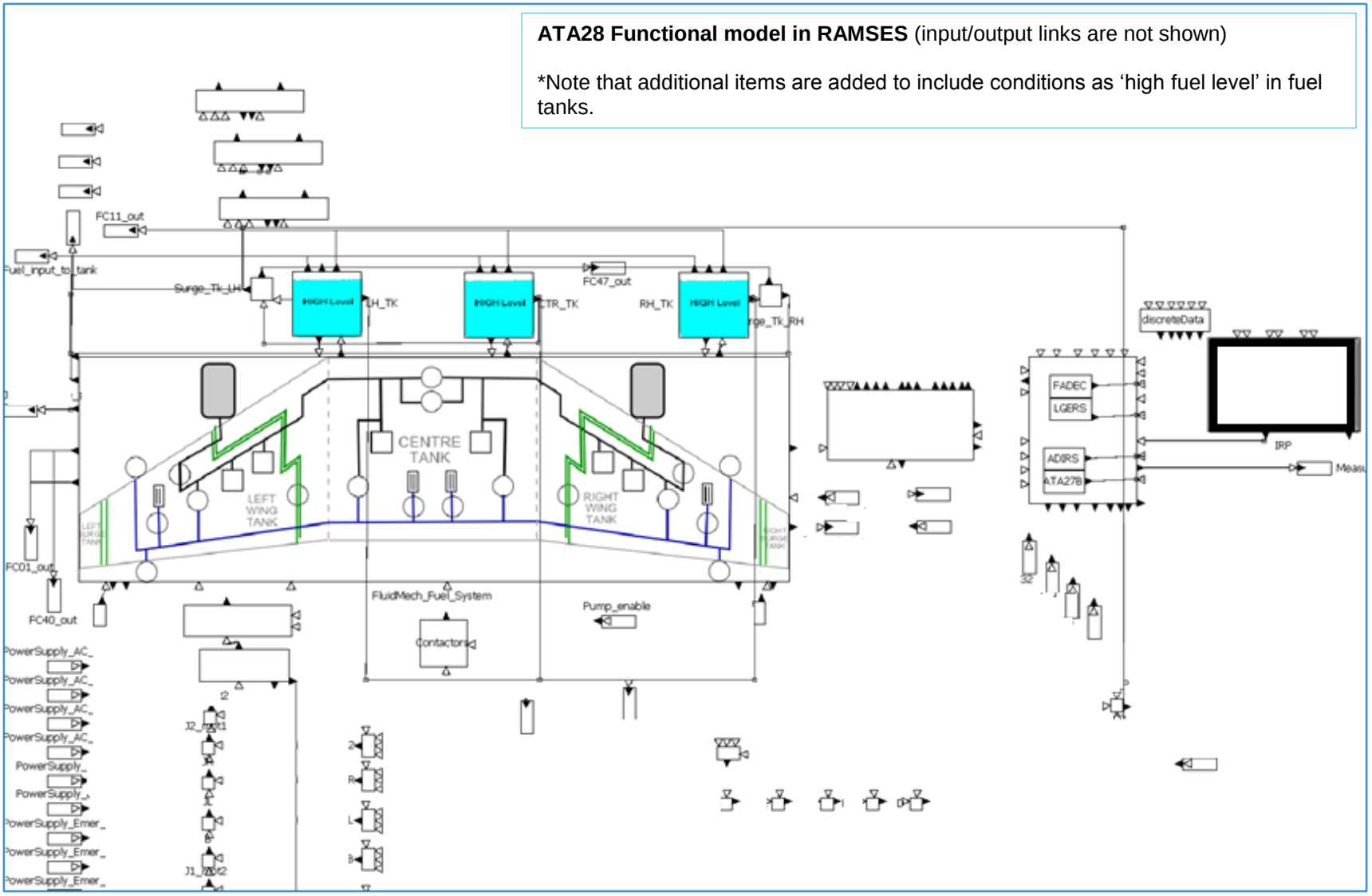
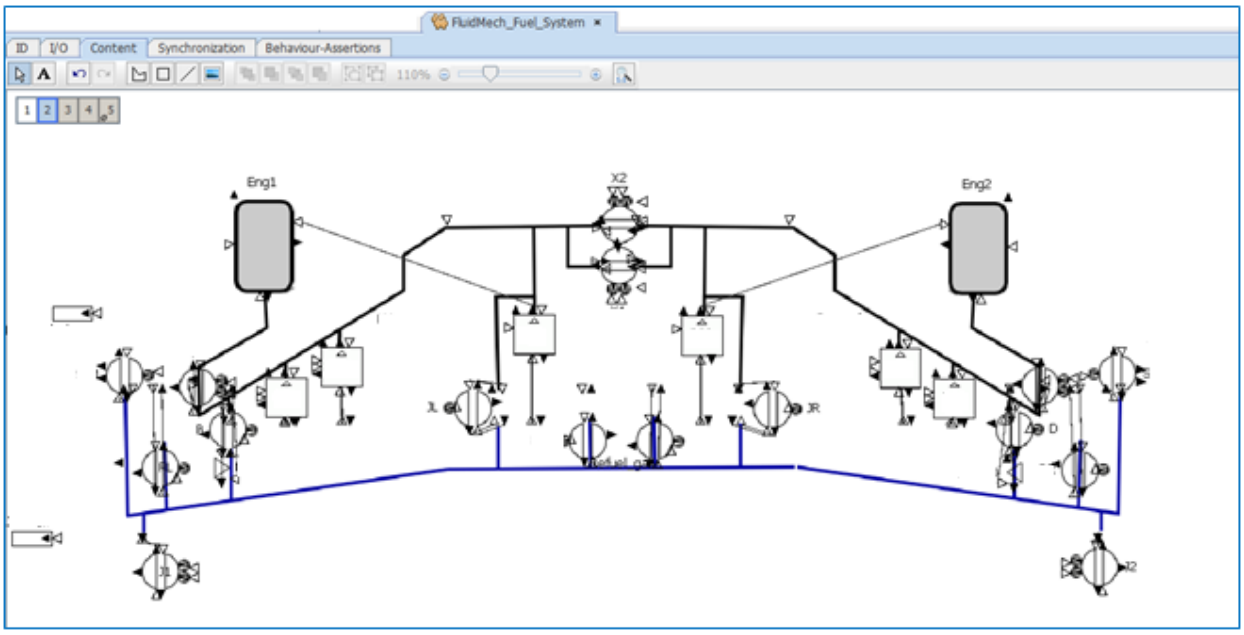


Figure 7-3: ATA28 Functional model in RAMSES

7.1.4 Fluid & Mechanical Block

This block includes all the fluid & mechanical equipment in the fuel tanks.



Note: Input/outputs links are not shown

Figure 7-4: Fluid Mechanical Blocks Modelled in RAMSES

7.1.5 Simulation and Model Analysis

The simulation functionality of the tool is used during model construction to show that the behaviour of the model is as expected and can be performed on an atomic node level, composite node level or for the complete model.
Likewise once it is possible to perform various proofs to show that the model behaviour is as expected as you construct the model and finally once all the Observers are constructed in order to perform your safety cut-set analysis. The tool allows you to perform much more complicated analysis, such as cut-sequence analysis that also considers the sequence behaviour of the logic within the model, but for simplicity the discussion here will only cover cut-set analysis.

7.1.6 Failure Scenario Simulation

A simulation is started by pressing the green traffic light button on the top left of the Ramses window.

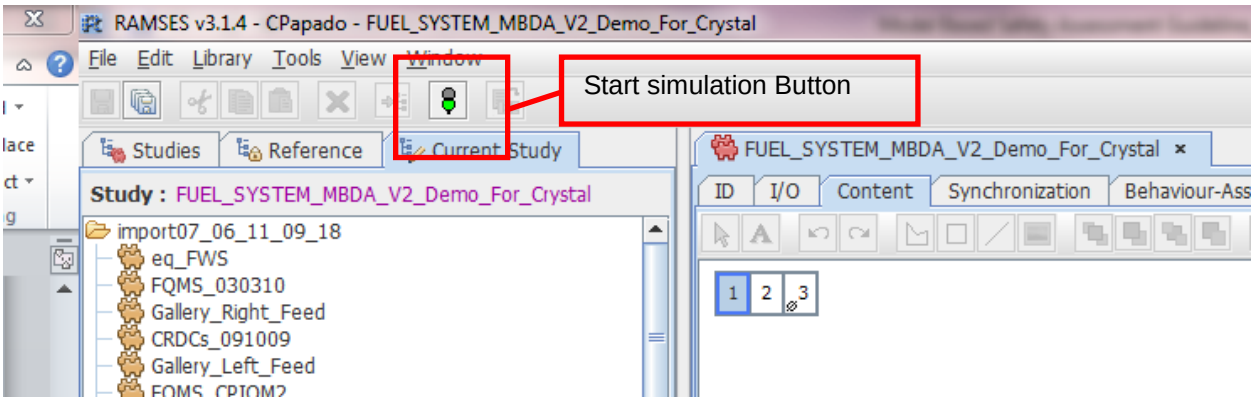


Figure 7-5: Command bar of the RAMSES tool

The traffic light signal changes to red and can then be used to stop a simulation.

The simulation window shows an interactive synoptic of the fuel system that can be interacted with to inject failures. See the figure below. The simulation tab on the right of the screen can also be used to inject failures.

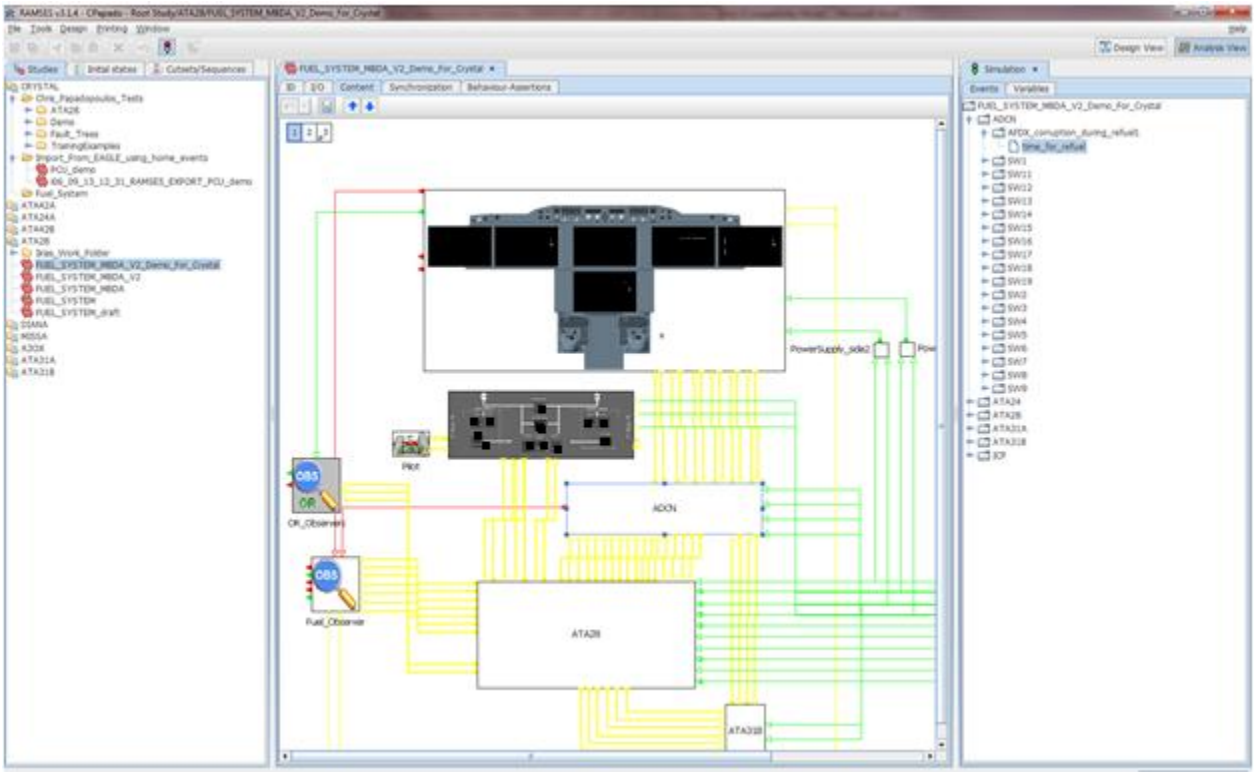


Figure 7-6: Fuel System simulation before injecting a corruption fault during refuel

The following figure shows the Fuel System simulation after injecting a corruption fault during refuel. Note the observer node shows some of the FC's have become falsefied (gone from green to red)

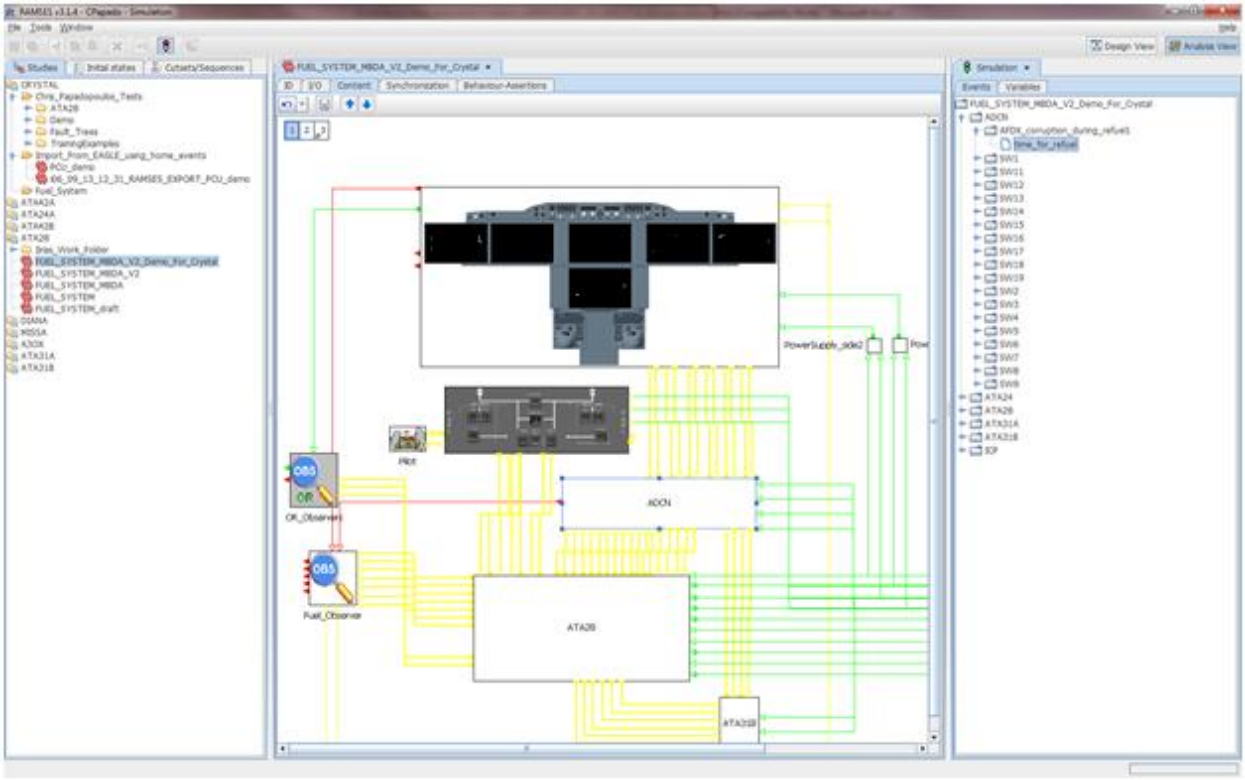


Figure 7-7: Fuel System simulation after injecting a fault

It is possible to step back to undo an injected failure

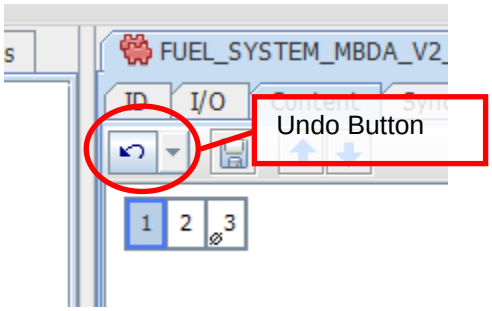


Figure 7-8: Undo Button

It is also possible to select the variables tab on the simulation pane on the right of the display to look at all the input, output and private variable values.

It is possible to save a failure scenario that has been played and to replay it. Once saved the failure scenario can be used to perform an analysis.

This can be used to play out dispatch conditions from the MEL to then see if the qualitative or quantitative safety objectives are still met.

7.1.7 Safety Analysis

The tool can be used to generate two types of results, cut-sets and flat fault trees. Cut-sets are the exhaustive list of combinations of failure events that lead to the loss (falsification) of a feared event (Failure condition or Observer).

The following diagram shows the configuration of the study.

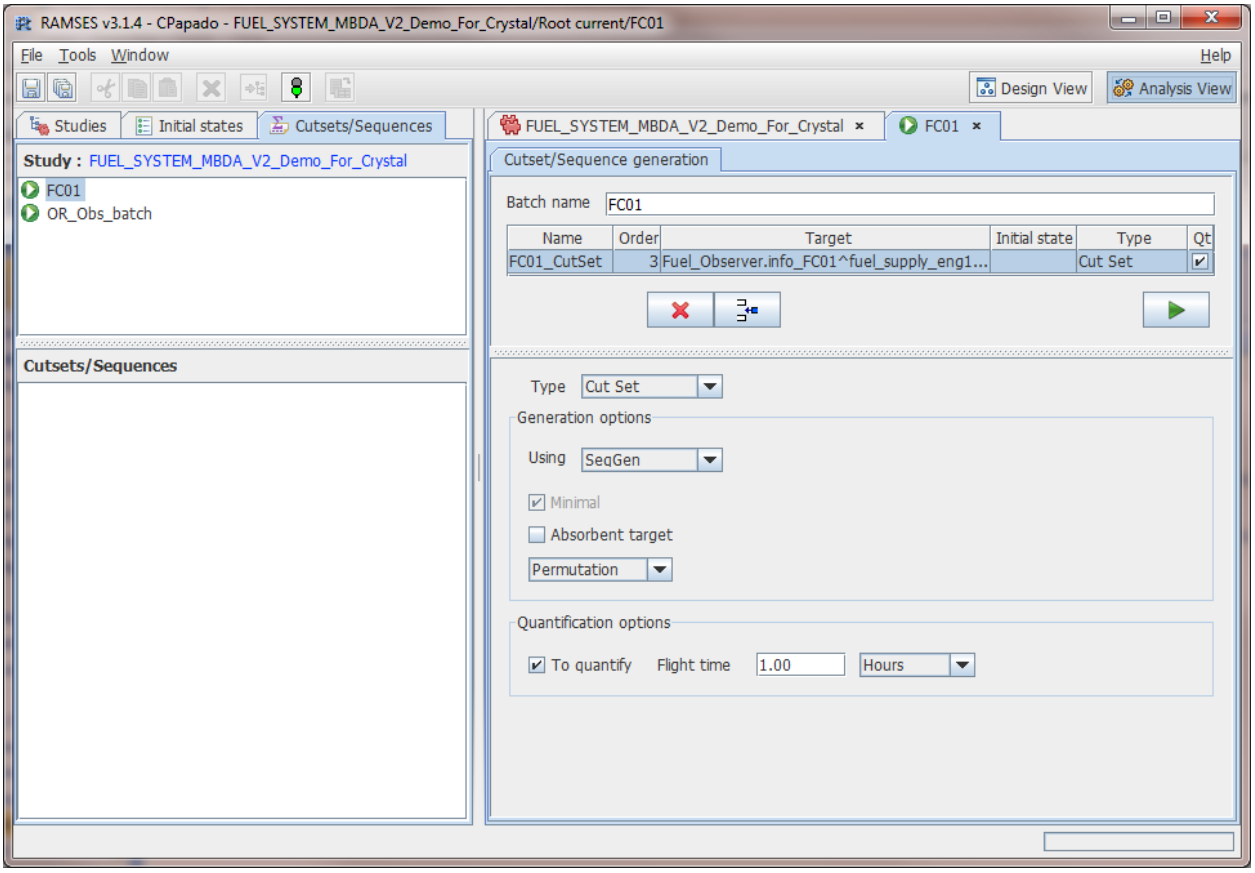


Figure 7-9: RAMSES Configuration Window

This is saved as FC01 under the study window. The study configuration contains the following

- The name of the Failure Condition that is being analysed,
- The order up to which the analysis will be performed,
- The target observer within the model, which is the actual observer that is being analysed,
- The analysis is looking for conditions where the observer becomes true,
- There is no initial state selected.
- The analysis type is cut-set.

In the analysis algorithm settings:

- The sequence generator engine is selected,
- The analysis is configured for permutation.
- The option to quantify is selected.
- The Flight time is set to 1 hours

Starting the study puts the study into the Analysis Queue. The screen capture below shows the information that is presented.

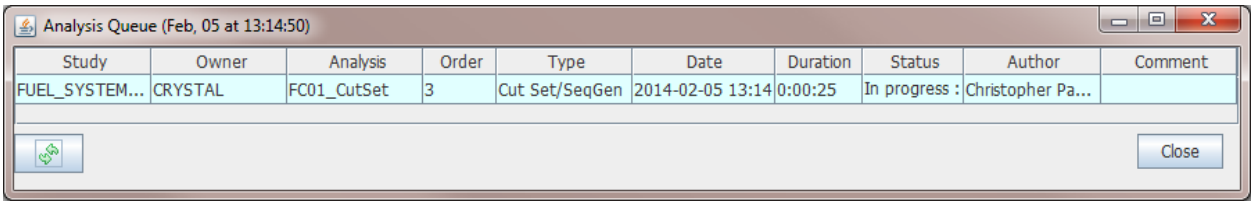


Figure 7-10: RAMSES Queue/Progress Window

Once the analysis is complete it is necessary to refresh the calculation tree which then provides a link to the analysis results. It is possible to view the results in a number of ways, as a flat fault tree, as a list. It is possible to animate a cut-set by selecting it, the nodes that are members of the cut-set are highlighted by the tool.

The Fuel Systems model still in the process of being developed and so a simple fault tree model (see the following figure) is used as an example for showing the results that are generated.

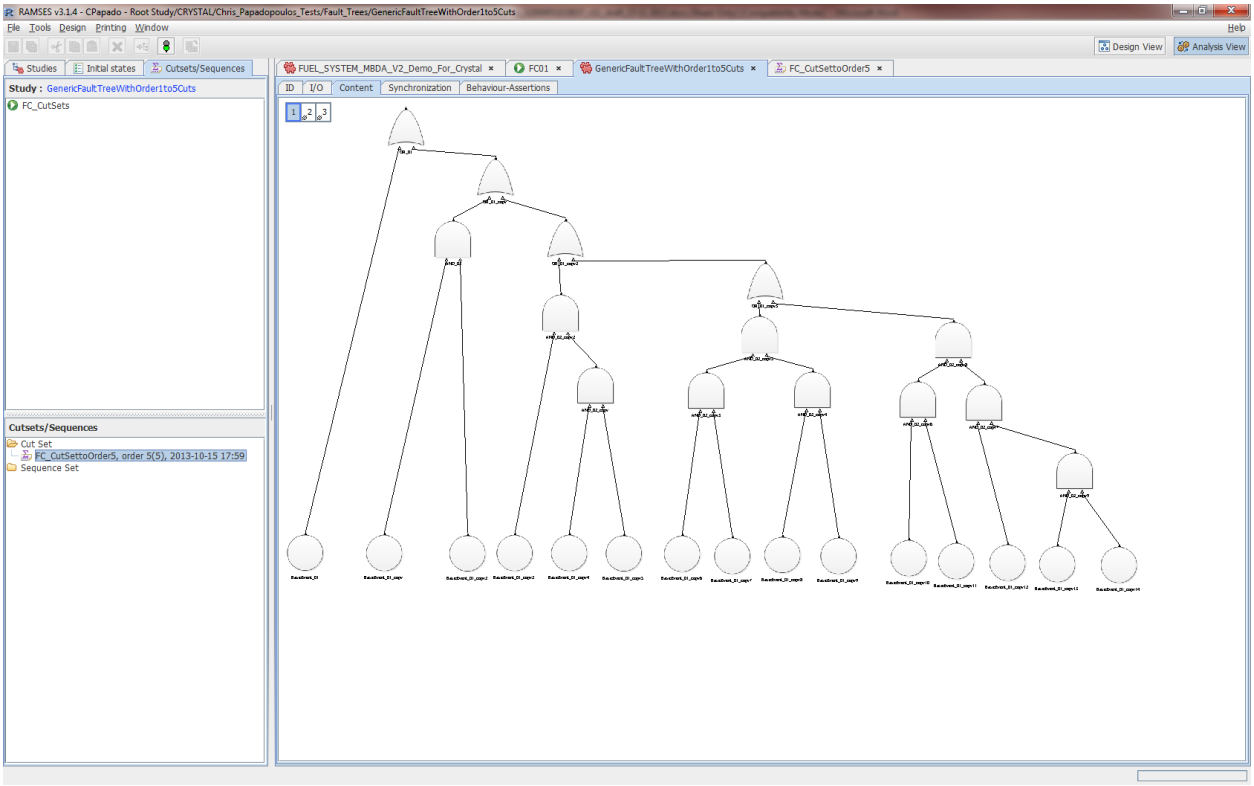
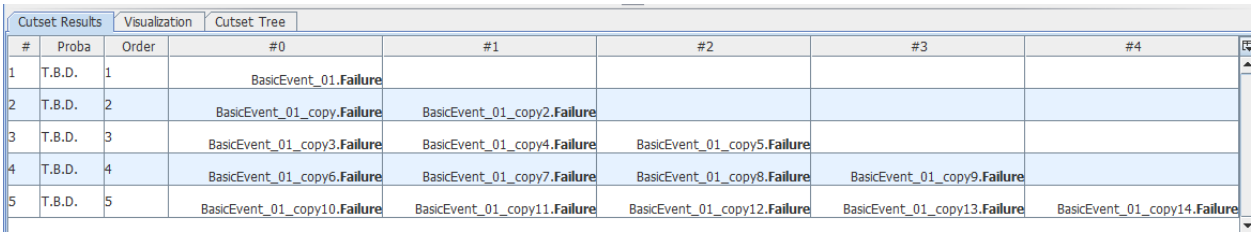


Figure 7-11: Fault Tree model with 5 cut-sets from order 1 to order 5

The analysis configuration process is the same. Starting the analysis is the same as described for the fuel systems model. The cut-sets / sequences pane on the bottom left shows the link to the analysis results. Selecting this shows the results in a number of ways:

1. As a list (see Figure 7-12)

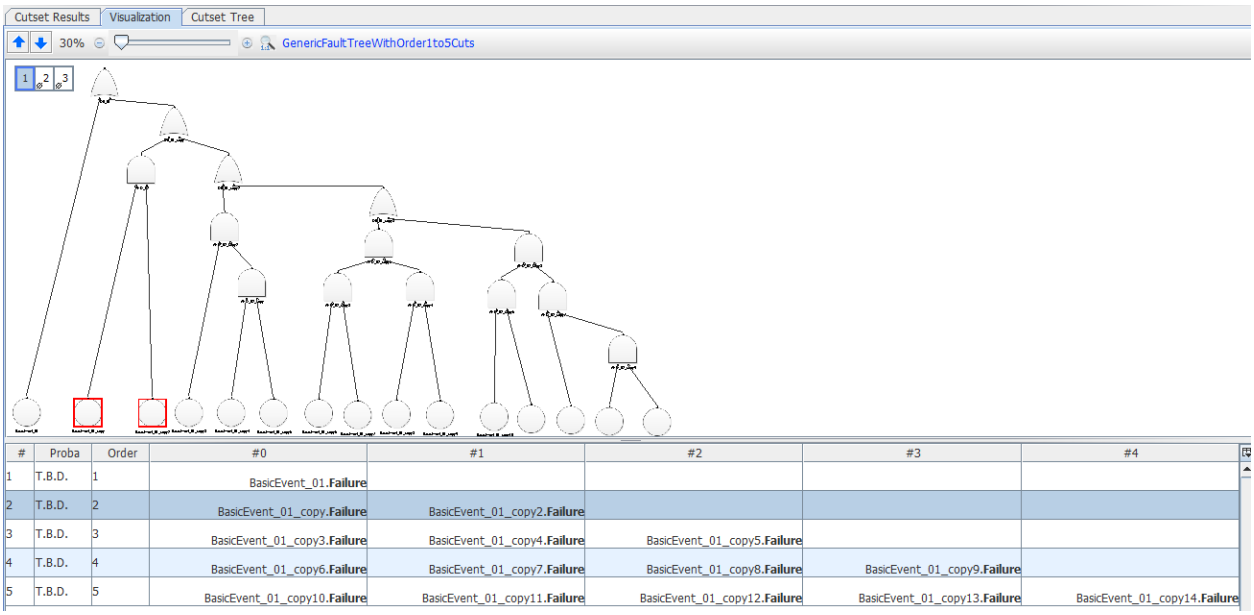
2. With a visualization of any selected cut-set on the model (see Figure 7-13)
3. As a flat fault tree (see Figure 7-14)



Cutset Results | Visualization | Cutset Tree

#	Proba	Order	#0	#1	#2	#3	#4
1	T.B.D.	1	BasicEvent_01.**Failure**				
2	T.B.D.	2	BasicEvent_01_copy.**Failure**	BasicEvent_01_copy2.**Failure**			
3	T.B.D.	3	BasicEvent_01_copy3.**Failure**	BasicEvent_01_copy4.**Failure**	BasicEvent_01_copy5.**Failure**		
4	T.B.D.	4	BasicEvent_01_copy6.**Failure**	BasicEvent_01_copy7.**Failure**	BasicEvent_01_copy8.**Failure**	BasicEvent_01_copy9.**Failure**	
5	T.B.D.	5	BasicEvent_01_copy10.**Failure**	BasicEvent_01_copy11.**Failure**	BasicEvent_01_copy12.**Failure**	BasicEvent_01_copy13.**Failure**	BasicEvent_01_copy14.**Failure**

Figure 7-12: Cut-Sets result file for the simple fault tree model shown in Figure 7-11



#	Proba	Order	#0	#1	#2	#3	#4
1	T.B.D.	1	BasicEvent_01.**Failure**				
2	T.B.D.	2	BasicEvent_01_copy.**Failure**	BasicEvent_01_copy2.**Failure**			
3	T.B.D.	3	BasicEvent_01_copy3.**Failure**	BasicEvent_01_copy4.**Failure**	BasicEvent_01_copy5.**Failure**		
4	T.B.D.	4	BasicEvent_01_copy6.**Failure**	BasicEvent_01_copy7.**Failure**	BasicEvent_01_copy8.**Failure**	BasicEvent_01_copy9.**Failure**	
5	T.B.D.	5	BasicEvent_01_copy10.**Failure**	BasicEvent_01_copy11.**Failure**	BasicEvent_01_copy12.**Failure**	BasicEvent_01_copy13.**Failure**	BasicEvent_01_copy14.**Failure**

Figure 7-13: Visualization of second order cut-set

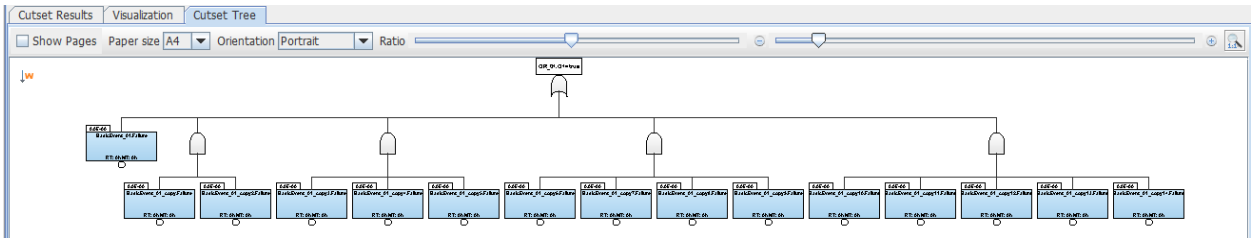


Figure 7-14: Cut-set Fault Tree View

7.2 Functional Modelling Methodology

A classic feedback “Plant/Controller” setup is employed;

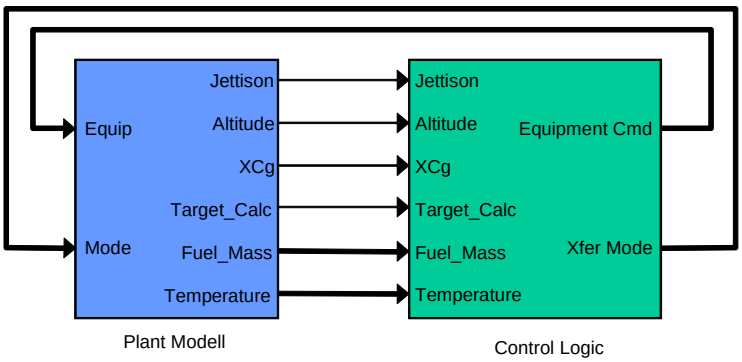


Figure 7-15: Classic Plant/Controller Feedback Model

7.2.1 Plant Model

The plant model comprises all the external factors affecting the aircraft fuel system. This includes the aircraft flight profile (speed, altitude over time), the external environment (air pressure, temperatures), as well as internal aircraft structures/systems that the Fuel System is affected by (e.g. Fuel Quantities, Densities, Temperatures) or has to control (e.g. Aircraft Centre of Gravity, Fuel System Physical Equipments).

The plant model is developed as a continuous system in Simulink. The block diagram of which is shown below:

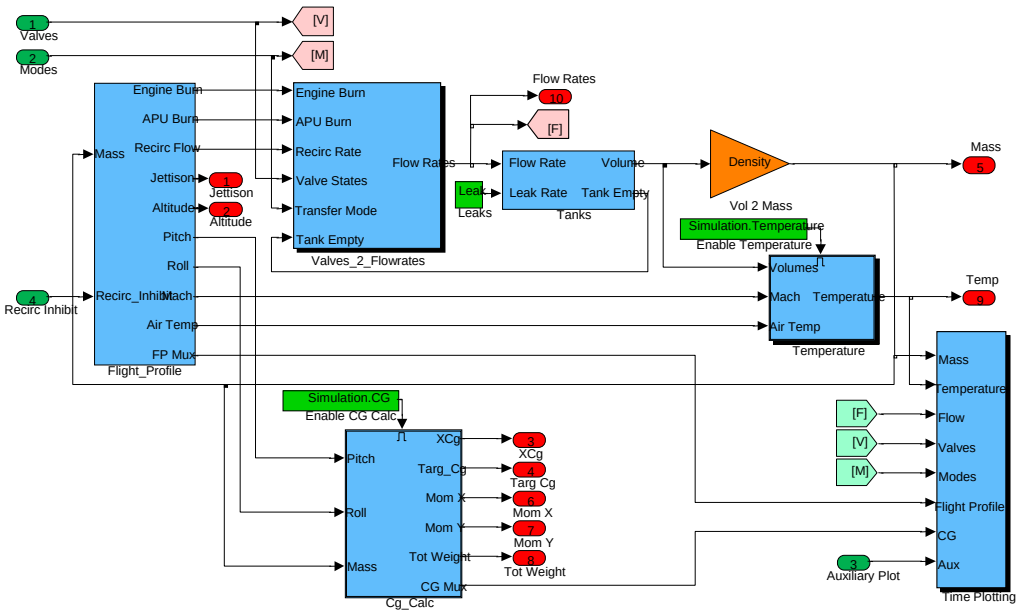


Figure 7-16: The Airbus Fuel System Modelling Environment

As can be seen, the plant model is fully partitioned and each block can be developed to any level of complexity required by the model application.

The “Valves_2_Flowrates” block will be the main repository for the physical models (see section 7.3.2 below), while the “Tanks” and “CG_Calc” blocks will be expanded by the Physical Tank Models (section 7.3.1). Other blocks, such as the Temperature block will need less detail as it is not important to the analysis of Rotor Burst affects.

The Plant Model Environment also comprises a set of GUI’s and Panels that allow the user to interact with the simulation and perform various analyses of the results. These GUIs are built upon the Matlab programming language and so can be extremely powerful.

A subset of these Panels is shown below:

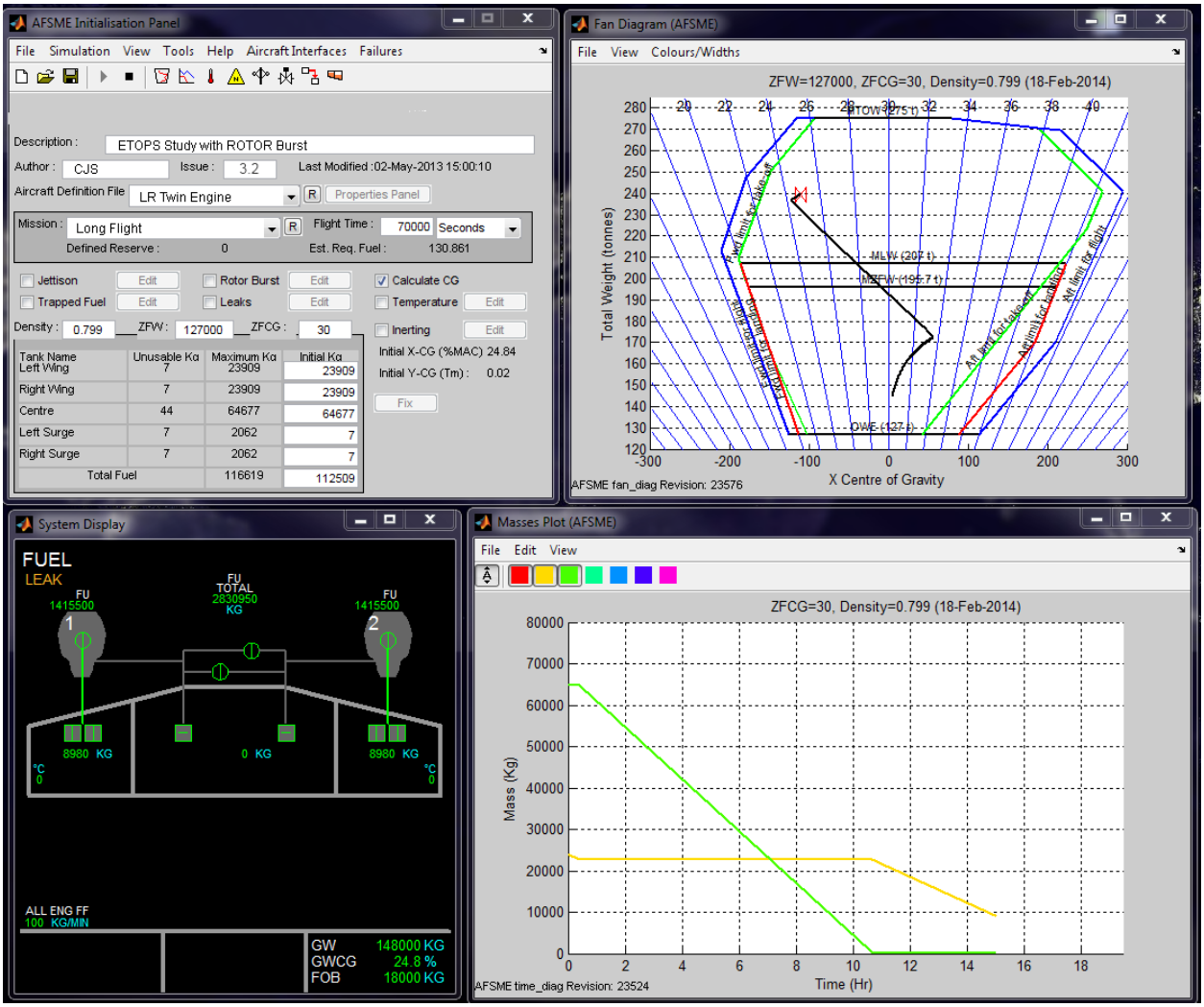


Figure 7-17: Fuel Modelling Environment Analysis Panels

7.2.2 Control Model

The Control Model represents the functional behaviour of the Fuel System. Effectively, it is the model of the functions that will eventually be implemented by software within the FQMS. The model is implemented in a hierarchy of state-charts.

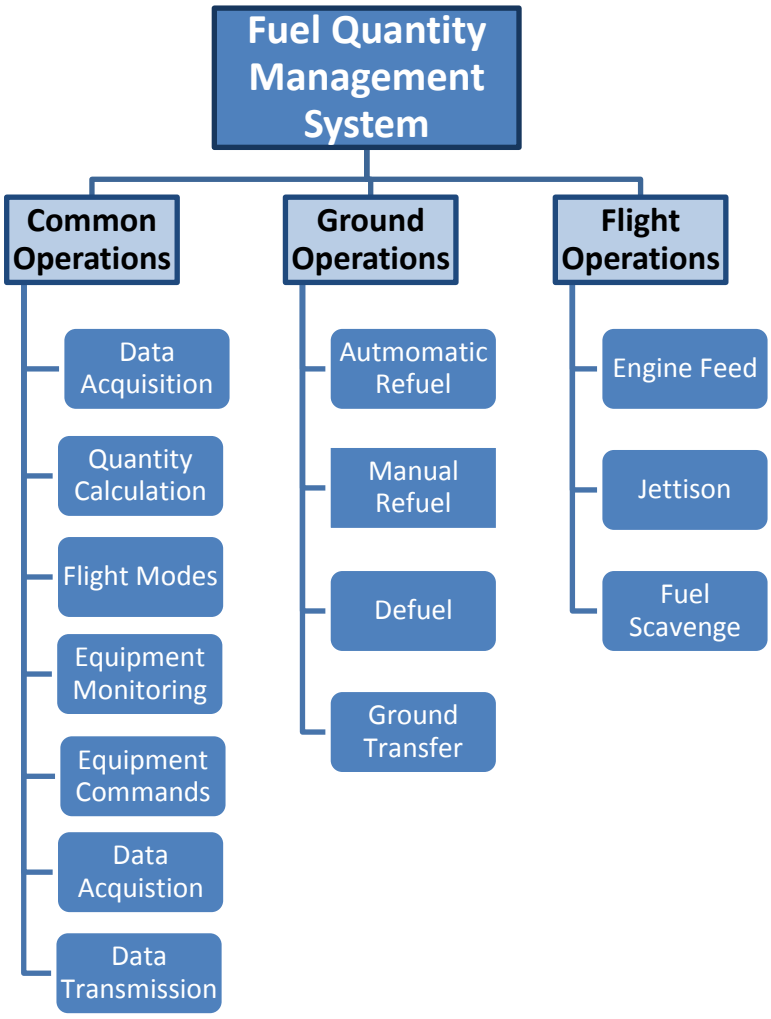


Figure 7-18: FQMS Functional Hierarchy

As the CRYSTAL use case (Uncontained Engine Rotor Failure) is effectively an in-flight safety issue, the Ground Operations parts will be removed. This will lead to simpler interfacing with the other models and avoid model "bloat" which ultimately leads to obfuscation of the model intents.

The statechart hierarchy can be seen by looking at the Jettison Section. The Jettison function is an optional addition to the aircraft whereby the pilot can select to dump fuel overboard to rapidly reduce the weight of the aircraft to below the Maximum Landing Weight (MLW). This is not a "normal" operation, but is provided for use in an emergency (such as an Uncontained Rotor Failure shortly after take-off when the aircraft is still full of fuel).

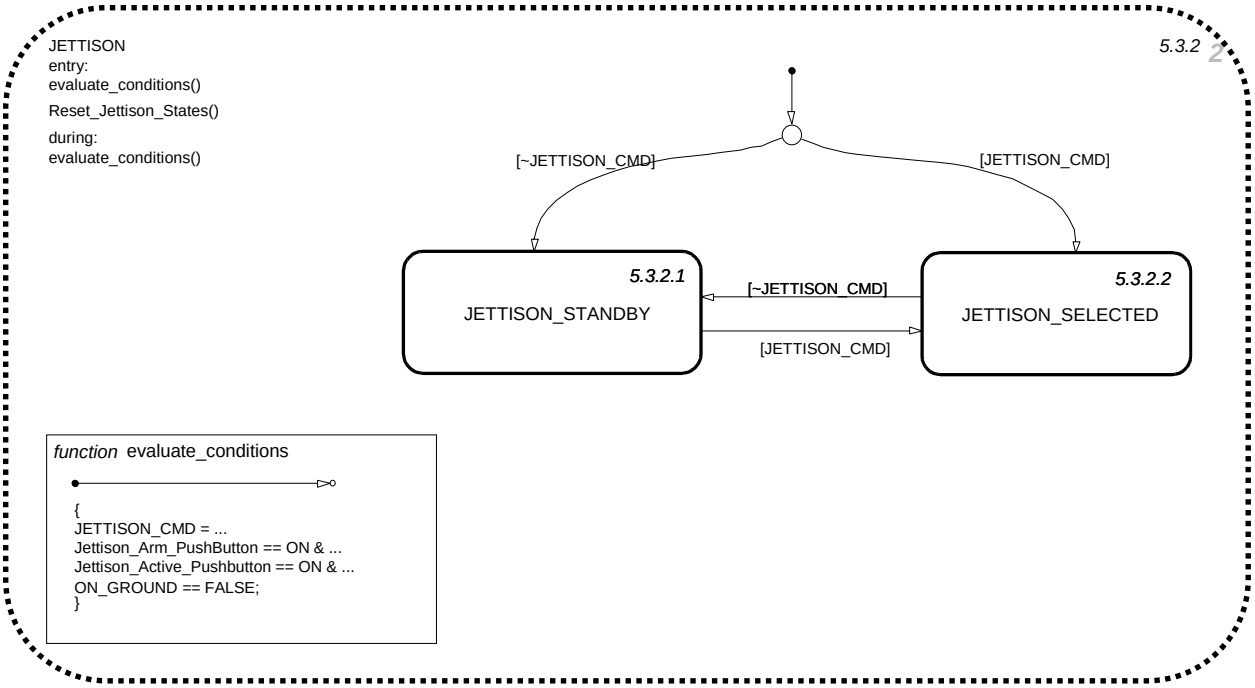


Figure 7-19: Jettison Selection Statechart

This chart plainly shows the binary nature of the jettison function. Either Jettison is selected, or it's not selected. The selection state is controlled by the "JETTISON_CMD" event which is continually calculated by "*evaluate_conditions()*" sub-function. In this case, it monitors the state of the pushbuttons on the overhead panel in the cockpit and the ground/flight status of the aircraft. These particular variables are obtained from the "Common Operations" section of the control system.

The number on the top-right corner of the statechart indicates the paragraph number of the associated Sub System Requirements Document (SSRD). This in turn relates to the requirement number as stored in the DOORS Requirements Repository.

The "JETTISON_STANDBY" state is effectively a "do nothing" state and is used to ensure that the system returns to a default state (i.e. shut all jettison valves and reset data for the crew.

The "JETTISON_SELECTED" state actively controls the jettison function. Inside this state is another statechart:

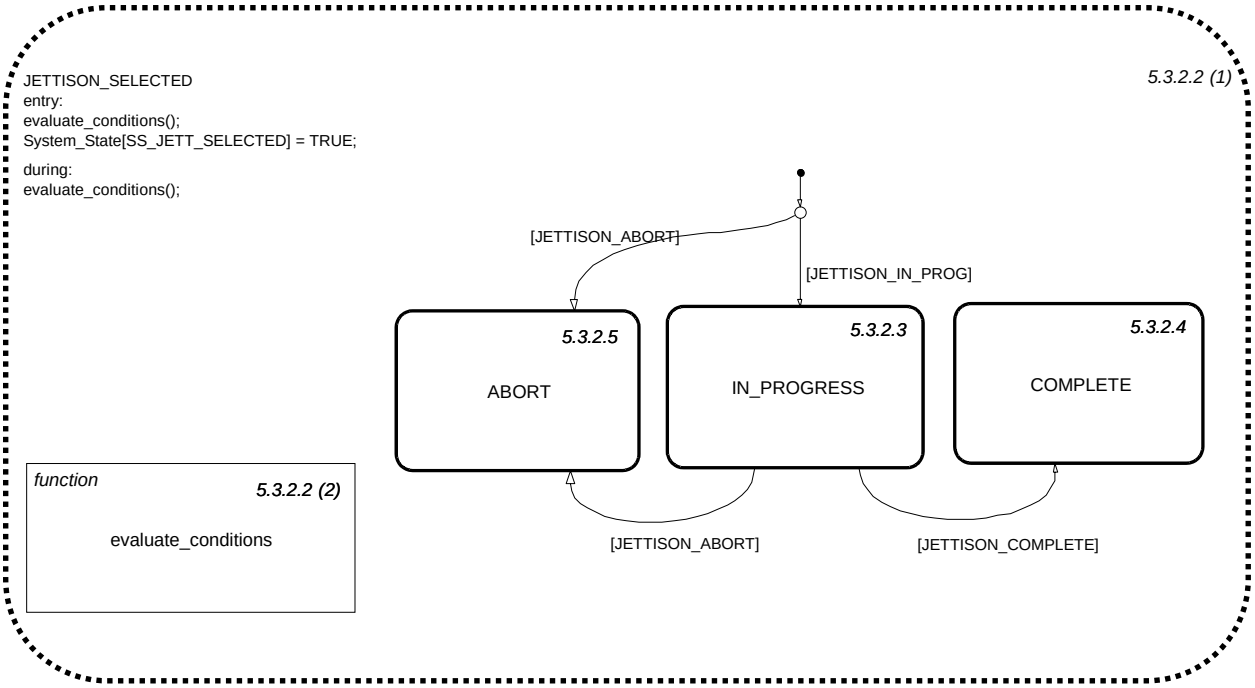


Figure 7-20: Jettison Active Statechart

Again, this chart is a simple (in this case) ternary system. The active state is controlled by any one of the "JETTISON_ABORT", "JETTISON_IN_PROG", or "JETTISON_COMPLETE" events. These events are, in turn, calculated in the "*evaluate_conditions()*" sub-function (which monitors pilot actions and other safety-related features such as tank-empty status and overall aircraft centre-of-gravity).

Each of these states also performs some actions and is further decomposed down to levels of refinement.

Using this technique, the entire system can be defined from a high level "functional view" down to equipment level control.

7.3 Physical modelling methodology

7.3.1 Fuel Tank Models

The fuel tanks are normally physically modelled in the CATIA 3d-CAD system to a high level of detail which includes the wing structure and fuel systems components (pipework etc):

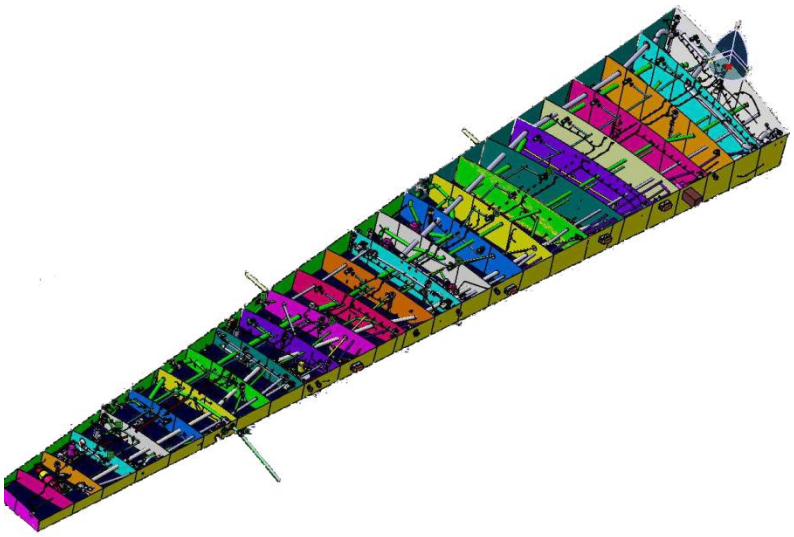


Figure 7-21: Catia model of Wing with Fuel Systems

However, this Digital Mock-Up (DMU) cannot be easily interrogated and other required operations such as wing deflections are impossible.

To overcome these shortfalls, it is intended to render the fuel tanks as a set of wire-frame models stored as a set of 3d (x, y, z) coordinates. The wing is split down into the various tank partitions (rib-bays) and simplified by removing all detail that is deemed superfluous to the safety analysis.

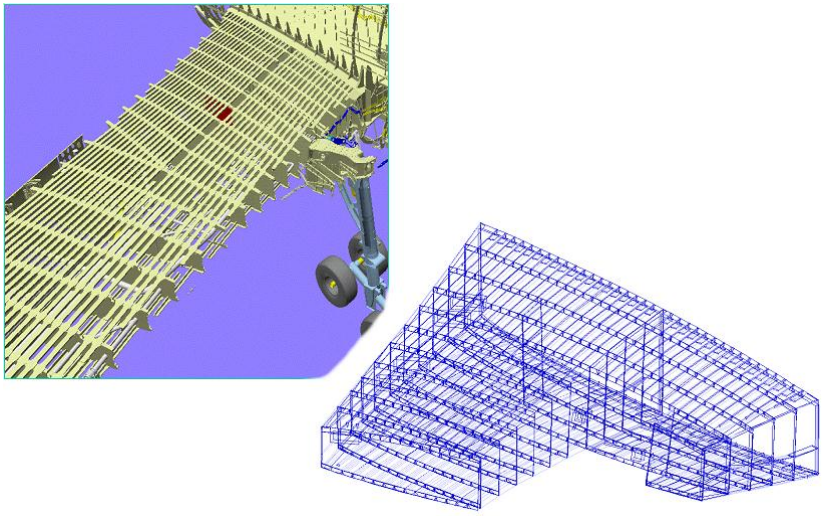


Figure 7-22: Reduction of DMU to Wireframe Fuel Tank Model

This 3d point-cloud can be deflected and manipulated based on Finite-Element-Mesh analysis data from the Loads, Aerodynamics and Aeroelastics departments. The volume of the wire-frame can be calculated using standard mathematical 3d hull techniques. This will allow for the determination of the remaining fuel in the tank after penetration from a part of the engine.

These wire-frame tank models can also include details of the pipework inside (locations, lengths and diameters). These will be used to determine which pipes may be cut by the UERF trajectories.

In case of penetration of the pipes and/or inter-tank boundaries, the fuel flow between the tanks can be modelled using standard flow equations.

7.3.2 Fuel System Component Models

A very simplified overview of the physical fuel system components is given in the figure below showing the overall layout of the valves and pumps that are controlled by the Control System described in section 7.2.2 above.

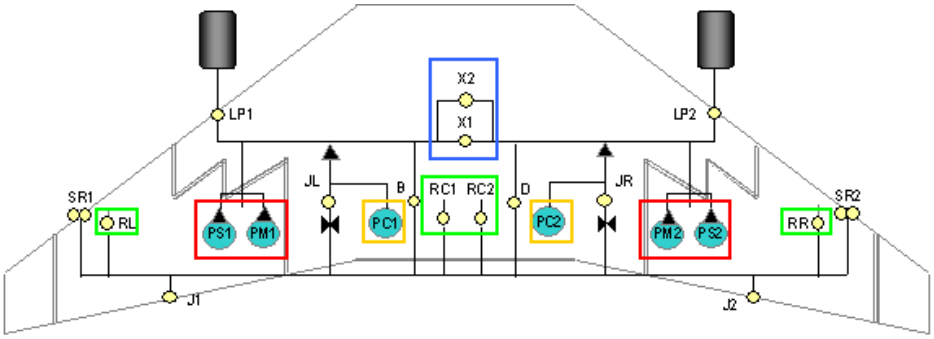


Figure 7-23: Simplified Fuel System Component Architecture

Each of these components will be modelled in the physical modelling tool; Modelica (section 6.1.8). Various details will be modelled based on the failure scenario that the safety analysis dictates. For example the Engine Feed Pumps and Crossfeed Valves, being the most critical, will be modelled to a higher level of fidelity than that of the refuel valves (which are unused during flight).

As well as the hydro mechanical equipment, the fuel tanks also contain components that are used to measure the quantity and monitor the properties of the fuel, as shown below:

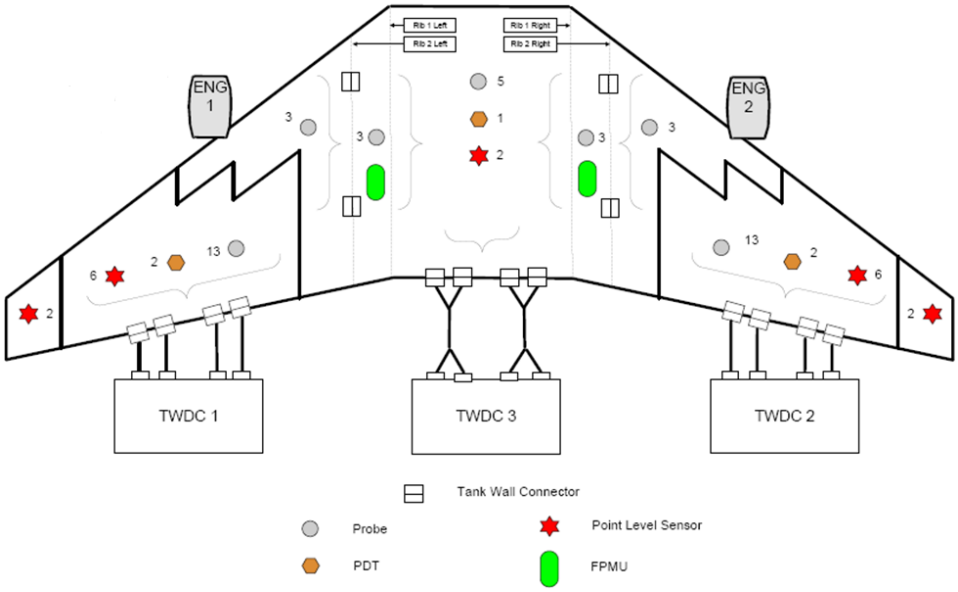


Figure 7-24: Measurement and Monitoring Components Schematic

The PDT sensors measure the density and temperature at different parts of the wing. The FPMU (Fuel Properties Measurement Unit) measured fuel permittivity. The Probes measure the physical level of the fuel inside the tanks. This is then turned into a fuel quantity and Centre of Gravity by the Fuel Quantity Management Computer using mathematical algorithms. The Point Level Sensors are secondary devices that warn the pilots when the fuel level falls below a specified amount.

The low-voltage wiring that supply each of these sensors feeds into Tank Wall Data Concentrators (TWDC) which performs signal conditioning, analogue to digital conversion and other rudimentary processing before sending the signals back to the cockpit.

The fuel control system reacts to measurements taken from these sensors and so they form an integrated part of the systems safety assessment. For example, if the sensors are damaged due to a UERF event, then a lateral imbalance due to loss of fuel in one wing could become hazardous if the measurement sensors are not able to detect the change in quantity.

The physical and electrical characteristics of each of these sensors will be modelled in Modelica.

7.3.3 Electrical Sub-System Models

All these hydro mechanical components and sensors are controlled and monitored by an electrical network. This network is routed around the aircraft to provide the necessary segregation based on the perceived hazard analyses.

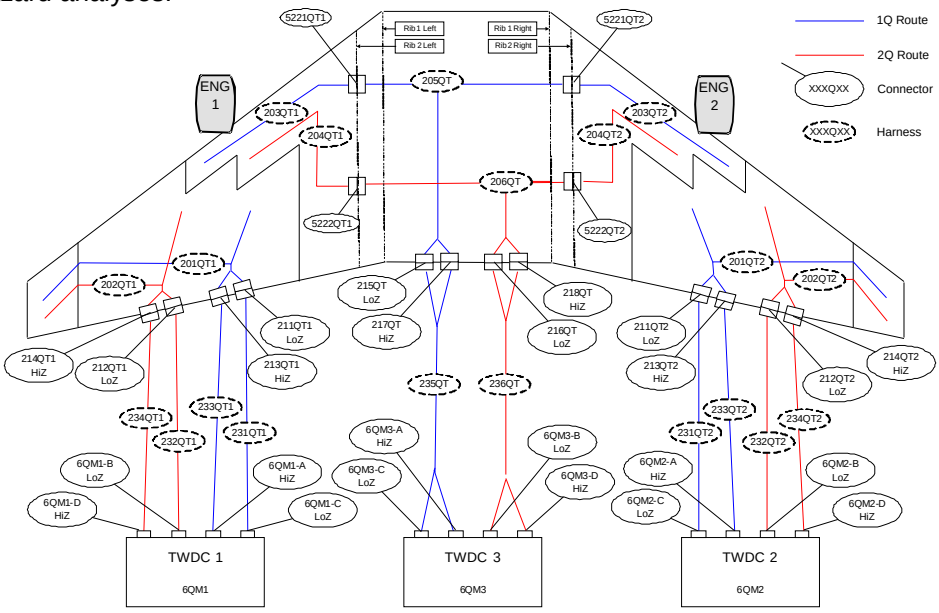


Figure 7-25: Electrical Network Schematic

It is intended to model this network in Modelica with the routing attached to the 3d-wireframe models created for the fuel tanks (section 7.3.1). This will enable the determination of which routes are severed by the UERF trajectories. Use of Modelica will allow the modelling of secondary effects such as grounding of chaffed wires and cross-connection of wires that are part of the same bundles.

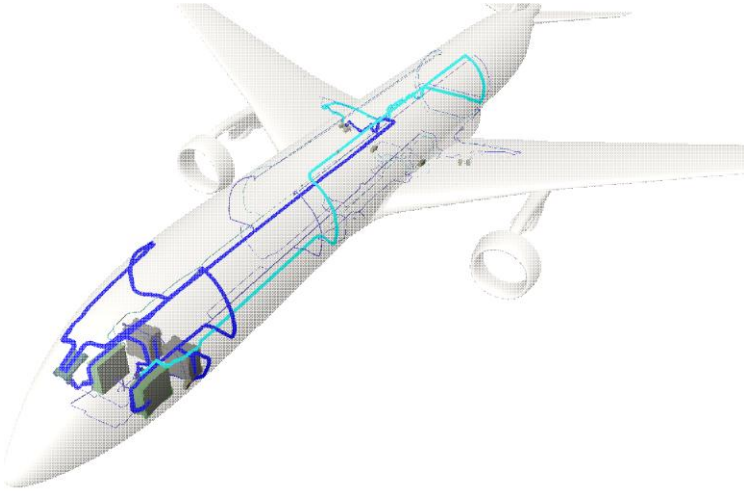


Figure 7-26: Example Electrical Control and Power Wire Routing

7.4 Dysfunction modelling methodology

This chapter describes how each of the above models can be modified to include the behaviour of the system/function/tank/component when it is affected by UERF trajectories….

… i.e. how do we model these dysfunctions:

(pictures taken from A380 Flight QF32)

7.4.1 Dysfunctional Fuel Tank:

Figure 7-27 A380 QF32 Holed Fuel Tank Leaking Fuel

7.4.2 Dysfunctional Electrical Network:

Figure 7-28 A380 QF32 Wiring Damage

Figure 7-29 A380 QF32 Damaged Signal Cabling

7.4.3 Very Dysfunctional Systems:

Figure 7-30 A380 QF32 Systems Damage

Figure 7-31 A380 QF32 Systems Disintegration

7.5 Interoperability between the Safety, Performance and Physical Models

The objective of CRYSTAL is to generate ways to perform interoperability between the different analysis tools. A number of types of operability have been considered.

1. For requirements traceability: tracing from a requirements database down to the relevant nodes that represent the embodiment of the requirement within the system models. An example of this might be to the Failure Conditions, or to aspects of an architecture that offers independence between combinations of events that lead to the loss of function.
2. Model Consistency using simulation: For helping with checking model consistency between a performance and safety model, using the safety cut-set results from the safety models analysis and the ability to simulate in both the performance modelling tools as well as the safety modelling tool to drive a simulation and directly make a comparison of the behaviour of the two models
3. Clarification of severity, e.g. within a safety model, you may have a failure event that describes e.g. a minor leakage, the idea would be to be able to link to an automated test or to a set of simulated results from the performance model to determine what a minor leakage means in absolute terms and if needed to demonstrate how this was determined by repeating the automated analysis.

7.5.1 IBM JAZZ platform – Engineering Traceability

There are different needs for the CRYSTAL Airbus UK Case study mentioned above:

- Integration of different types of data (Requirements, Design Model, Safety related data, etc.) managed by several tools to enable traceability related capabilities, such as search and query for data and data relationships, and change request impact analysis.
- Enabling co-simulation and heterogeneous simulation to improve system architecture trade-off analysis.
- Providing model management capabilities, such as configuration management and collaborative working on fine-granular levels for design, safety, and simulation models.

In addition, the CRYSTAL project aims at realizing interoperability needs by defining an open and standardized Interoperability specification, which will be based among others on the emerging OSLC standard.

Airbus Group Innovations Hamburg (EADS-IW) has implemented a first demonstrator environment based on IBM Jazz platform. First results show that key required aspects of Airbus UK Use Case and of the CRYSTAL project can be realized by IBM Jazz. For example, the current IBM Jazz platform already provides an OSLC implementation of various tools for Requirements Managements and Design Model Management, such as Doors and Rhapsody. This integration can be used to realize traceability related scenarios. Furthermore, the IBM Jazz platform provides capabilities to manage models under configuration at a fine-granular level. As such it provides a good basis for realizing Airbus UK case study needs.

Further investigations and product improvements are however required, for example in order to realize co-simulation/heterogeneous simulation. However, since IBM is a partner of the CRYSTAL project and will be

involved in the Airbus UK case study it is the aim to influence future IBM Jazz development to realize such additional relevant Airbus needs."

The IBM Software and Systems Engineering solution (SSE) is an open and extensible platform for integrating best of breed tools using OSLC. It encompasses the DOORS Requirements Management solution which Airbus uses today.

The next step is to establish a demonstrator environment based on IBM Jazz platform with Airbus Group Innovations (EADS-IW). Further information will be provided by the next version of report.

8 Detailed Description of the Use Case Process

8.1 Activities

1. Evaluation of multi-physic simulation of Fuel Management System within the Safety Analysis context.
    a. Fuel management system function simulation - Required fuel feed supply to the engines, fuel quantity measurement and fuel distribution.
    b. Build assertive models of programmatic and multi-physical components
    c. Model-base safety analysis. Applying Particular Risk Analysis with respect to Uncontained Engine Rotor Failure (UERF) associated Failure Conditions, to generate fault trees and minimum cut sets with the impacted components including systems, sub-systems and system interfaces.
    d. Control and indication interface integration in the flight deck
2. Assess technology bricks related to Fuel Management Risk Analysis use case.
    a. Produce computational components
    b. Compose candidate architectures
    c. Predict behaviour and performance of candidate solutions based on simulation and formal proof activities.
3. Express architectures as a set of interconnected and interacting components
    a. Produce IOS architecture: using IBM JAZZ platform to have the impact analysis on traceability features for the following Tool chain: DOORS, Rhapsody, Simulink, and Dymola/Open Modelica.
    b. The simulations and co-simulations will be targeted to use FMI platform.
4. Consolidate the interface and data exchange between vendor modelling tools.

8.2 Requirements Management Process

Safety databases (held in the SARAA and MV2 tool) can allocate requirements to system requirement documents (including Specifications, SRDs, SIRDs, and SWRDs). These are then copied into different folders within the DOORS.

8.3 V&V management process

Use model based system/functional approach, to analysis and simulate the failure scenario in an interactive manner to validate the requirement and verify the product supported by test evidence. The evidence can be recorded in DOORS.

8.4 Change control management process

Impact analysis including traceability can be achieved by IOS architecture (as a set of interconnected and interacting components) platform and co-simulation.

8.5 Stakeholders & Roles

Stakeholders	Role
Requirement engineer	Write the requirements
Particular Risk Analysis specialist	conduct the Particular risk analysis tasks such as Build up the appropriate models
System safety analysis specialist	In charge of system safety analysis tasks such as Build up the appropriate models
Aircraft Safety analysis specialist	In charge of multi-systems analysis tasks such as Build up the appropriate models
System modelling engineer. There are different kinds of system modelling engineers: as many as domains (thermal, functional, mechanical, …)	Build up the appropriate models and analysis.
System design engineer (or designer)	Specify the system design
System installation engineer	Specify the system installation
3D modelling engineer	Build up the 3D mock-up

9 Terms, Abbreviations and Definitions

Please add additional terms, abbreviations and definitions for your deliverable.

AMC	Acceptable Means of Compliance
CMA	Common Mode Analysis
CDD	Common Data Document
DD	Dependence Diagram
DSF	dependent System Function
ECAM	Electronic Centralized Aircraft Monitoring
CPIOM	Core Processing Input Output Module
FHA	Functional Hazard Analysis
FMEA	Failure Mode and Effects Analysis
FMES	Failure Mode and Effects Summary
FQMS	Fuel Quantity Management System
FC	Failure Condition
ICP	Integrated Control Panel
ITS	Interfaced Transition System
FPM	Failure Propagation Model
FT	Fault Tree
MA	Markov Analysis
MBSA	Model Based Safety Analysis
PRA	Particular Risk Analysis
PSSA	Preliminary System Safety Assessment
RAMS	Reliability and Maintainability Systems
SDD	System description Document
SID	System Interface Document
S/R	Safety and Reliability
SRD	System Requirements Document
SSA	System Safety Assessment
UERF	Uncontained Engine Rotor Failure
ZFW	Zero fuel Weight

Table 9-1: Terms, Abbreviations and Definitions

10 References

Please add citations in this section.

[Author, Year]	Authors; *Title*; Publication data (document reference)

11 Annex I: Detailed Descriptions of the Engineering Methods

Engineering Method: Fuel System Modelling and Simulation					
Purpose: V&V Engineer wants to run simulation of the Fuel System					
Comments: This is currently Draft and has not been validated.					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
Functional model exists in a variety of modelling applications; Modelica, Dymola, Simulink, Stateflow. These models are controlled in a versioning System, e.g. Subversion or FORGES. Some Models are managed and provided by the systems supplier. An optional model of the Cockpit Displays are provided (WAD/WAR/FWS). A pre-defined set of tests together with associated dysfunctional models.		1. In one of the modelling tools, launch service "Request list of available simulation model" 2. Request is forwarded to other tools and/or the versioning system 3. Other tools (Modelica, Simulink etc) send back list of available simulation models with validated interfaces. 4. V&V Engineers receives list of available simulation models. He selects the model/version of each of the tools, confirming the interfaces are compatible. 5. After selecting the model, launch service "Get Simulation Models" 6. Request is forwarded to all modelling tools. 7. Run combined simulation using schedular (e.g. FMI) 8. Once simulation/test verified, launch service "Send/Update Requirement" to DOORS		Tested/Verified/Validated Fuel System Requirements.	
Notes: List of tools not fixed Currently, method of replacing functional model with dysfunctional model still needs to be identified.		Notes: Look to the INSIDE R&T Project (part of SMS) for description of FMI (Functional Mockup Interface).		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
Name	Input Simulation Model	Name	FMI Model	Name	Output Simulation Model
Generic Type: (Tool or language independend type)	Simulation Models	Type:	FMI Standard Model	Generic Type: (Tool or language independend type)	Simulation Model
Required Properties: (Information required in interactions between steps)	- Simulation Model ID - Simulation Model version - Simulation Model description (e.g., "simulation of the fluid flow between tanks") - List of properties representing the inputs required by the simulation (e.g., event "launch deicing fluid")	Properties:	TBD	Provided Properties: (Information provided in interactions between steps)	- List of properties representing the results of the simulation (e.g., "Fuel Transfer Time"). - Additional list of properties defined by the FMI Standard
Description & Interoperability Additional Constraints: A "Simulation Model" Type acts as a wrapper of any kinds of simulation models handled internally by simulation tools and		Description:		Description & Interoperability Additional Constraints:	
Name	Interface			Name	Interface Coverage
Generic Type: (Tool or language independend type)	Interfacing Scope			Generic Type: (Tool or language independend type)	Exercise state of each signal in interface
Required Properties: (Information required in interactions between steps)	Interface Control Document (ICD). List of required signals (mandatory for simulation) List of optional signals (not mandatory for simulation)			Provided Properties: (Information provided in interactions between steps)	Executed/PASS/FAIL of signal and requirements Coverage metrics of signals
Name	Tests			Name	Test Complete
Generic Type: (Tool or language independend type)	Test scripts to exercise parts of model and/or to verify/validate model			Generic Type: (Tool or language independend type)	List of tests and their completion states
Required Properties: (Information required in interactions between steps)	Link to requirements in DOORS. Test version linked to model versions. Executed/PASS/FAIL Flag. Link to safety cases			Provided Properties: (Information provided in interactions between steps)	Executed/PASS/FAIL of test and requirements Coverage metrics of models

Engineering Method: Model Based Safety Analysis					
Purpose: The safety designer would like to generate fault trees corresponding to a list of failure conditions.					
Comments:					
Pre-Condition		**Engineering Activities (made of steps)**		**Post-Condition**	
The equipment safety data is provided by the system/equipment supplier. **The System's dysfunctional models are defined in the Functional Hazard Analysis process.**		**1. Create the system model to recreate:** **-System Architecture** **-System Interfaces** **2. Define applicable failure modes for equipment, to recreate dysfunctional models (failure conditions)**		**Analysis results must show the availability status of the model (system) under failure conditions.** **Relevant Safety data is presented: Fault Tree Models and Cut-Sets.**	
Notes:		Notes:		Notes:	
Artefacts Required as inputs of the Activities		**Artefacts used internally within the Activities (optional)**		**Artefacts Provided as outputs of the Activities**	
System Desing documentation and Reliability&Safety data for RAMSES model generation	**-System Design Description Document** **-System Interfaces Description Document** **-Functional Hazard Analysis** **-Failure Modes and Effects Summary**	**Model Development:** **-Gap analysis** **-Equipment and interfaces modelling** **-Failure modes and failure conditions recreation**	**Developed model including all relevant features to be considered for safety analysis.** **Recreation of failure scenarios by failure modes model combination**	**System Safety Analysis (qualitative)**	**Fault-Tree Models (with appropriate detailed descriptions) and asociated safety data**
RAMSES tool based on Alta Rica	**System Model**			Generic Type: (Tool or language independend type)	**Fault-Tree Model**
Required Properties: (Information required in interactions between steps)	TBD			Provided Properties: (Information provided in interactions between steps)	TBD
Description & Interoperability Additional Constraints:				Description & Interoperability Additional Constraints:	
Name	Safety Data (with appropriate detailed descriptions)			Name	TBD
Generic Type: (Tool or language independend type)	Safety Data			Generic Type: (Tool or language independend type)	TBD
Required Properties: (Information required in interactions between steps)	TBD			Provided Properties: (Information provided in interactions between steps)	TBD
Description & Interoperability Additional Constraints:				Description & Interoperability Additional Constraints:	

12 Annex II: Technology Base Line & Progress Beyond

This information will be collected globally, and the respective part will be inserted here. Basically it could be something like a table with a row for each engineering method and a column for the current functionality, which is the technology baseline (e.g., “data has to be transferred by hand”), and a column for the expected progress in CRYSTAL (e.g., to be implemented in CRYSYTAL / “future work”).

The exact content of this section will be defined in the next technical Board Meeting.

13 Standard for Figures and Tables

13.1 Figures

Please use the caption as shown in the example.

Figure 13-1: add title

13.2 Tables

Please use the caption as shown in the example.
