 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 Enginieering **A**cce**L**eration

Use Case Definition
Airbus-Germany
Environmental Control Systems
D201.011

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Environmental Control Systems
Deliverable No.	D201.011
Dissemination Level	CO
Nature	R
Document Version	V1.0
Date	2014-01-29
Contact	Dietmar Sander
Organization	A-G
Phone	+49 40 743 64199
E-Mail	dietmar.sander@airbus.com

AUTHORS TABLE

Name	Company	E-Mail
Dietmar Sander	Airbus Operations GmbH	dietmar.sander@airbus.com
Arne Rosenbohm	Airbus Operations GmbH	arne.rosenbohm@airbus.com
Mathias Maruhn	Airbus Operations GmbH	mathias.maruhn@airbus.com

REVIEW TABLE

Version	Date	Reviewer
0.2	24.1.2014	Andreas Mitschke – Airbus Group Innovations
0.2	24.1.2014	Thomas Kuhn – Fraunhofer IESE

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
0.1	20.9.2013	Initial Description	all
0.2	20.12.2013	Draft Version	all
1.0	29.1.2014	First Version after review	all

CONTENT

1 Introduction

1.1 Role of deliverable

This document has the following major purposes:

- Define 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 ECS Use Case Description is linked by the engineering methods with their objectives and derived requirements to the CRYSTAL Interoperability Standards and approaches towards Heterogeneous Simulation. Furthermore, the document is also linked to the Tool-Chain Demonstrator as the implementation and experimentation platform for the developed methods - D2.1.1.2.

1.3 Structure of this document

The ECS Use Case Description shows the technical context of the system to be designed by model based methods with interoperable links to model based safety analysis. For the latter purpose, various safety analysis methods are described in detail to understand the challenges for the novel approach of supported design methods by interactive safety assessment.

Requirements to perform the engineering methods will be derived for the next version of the document.

2 Use Case Overview

The Environmental Control Use Case shall support a collaborative model based approach between system design and safety domain at Airbus. The environmental control domain is responsible for a set of systems that supplies and maintains the air in the pressurized fuselage compartments of the aircraft at the correct pressure, temperature and freshness for passenger and crew comfort and equipment cooling where required Figure 1. Following system functions ensure this objective in an interacting from (closed loop) – both by the pneumatic equipment as well as by interoperating controller applications embedded on IMA-computer-network:

1. Ventilation Control,
2. Temperature Control,
3. Air Generation,
4. Pressurisation Control,
5. Supplemental Cooling.

The A-D CRYSTAL use case targets the Ventilation Control System (VCS) which supplies the AC with constant flow (mostly steering mode) – whereas close control loop are integrated via the Temperature Control and Cabin Pressure Control System.

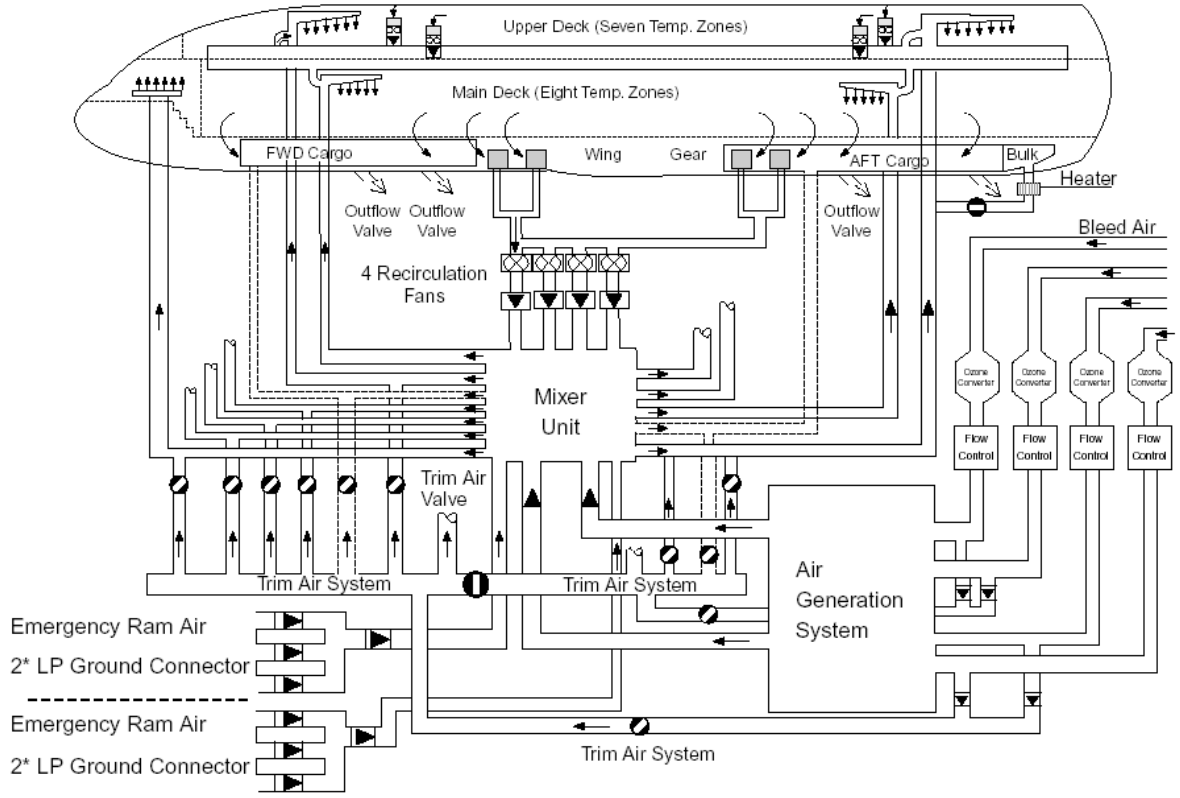

Figure 1 Air supply architecture of the A380

The system design in the environmental control domain is supported by various model based methods using e.g. CFD, Flowmaster or also SCADE of equipment and controller specifications. Functional model based design within ventilation control department is performed currently on executable models in Matlab / Simulink that are used to validate requirements and verify controller

applications for nominal and degraded system and equipment behaviour. Safety aspects are studied in the role of design observers in associated domains and mostly on higher level: In the current process, safety models are manually derived from Matlab / Simulink models to create relevant design entities in e.g. Fault Tree models within the specific context. Additionally, Reliability Block Diagrams (RBD) are created in Airbus Proprietary Safety Database SARAA to model and compute qualitatively and quantitatively Probabilities of Functional Hazards. Although cross-domain iterations ensure currently consistent design, CRYSTAL targets to cross the parallel engineering into an integrative approach.
The model based approach in the ECS use case shall bridge the design and safety activities in an integrative and data interoperable manner supported by e.g. multi-view point modelling with preferentially associated analysis methods. Figure 2 shows the different model based approaches in system design and safety domain, which depicts various model types of different model representations. Multiple design aspects like non-functional and functional design properties shall be integrated in one cross-domain model which combines also nominal as well as erroneous behaviour of the controller application as well as equipment functions. The non-functional aspects formalize performance and safety properties for at least two different ‘descriptive‘ viewpoints in which the design shall be assessed with respect to dynamic and/or state-dependent executable models. An integrated framework shall be developed for defining overall design and assessment entities based onto an appropriate data model that also supports transformation and interoperability with standard analysis tools. The environmental control use case performs concurrent modelling and analysis for detailed design on system and equipment level that improve early maturity and reduce design costs.

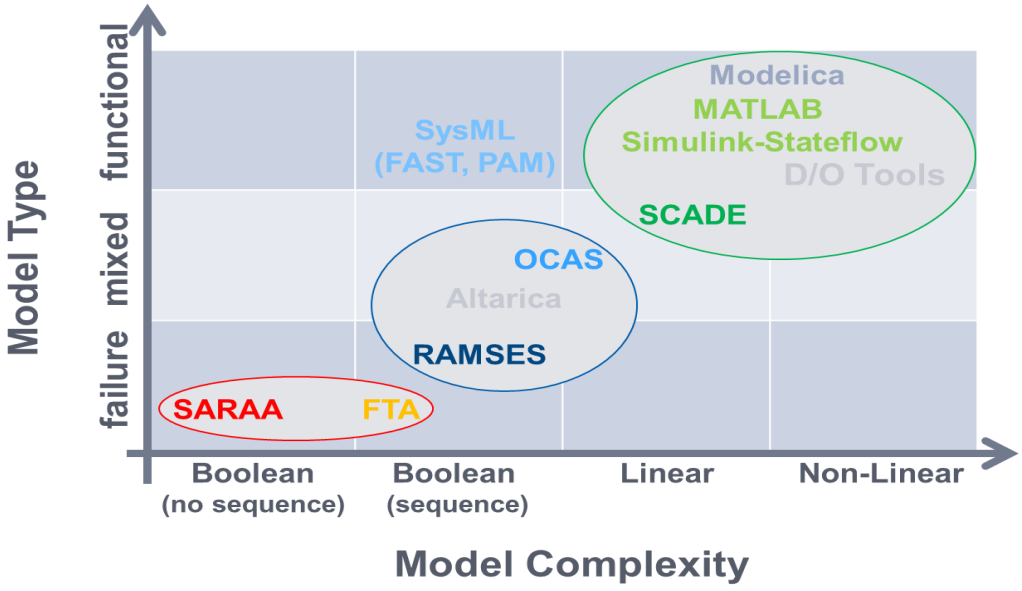

Figure 2 M&T landscape for system design

For the major safety analysis, the Airbus safety departments receive the following input from the design office, respectively system supplier:

1. Functional block diagram to define the interrelations between system function and sub-functions. The input is used to generate the FHA.
2. SRD/PTS to define the architecture and equipment list of the technical system. The input is used to generate the PSSA.
3. SIRD and IHA to determine the aircraft installation location of the system equipment. The input is used for the safety classification of (sub-)functions.

There is a close interrelation between design office and safety domain to achieve the SSA. E.g.: If the installation position of a malfunctioning equipment introduces a risk, such as an overheat in the aircraft fuel vapour zone, it maybe decided to move the equipment to a different location.
The Safety domain of Airbus is responsible for the following major safety analysis documents, generated during aircraft development and used for aircraft type certification:

Functional Hazard Analysis (FHA)
The FHA contains a functional breakdown of the technical system under development and identifies the interfaces to other a/c systems, such as the electric power supply, the hydraulic system, the Bleed Air System or the Integrated Modular Avionics (IMA). The functional breakdown is extended with malfunctions derived from a trivalent mental failure model by each safety engineer. Each functional failure is assessed according to its effect on the aircraft and humans to determine the safety classification (CATASTROPHIC, HAZARDOUS, MAJOR, MINOR) for each sub-function. The safety classification is a strong, if not the strongest driver for the system architecture and design quality. The FHA is completed during the aircraft concept phase (Maturity Gate 4) and early in the development phase of the aircraft.

Preliminary Safety Assessment (PSSA)
The PSSA is an extension to the FHA and contains additionally the mapping of functions to a list of equipment and a/c installations[1]. The safety classification of each sub-function determines the Design Assurance Level (DAL-A, -B, -C, -D) for the equipment. The PSSA will contain the first qualitative Fault Trees and is completed during the a/c definition phase (Maturity Gate 7), Figure 3.

System Safety Analysis (SSA)
The SSA is an extension of the PSSA and contains the completed equipment list, along with the Failure Mode & Effect Summary (FMES) derived from supplier input Failure Mode & Effect Analysis (FMEA) data. Furthermore the SSA contains the quantitative fault trees, derived by Fault Tree Analysis (FTA) or analysis of Dependency Block Diagrams (DD) to demonstrate compliance to the requirements for the occurrence probability for failures of system functions with a safety classification higher than MINOR. This document is used for aircraft type certification.

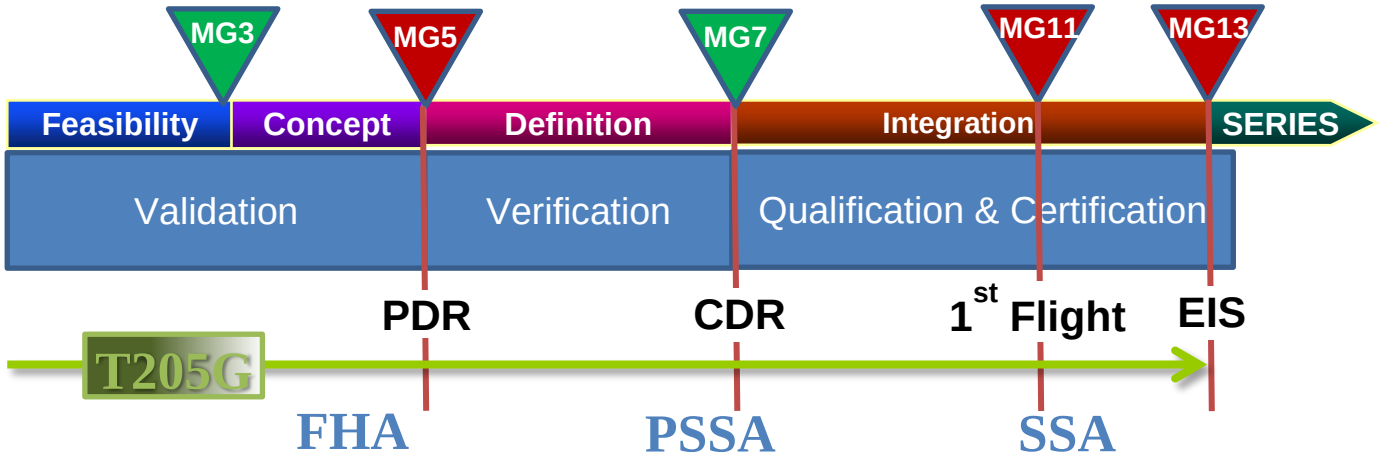

Figure 3 A/C-Development and Safety Analysis

[1] The location of the installed equipment in the aircraft may require further safety analysis, which belongs to the Zonal Safety Analysis (ZSA).

T205G
It is the collection of all safety requirements concerning one ATA (sub-)Chapter and is used for requirement validation and -verification during a/c development. In order to complete the a/c development phase the T205G is used for equipment qualification.

Tool Support
The main tool in use by the Airbus safety domain to support the system safety process and its methods is SARAA (FHA, PSSA, SSA). Additionally, some tools are used to support the interchange with supplier safety data derived from the following safety analysis:

1. Fault Tree Analysis (FTA): FTPlus or Relax
2. FMEA/FMES: Microsoft Excel, FTPlus, Relax, IQ-FMEA

CRYSTAL Objectives
Key objectives for the use case are CRYSTAL methods and bricks for seamless data interoperability and multi-viewpoints systems engineering. Generic design entities are defined in a sub-structured environment as the instantiation of a domain Reference Technology Platform (CRYSTAL RTP). Standard tools like Matlab / Simulink and e.g. safety analysis tools are connected via the RTP ensuring data exchange by IOS. By exercising RTP – system domain models are transferred and integrated in safety models in order to consider inadvertent failures and emergent functional behaviour. In particular methods to determine qualitatively and quantitatively the impacts of multiple-failure occurrences that are subject to the ECS use case, as well as failure propagation and fault tree analysis including timed and state-dependent failure analysis.

Following high-level activities will be performed within the ECS use case in CRYSTAL:

1. exercise re-use and seamless data link of functional models in different domains
2. modelling of design entities of different domains in an integrated (intersecting) model
3. modelling of functional requirements for design validation
4. execute functional models for controller application as well as the environment for validation and verification purposes
5. integrate safety attributes into functional models to enable seamless safety analysis from functional model
    - include safety attributes (non-functional statistic properties) attached to physical components
    - elaborate on functional safety properties e.g. Inadvertent failures like delay / loss of signal
    - Erroneous properties like commission / omission failures, deviated transfer functions
6. elaborate emergent behaviour on
    - combine erroneous functions for analysis of double / triple failures
    - elaborate daisy chain of multiple functions
    - run random sequencing of functions and dysfunctions
    - elaborate method for decomposition and detailing erroneous functions
    - validate methods on failure propagation
7. derive and harmonize requirements from the use case for CRYSTAL

3 Detailed Description of the Use Case

Design office tools use functional models of linear or nonlinear complexity as shown in Figure 2. Due to the close interrelating work between design office and the Safety Domain, the same system simulation tool shall be used for functional and malfunctioning behavior. The benefits of this approach are obvious:

1. **Reduction of workload:** One model can be used as the basis of the other
2. **Approval for Certification:** Reduced discussion/review time for acceptance of the extended design office model compared to a new model for simulation of malfunction behavior.
3. **Costs of database management:** One tool will use one database for libraries and one version management (traceability).
4. **Extended use of the new simulation model:** Aircraft development (validation, verification, qualification, certification) and aircraft in-service (Maintenance and Major Mods).

Therefore CRYSTAL uses typical design office tools to define and simulate failure behavior.

3.1 Model-based System Engineering (MBSE)

Generally speaking a formal description of systems is used within Airbus design departments to simulate its nominal behavior. The Finite State-Machine formalism combined with the mathematical description of dynamic system behavior is most common to simulate a system within Airbus.

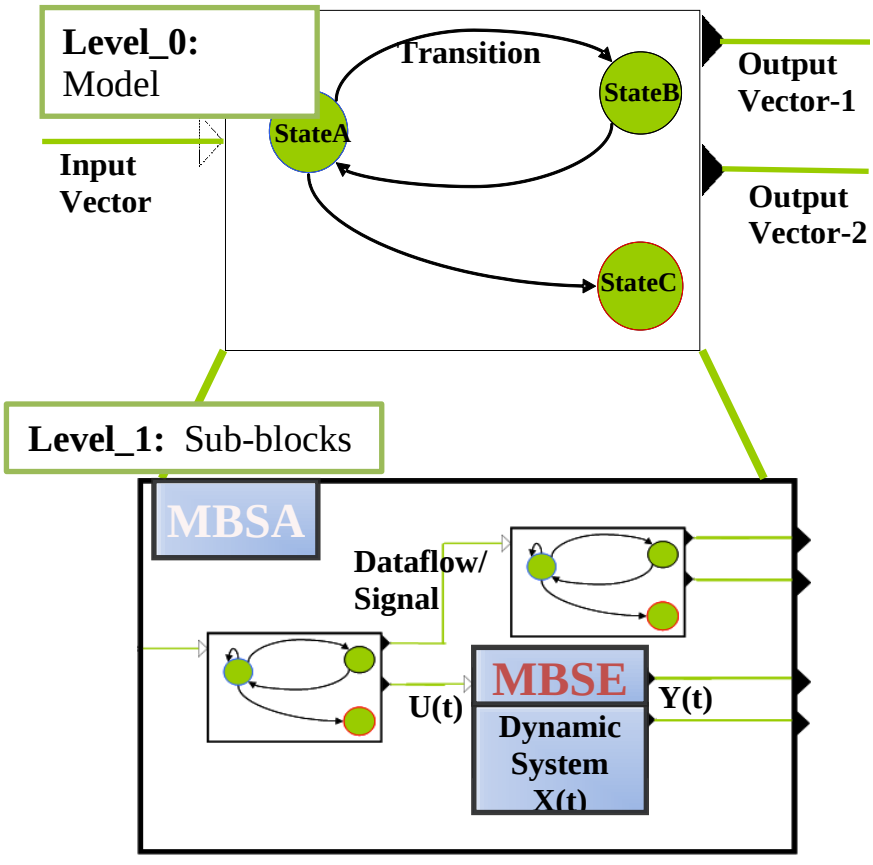

Figure 4 Hierarchy of System Models

The tools, MATLAB/SIMULINK -including Stateflow- and SCADE define the nominal behavior for simulation and deal with complexity through the introduction of model hierarchy. Model hierarchy or the consists-of relation between several Simulink blocks, enables the user to group more than one Simulink block within one subsystem, generally several blocks corresponding to one function or one piece of equipment (controller, sensor, actuator, etc.). MATLAB/SIMULINK and SCADE use models for functional simulation and are therefore referred to as models for Model-based Systems Engineering (MBSE). The functional simulation reflects the nominal behavior of a technical system, or in Airbus wording: Parts of an ATA-(Sub-)Chapter. The nominal behavior of the model corresponds to the specification of the technical system. The nominal behavior, includes degraded or failure behavior in terms of specified 'robustness' of systems (e.g. for designed coverage of single failures).

To name a few MBSE in use by Airbus design offices the following incomplete list is named hereafter:

1. **Hamburg**
   ATA 21, Air-conditioning Systems, uses a variety of Simulink models throughout its ATA Subchapters. These are the Ventilation Control System (VCS), Supplemental Cooling System (SCS) and Cabin Pressure Control System (CPCS).
2. **Bremen**
   ATA 27, Secondary Flight Controls, administer a Simulink library.
3. **Toulouse**
   ATA 27, Primary Flight Controls, administer a SCADE library.

The following sections detail the activities to generate the MBSA using the MBSE as the basis.

3.2 MBSE applied on VCS Use Case

The ECS Use Case shall support a collaborative model based approach between system design and safety domain. An integrative model approach represents design aspects from different domains shall enable multi-view point modelling with associated analysis. Both functional and non-functional design properties are modelled, combined with nominal as well as erroneous behaviour of the controller application as well as equipment functions. The non-functional aspects formalizes performance and safety property for least two different ‘descriptive‘ viewpoints in which the design shall be assessed with respect to dynamic and/or state-dependent executable models. An integrated framework shall be developed for defining overall design and assessment entities based onto an appropriate data model that also supports transformation and interoperability with standard analysis tools. The environmental control use case supports concurrent modelling and analysis for detailed design on system and equipment level that improve early maturity and reduce design costs.
Another model-based approach was for example the model based functional specification in Rhapsody: Here, the functional part of a system specification was modelled parallel to the conventional document-based system development process. The model was created with the simulation tool Rhapsody from IBM. In Rhapsody, System Modeling Language (SysML) is used. The aim of the pilot project was to demonstrate that the demands made by the ECS are satisfied by SysML, and on the other hand, that the collected knowledge during the modeling process helps to improve the PTS.

3.2.1 System Design Model

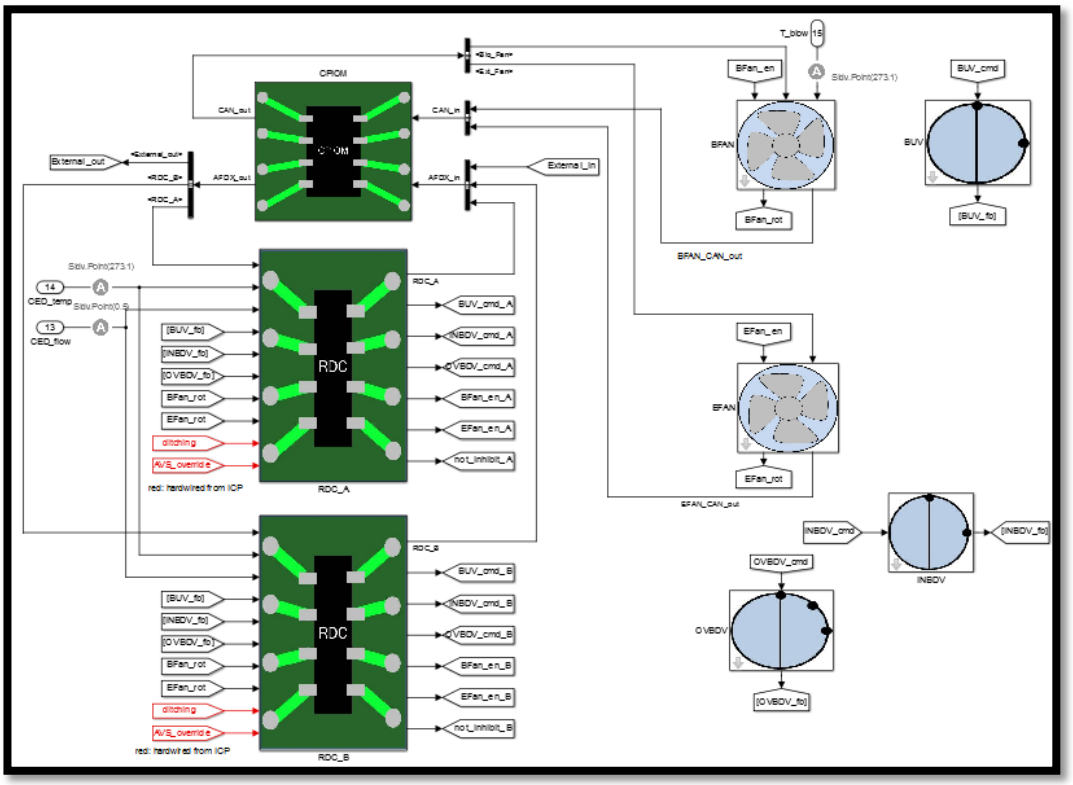

Figure 5 Avionics Ventilation Model

In **Error! Reference source not found.**, a screenshot of the specification model, designed in Simulink, is shown. There is the possibility to simulate the model by manipulating the input and failure signals. Furthermore, it is possible to check the model with the design verifier. The design verifier is explained in chapter 3.3.6.

The VCS Systemmodel includes some kind of equipment, interfaces, connections and wiring. On the left side of Figure 5 the two RDCs and the CPIOM are shown. In these modules, the logic behaviour of the system is implemented. Input and output signals and status signals of the equipments are processed and the system components are driven.

The system components can be divided into three types. The first type are the fans. This includes the Avionics Extraction Fan and the Blowing Fan. Furthermore there are two kinds of Valves. The Inboard Valve and Backup Valve are kind of two-point valves. These valves can only be fully closed (FC) or fully open (FO). In addition to that kind of valves there is the Overboard Valve. The OBV can be FC, partially open (PO) or FO.

The interconnection of the equipment, the RDCs and the CPIOM is done via single signal connections and bus bars.

The model can be used to simulate the specific conduct of the Avionics Ventilation and check the accuracy. The environment model is important to check the correct function of the system. For that, the system components of the environment model are needed to give back feedback of the occupied state. This feedback signals are needed for the system to respond to errors.

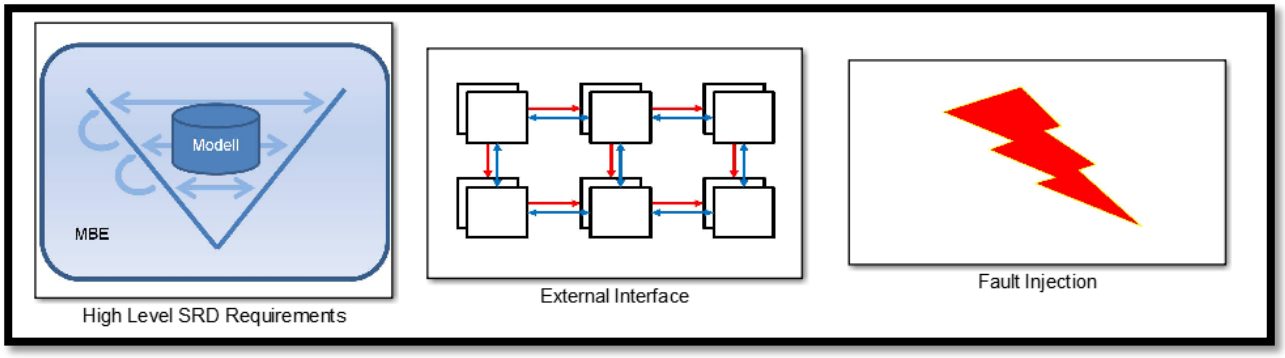

Figure 6 Model Components

On the right bottom of Figure 6 subsystem for a failure injection is pictured. With the failure injection it is possible, to stimulate the model and simulate failures of the fans and valves and check the reaction of the system. It is possible to simulate one or more components faulty and check the expected behaviour.

In the next subsystem named External Interface, it is possible to manipulate the internal and external signals needed for the system. For example there are signals coming from the Cabin Pressure Control System (CPCS) or from Landing Gear System (LGS). These signals will be evaluated by the system and the system reacts accordingly. On the right side of **Error! Reference source not found.** all interface signals are shown. These signals are the input signals for the model. The data types are boolean, int and double as well. The other signals on the left side of Figure 7 are bus signals to simplify the signal handover inside the model.

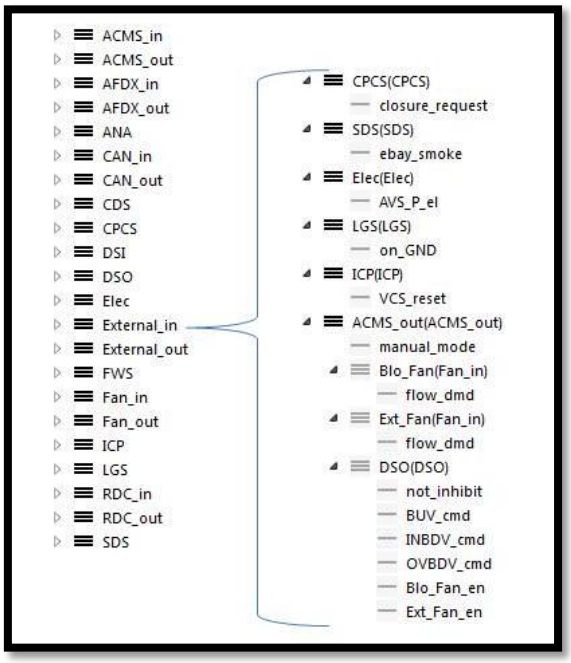

Figure 7 Interface Signals

The next subsystem, named High Level SRD Requirements shown in Figure 6, is needed to include top-level requirements into the model and check the correct behaviour of the system. This is done automatically by the Design Verifier included in MATLAB/Simulink which is described later.

The simulation parameters are set to:

- Start time: 0.0
- Stop time: inf
- Type: fixed-step
- Solver: discrete (no continuous states)
- Fixed-sted size: 1.0

3.2.2 Abstraction Level of the Model

The ECS use case model has a high level of detail for the controller architecture. Nevertheless, the whole model has a high abstraction level as it is shown in Figure 8.

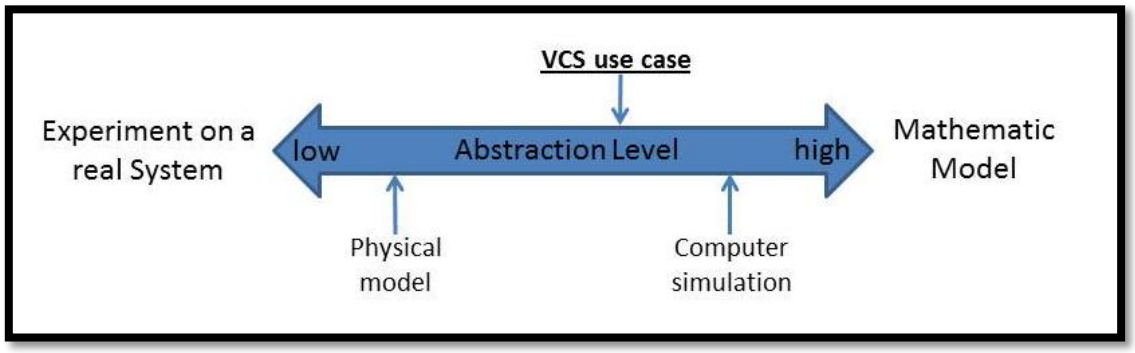

Figure 8 Abstraction Level

The reason is that the environment model is only designed to react to the signals of the controller and give back feedback to the controller. Therefore, the environment model consisting of the fans and valves is not as detailed as the real equipment and the controller architecture. This is the reason why the model has a relative high abstraction level. A simple two-point valve is designed in **Error! Reference source not found.**9 below.

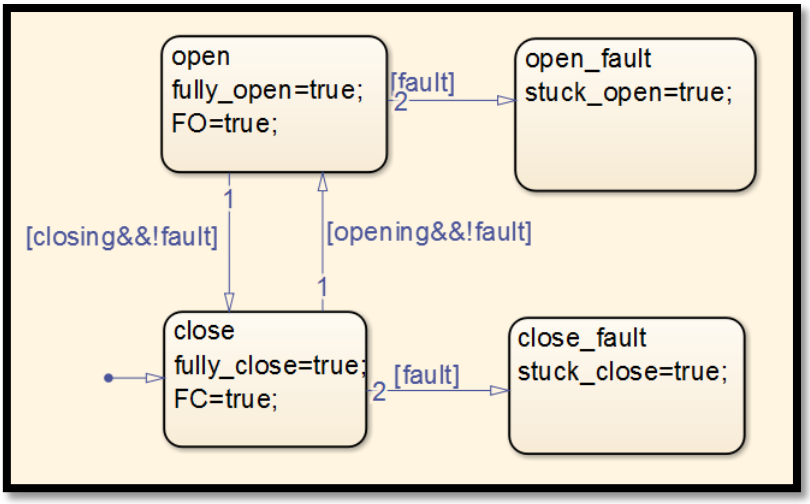

Figure 9 Two-point Valve

The behaviour of the equipment is modelled by Stateflow. There are only three input signals and four output signals. In a real system there are much more in- and output signals of a valve or fan. On the other hand, the functions, modes, warnings and so on are much more detailed than the equipment is. For the creation of the controller, different functions of MATLAB, Simulink and Stateflow are used. Some dedicated drop-down menus are shown in Figure 10: On the left side of **Error! Reference source not found.** 10, MATLAB code is used to calculate temperatures to check if cooling is sufficient. On the right side, a truth table are used to set modes.

```
function check_cooling=f(CED_temp,CED_flow,P_el)
%ARINC600 blowing air mass flow reccomendation (in kg/S/kW):
if CED_temp<30
    k=0.0377;
elseif CED_temp>40
    k=0.0611;
else
    k=0.00234*CED_temp+0.0377;
end
%determine recommended blowing air mass flow:
ARINC_flow=P_el*k;

%compare to value measured by cooling effect detector (monitoring device):
if CED_flow<ARINC_flow;
    check_cooling=true;
else
    check_cooling=false;
end
```

	Description	Condition	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
1	on ground indication	External_in.LGS.on_GND==1	T	F	F	F	T	T	-	-	-	-
2	smoke indication	External_in.SDS.ebay_smoke==1	-	T	F	T	F	T	-	-	-	-
3	skin valve closure request	External_in.CPCS.closure_request==1	-	F	-	T	-	-	F	T	-	-
4	manual control of equipment requested	External_in.ACMS_out.manual_mode==1	F	-	-	-	T	-	-	-	-	-
5	inboard valve stuck close	INBDV_stat==2	F	F	F	F	F	F	T	T	F	T
6	overboard valve stuck close	OVBDV_stat==3	F	F	F	F	F	F	F	F	T	T

Action Table

#	Description	Action
1	normal ops on gnd or smoke detected in flight OVBDV_FO, INBDV_FC, BUV_FC	central_control_mode=1;
2	normal ops in flight OVBDV_FC, INBDV_FO, BUV_FC	central_control_mode=2;
3	smoke detected in flight and closure request from CPCS OVBDV_FO, INBDV_FC, BUV_FC	central_control_mode=3;
4	manual mode equipment is commanded by Central Maintenance System	central_control_mode=4;
5	this logical combination is assumed to never occour	central_control_mode=0;

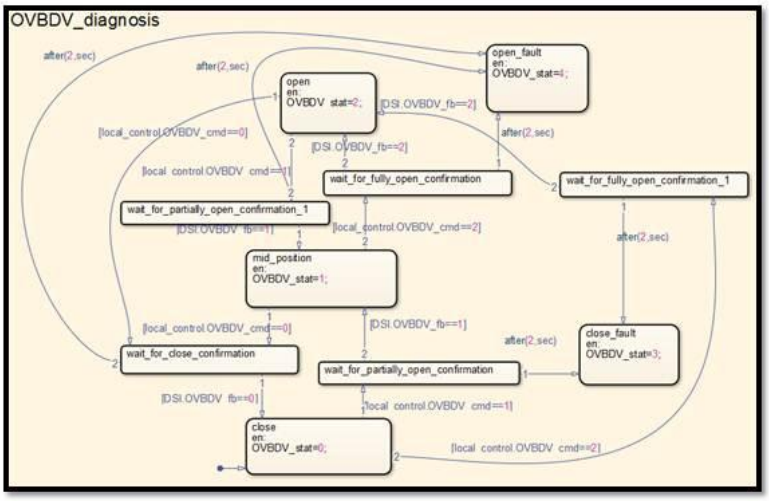

Figure 10 Tools used for controller modelling

3.3 Functional Failure Analysis

Within the Airbus safety domain the safety engineer uses a mental and trivalent dysfunctional model. In order to perform a Functional Hazard Assessment (FHA), the safety engineer assesses the effects of functional failures using the functional breakdown of the technical system and a Functional Block Diagram as input.
The Functional Failure Analysis (FFA) can be used as the basis for Safety & Reliability Assessments. We can distinguish two basic kinds of failure assessments in Figure 11 and in Figure 12:

A) Failure Mode Effect Analysis (FMEA)

FMEA assesses single failure modes in a forward propagation and evaluates the severity of effects.

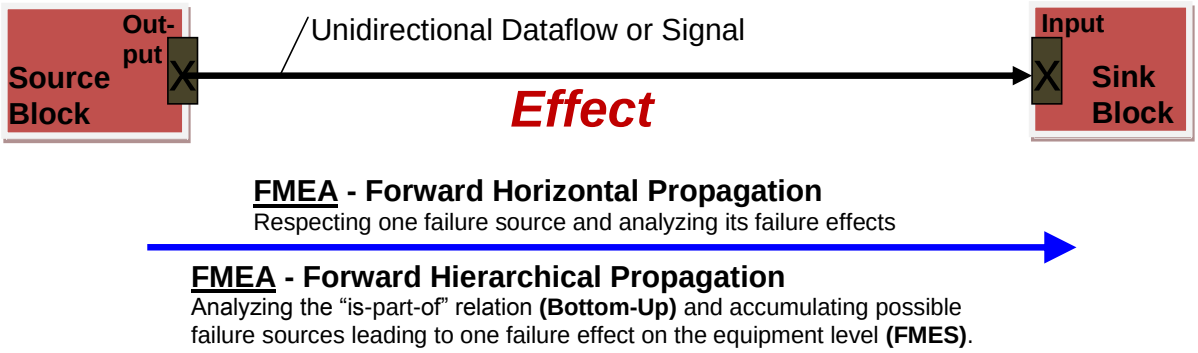

Figure 11 FMEA and Functional Block Diagram

B) Fault Tree Analysis (FTA)
FTA assess an undesired Top Event (Failure Condition, FC) in a backward propagation by determination of all causes - all possible minimized failure mode combinations leading to the Top Event (FC).

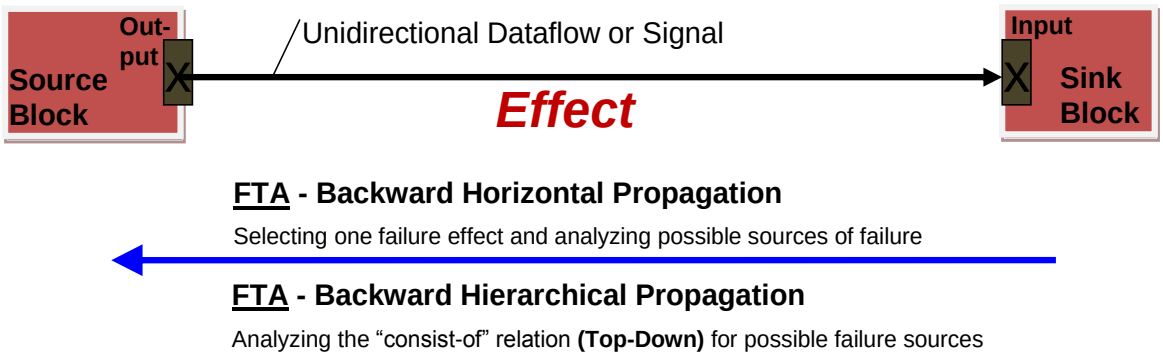

Figure 12 FTA and Functional Block Diagram

3.3.1 FFA - General Use Case

Action 1: Define the Functional Breakdown
Defining the main system function and breaking it down to adequate sub-functions, e.g. an arbitrary control application consists of:
System internal:

- Sensing equipment (sensors) and Control CPU (hard-&software)
- Controller equipment
- Actuation (motors, valves, pumps, etc.)
- Communication interlinks – CAN and discrete signals

System Interface:

- Power supply to drive the actuation (electric, hydraulic, etc.)
- Communication interlink - IMA and AFDX, System Controller and CAN

Action 2: Extend the functional breakdown with a Failure Model
Use the functional failure model and its malfunctions on each identified function, sub-function and functional unit.

Action 3: Exclude Malfunctions and justify

Exclude non-applicable malfunctions and note the rational (Requirement Validation).

Action 4: Perform the forward failure assessment (FMEA)
Assess the effects of malfunctions and evaluate its severity. FHA Maturity: Classification of Failure Conditions (FC).

Action 5: Map functions to equipment and perform backward failure assessment (FTA)
In the course of development, it becomes clear of how the required system function will be implemented by equipment. PSSA Maturity: Perform a qualitative FTA by respecting all failure combinations that lead to the undesired effect, the FC on aircraft level.

Action 6: Justified Failure Mode selection and summary (FMES)
List the identified failure modes (short-circuit, fail-close/-open, stuck-close/-open…), which lead to one identified equipment malfunction, the so-called Equipment Failure Condition (EFC) within the FMES form sheet.
Exclude non-applicable failure modes, note the rational and summarize in accordance to the Equipment Failure Conditions (EFC). FMES: Accumulate the failure rates of equipment failure modes in respect to one EFC, e.g.: “Overheat, Leakage, Rotor Burst, …”

Action 7: Database of failure rates
Interlink failure modes with equipment failure rates utilizing FMEA/FMES form sheets and use as input to the FTA and classified FC on aircraft level (SSA Maturity).
Revaluate Action 5 if safety or reliability requirements are not met. Change system architecture, equipment reliability or monitor strategy.

3.3.2 Failure Model for MBSA

CRYSTAL will use MATLAB/SIMULINK models as input to extend the MBSE with failure behavior derived from the Functional Failure Analysis. The document refers to these extended MBSE as models for Model-based Safety Analysis (MBSA).

Generally the following failure modes of a dataflow or signal can be distinguished in Figure 13:

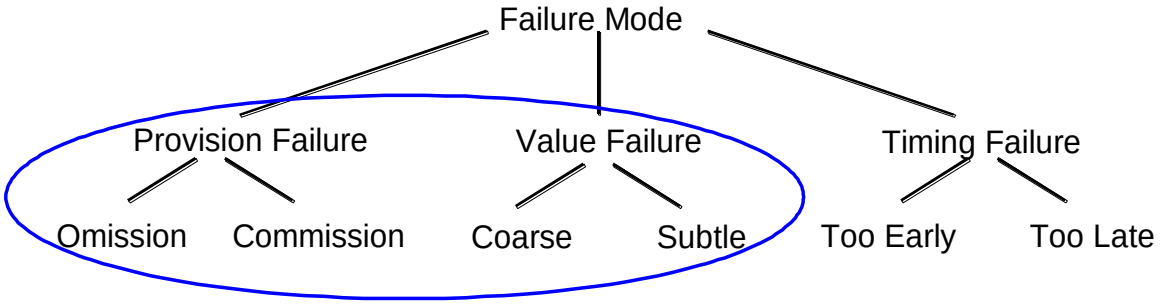

Figure 13 Failure Model for Signals

The failure class Provision Failure respects a faulty data supply. “A computing system can not fulfill the requested demand.”

Omission: Absence of input for the receiver.

Commission: Not expected input for the receiver.
The failure class **Value Failure** respects a wrong value regarding expected data.

Coarse Value Failure: Detectable wrong input for the receiver.

Subtle Value Failure: Undetectable wrong input for the receiver.
The failure class **Timing Failure** respects the timely failure behaviour of dataflows.

Too Early: Early input for the receiver.

Too Late: Late Input for the receiver.

Cyclic working computing systems, such as any control application, do not need to regard Timing Failures[2], instead *Too Late* can be mapped to *Omission* and *Too Early* to *Commission*.
The abstract dataflow or signal failure model does not make any statements about possible causes of the failure, such as component loss, design- or human error. The lack can be resolved, if the source of the dataflow, the functional block, defines possible cause of failures (the component, the source of the dataflow fails, because ...).The following introduces the trivalent failure classes that are exemplarily applied to a braking- and steering system, considering only single output functions for descriptive reasons.

The dysfunction **Loss of Function** of a source component for a dataflow/signal, results in an output deviation of ***Omission failure*** class, and is defined as:

„No function on demand“

Applied to the function of an embedded system, e.g. Braking or Steering, the dysfunction can be named as

„No breaking effect on demand“ or „No steering effect on demand“

The deviation above is well established and broadly known from conventional mechanical systems and generally speaking referred to by expressions, such as: „the equipment is broken, defected or in-operational.“

The dysfunction **Inadvertent Function** of a source component for a dataflow/signal, results in an output deviation of ***Commission failure*** class, and is defined as:

„Nominal function without demand“

Applied to the function of an embedded system, e.g. Braking or Steering, the dysfunction can be named as

„Breaking effect without demand“ or „Steering effect without demand“

The dysfunction **Erroneous Function** of a source component for a dataflow/signal, results in an output deviation of ***Value Failure*** class and is defined as:

„Nominal function on demand performed with wrong intensity“

Applied to the function of an embedded system, e.g. Braking or Steering, the dysfunction can be named as

„A breaking effect on demand does not match the pilots demand“ or
„A steering effect on demand does not match the pilots demand“

[2] Only if not of special interest. This might be different for client/server systems.

Regarding programmable systems, a vast variety of Value Failures Classes can be considered, such as too much-/too little-braking or different breaking effects on different wheels, as well as too much-/too little-steering or steering in the opposite direction.
It is non-confirmable accepted that no functional failures exist within a cyclic-operating computing system, that cannot be mapped to one of the three above listed dysfunctions.

3.3.3 Activities for MBSA

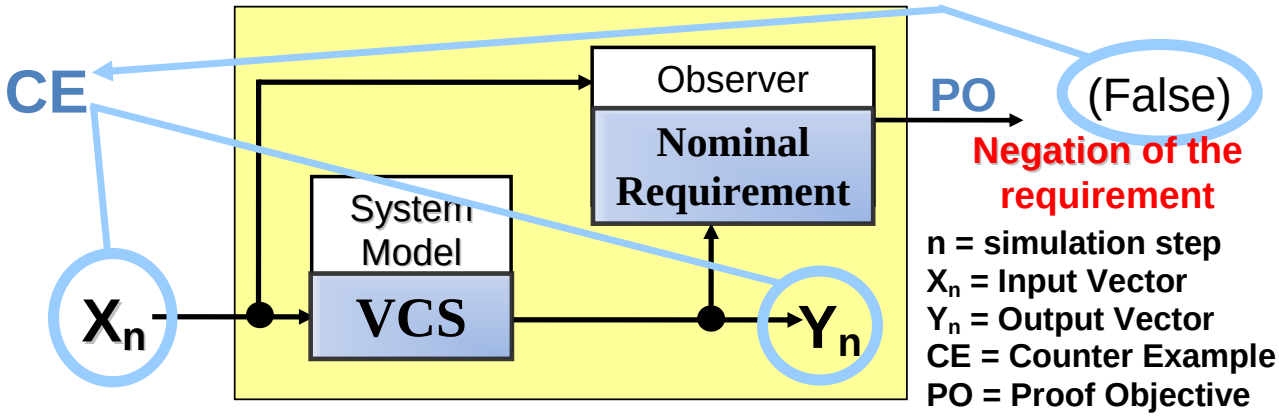

Figure 14 Principle of Model Verification

The MBSE is extended with failure behavior by performing the following actions of model verification in the VCS model, Figure 14:

Action1. Introduce Failure Behavior

Perform a functional failure assessment and map derived failure modes to the dataflow/signal of the MBSE. The additional logics to implement failure modes shall be hidden from the user to avoid cluttering the Simulink model. A failure mode library shall be available that can be used to drag & drop failure modes on the respective Simulink signals. The components of the model can fail to produce correct outputs because of two possible reasons: Wrong Input values or internal computation errors leading to wrong Output values.

Task A. Introduce Model **Input Failures**

Insert falsified values to the Input Vector of the Simulink block/node.

Task B. Introduce Model **Output Failures**

Implement additional failure states according to the three different classes, which lead to a deviation of the output signals for each node/block representing a function, according to the FFA. See section 3.3.2 for details.

Task C. Repeat the task A+B according to the FFA

The implementation of failure modes on the input data and the node/block internal failure behavior (dysfunctions) depend on the syntax of the Simulation Tool in use, here Matlab/Simulink.

Action2. Verify the nominal behavior of the MBSA

Before starting the failure analysis, assure that the nominal behavior is not corrupted through the extension with failure behavior. If the nominal behavior is not verified, than no statement can be made to its faulty behavior, instead anything can happen.

Action3. Select the Observer Node

The Observer contains the VCS Safety Requirement described in the Simulink syntax. The model-checker stops on the first negation of the Prove Objective (PO).

Action4. Start the Failure Assessment

The failure assessment shall be realized by using the Design Verifier of Simulink. Behind the design Verifier a model checker is implemented, that shall collect all the counter-examples, which falsify the Observer output (PO). The model-checker will perform an exhaustive assessment, therefore it is recommended to constrain the analysis for a limited number of failure combinations. Experience with SCADE has shown that it is quite handy to reduce to single failure on the first simulation and after to double, triple or quadruple failure combinations. More than quadruple failure combinations have not shown benefits so far.

3.3.4 Framework for MBSA

In order to achieve an exhaustive failure assessment depicted in Figure 15, the application to control the Simulink Design Verifier needs to start the model checker repeatedly, since he will stop after detection of a counter example and we need to find all effects of single-, double-, triple-, quadruple, etc… failure.

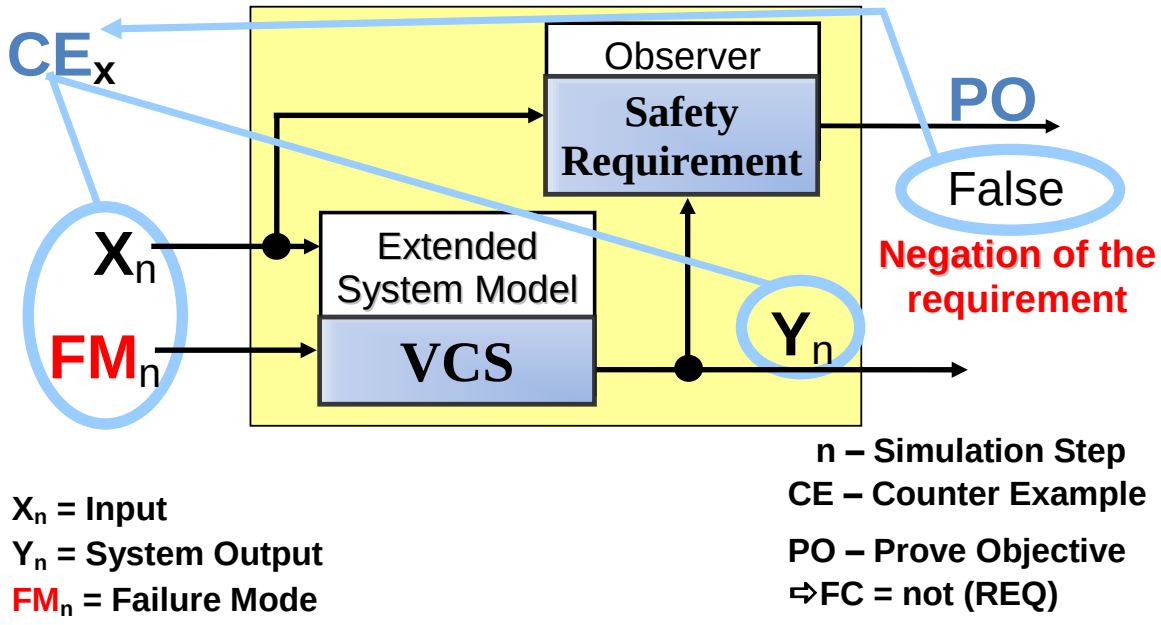

Figure 15 Principles for the model-based Failure Analysis

3.3.4.1 Library of Failure Modes

Failure Modes can be grouped in a library and mapped to model parameter and dataflow similar to the Fault Tree Manager in SCADE 5, see Figure 16 16.

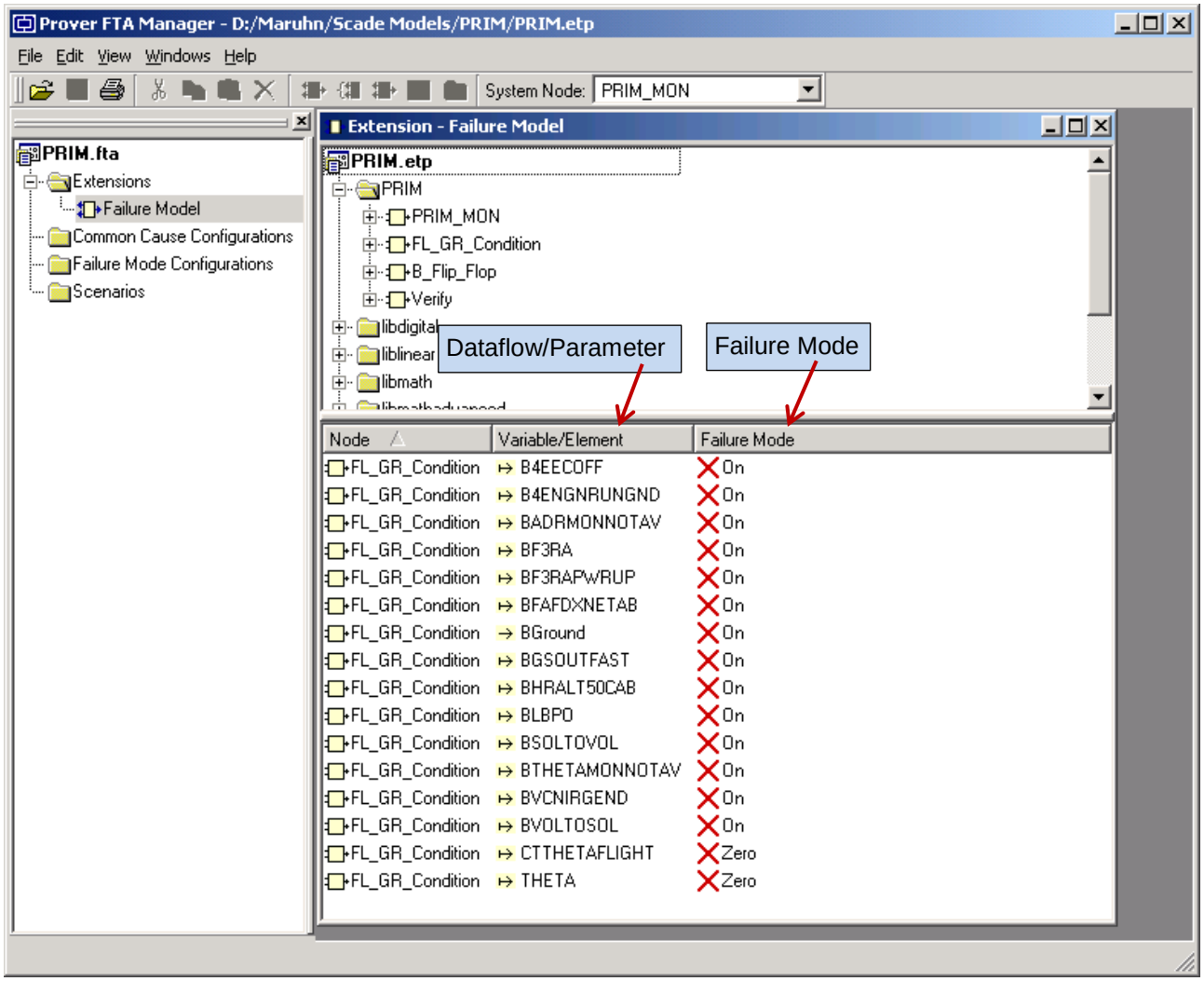

Figure 16 Fault Tree Manager of SCADE

The following Failure Modes need to be implemented in order to create dataflow deviation on the Simulink model input vector and model internal communication, such as CAN and AFDX.

Failure Mode

- **On/Off** [TRUE, FALSE] to falsify Boolean parameters (B).
- **Zero** ['0'] to falsify parameters of natural number (N).
- **Sign Reversal** [(-1) * P_{value}] to falsify Integer and Float values (I, F).
- **Delta** Decrease/Increase [$\pm\Delta$] to deviate Integer and Float parameters (I, F).
- **Stuck_At_Value** [N_n, N_{n+1}, N_{n+2},…] , [I_n, I_{n+1}, I_{n+2},…] or [F_n, F_{n+1}, F_{n+2},…]. After activation of this failure, the parameter is frozen to the current value (N, I, F) of simulation loop n.

3.3.5 Observer and Formalized Safety Requirements

In this chapter all previously created requirements are listed and explained. For a detailed description of how an observer works please have a look in chapter **Error! Reference source not found.**

Override

Ten timesteps after AVS_override is true and ditching is not true, the OVBDV shall be in PO position and the BUV shall be in FO position. This function shall be implemented on the local control application.

Assumption: OVBDV and EFan are not faulty.

Rationale: This is the safe state of the plant. It allows blowing and extraction ventilation by means of differential pressure between mixing unit and avionics bay (blowing side) as well as differential pressure between avionics bay and outside air (extraction side) in case of a loss of the central control application as well as a blowing and extraction fan fault.

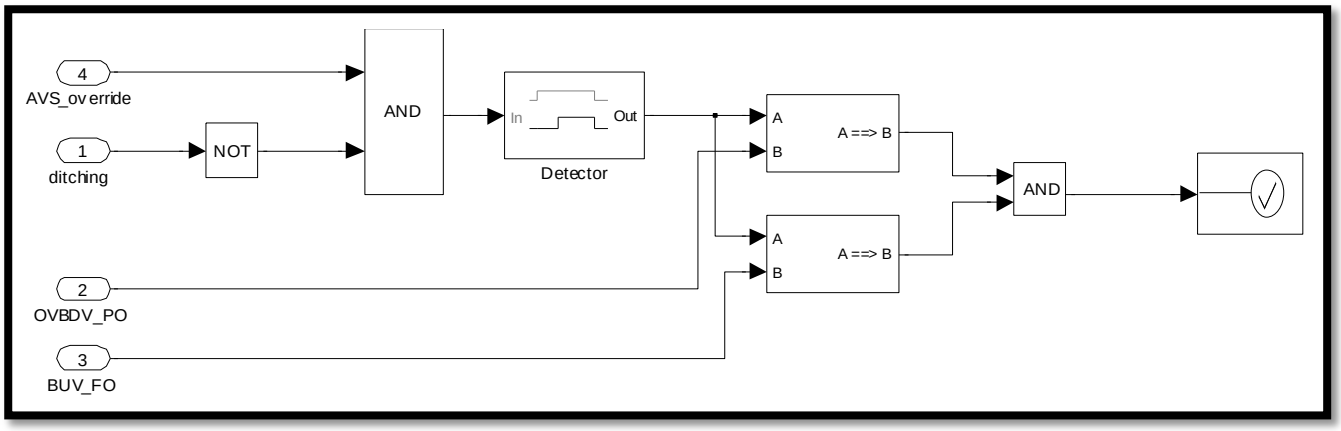

Figure 17 Override Requirement

Closure Request

Ten seconds after closure request is true and smoke is true and on_GND is not true, the OVBDV shall not be in fully open position. This function shall be implemented on the central control application.

Assumption: OVBDV is not faulty.

Rationale: To prevent depressurization during flight.

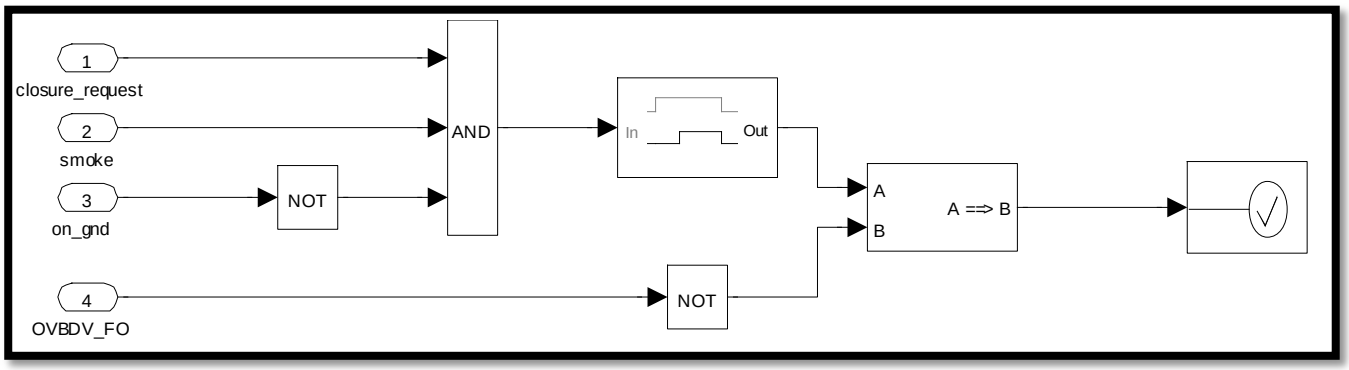

Figure 18 Closure Request Requirement

Blowing Redundancy

Seven seconds after the blowing fan is faulty, the BUV shall be in FO position

Assumption: The BFan is faulty and the BUV is not faulty.

Rationale: To ensure blowing ventilation by mixer unit in case of fan fault.

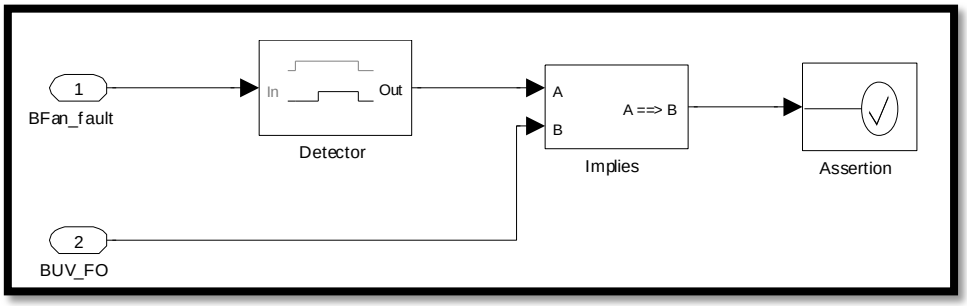

Figure 19 Blowing Redundancy Requirement

Extraction Redundancy

Seven seconds after the EFan is faulty, the OVBDV shall be in PO position

Assumption: The EFan is faulty and the OVBDV is not faulty.

Rationale: To ensure extraction ventilation by differential pressure in case of fan fault

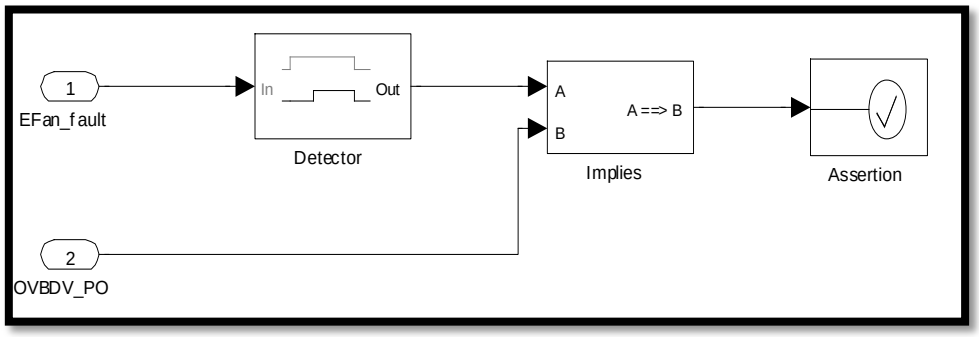

Figure 20 Extraction Redundancy Requirement

Ditching

The OVBDV shall be in FC position ten seconds after ditching is true. This is a DAL A function shall thus be implemented on the local control application.

Assumption: OVBDV and INBDV are not faulty.

Rationale: To prevent ingestion of water via the skin valve (OVBDV) in case of ditching.

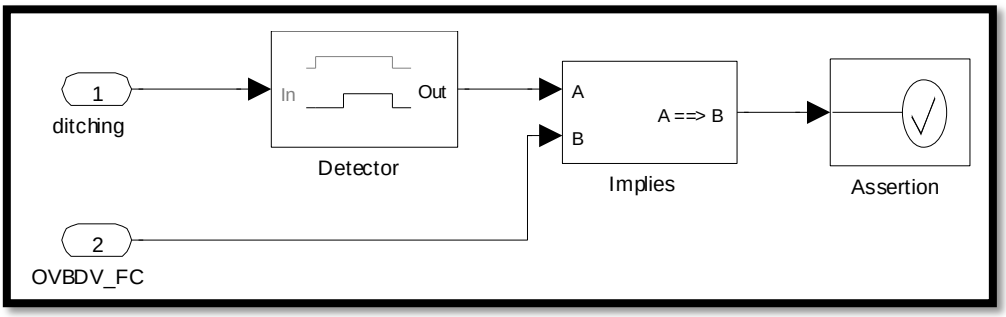

Figure 21 Ditching Requirement

Smoke

Ten seconds after smoke is true and closure request is not true and ditching is not true and AVS_override is not true, the OVBDV shall be in FO position and the INBDV shall be in FC position. This function shall be implemented on the central control application.

Assumption: OVBDV, INBDV and EFan are not faulty.

Rationale: To dump smoke overboard at the highest flow rate possible.

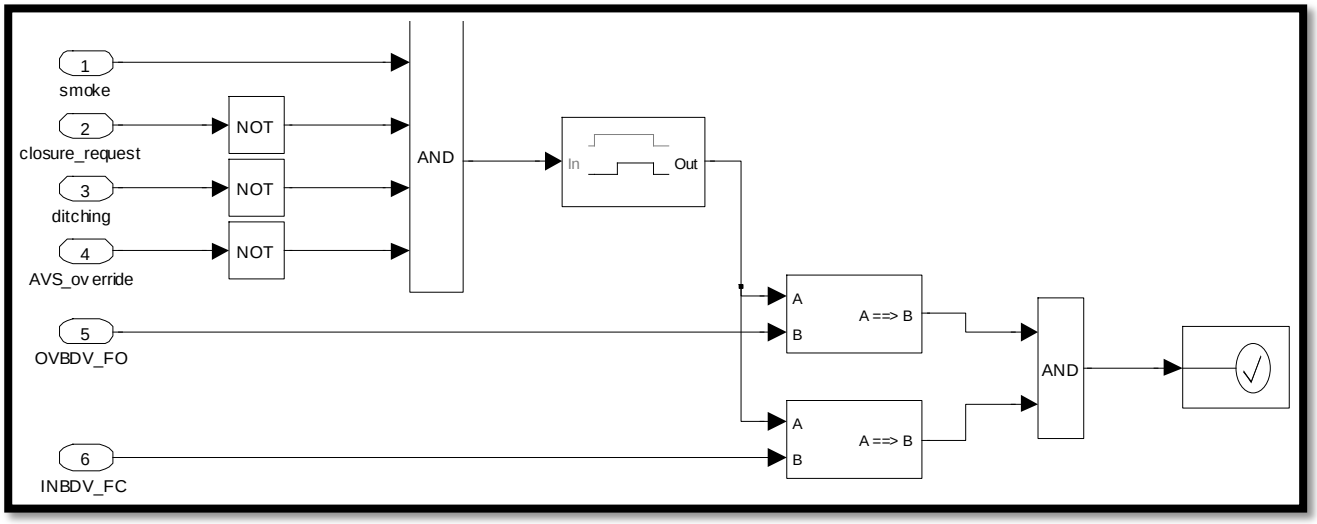

Figure 22 Smoke Requirement

Extraction Path

If the EFan is not off, the INBDV and the OVBDV shall not be in FC position at the same time.

Assumption: The EFan is not faulty.

Rationale: To prevent damage to the extraction fan.

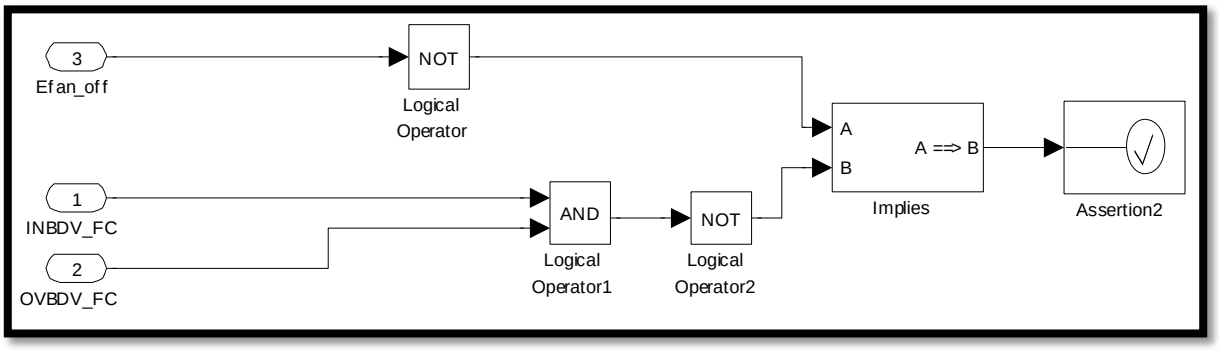

Figure 23 Extraction Path Requirement

Back-up Performance

10 seconds after the BFan is faulty or off, the BUV shall be in FO position and the inflow into the avionic compartment shall be less or equal than 1.5 KG/s and greater or equal than 1 kg/s.

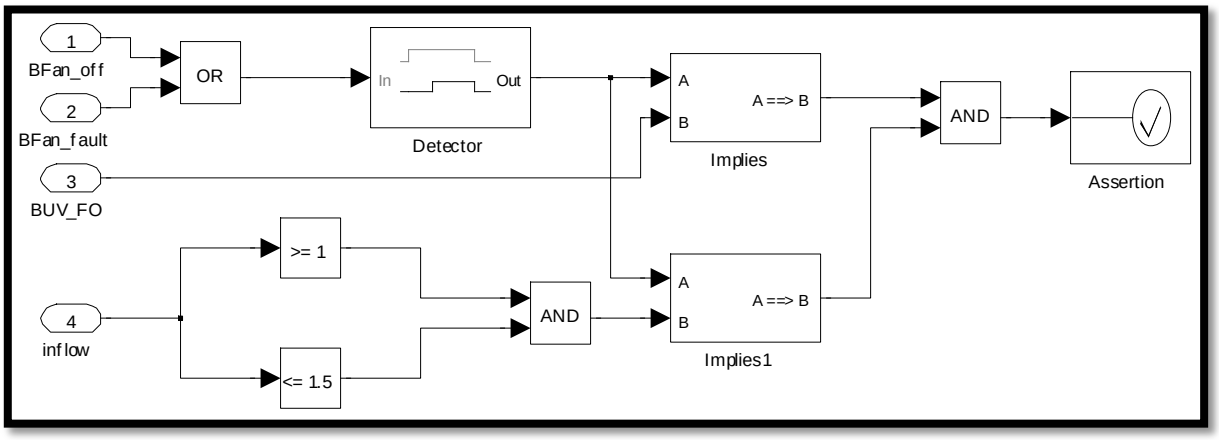

Figure 24 Back-up Performance Requirement

Blowing Airflow Performance

50 seconds after the BFan is not faulty and not off, the inflow into the avionic compartment shall be less or equal than 1.5 KG/s and greater or equal than 0.2 kg/s.

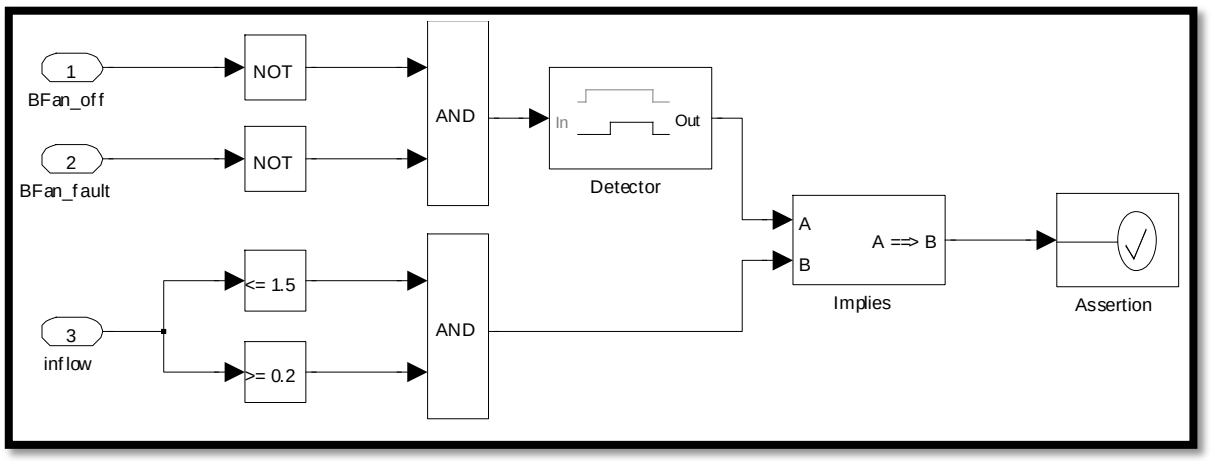

Figure 25 Blowing Airflow Performance Requirement

Smoke Propagation

If the EFan is not off and not faulty, the leakage air mass flow from the avionic compartment shall be positive.

Assumption: Outgoing air mass flow from the compartment is measured as positive. Rationale: To prevent smoke propagation into other compartments.

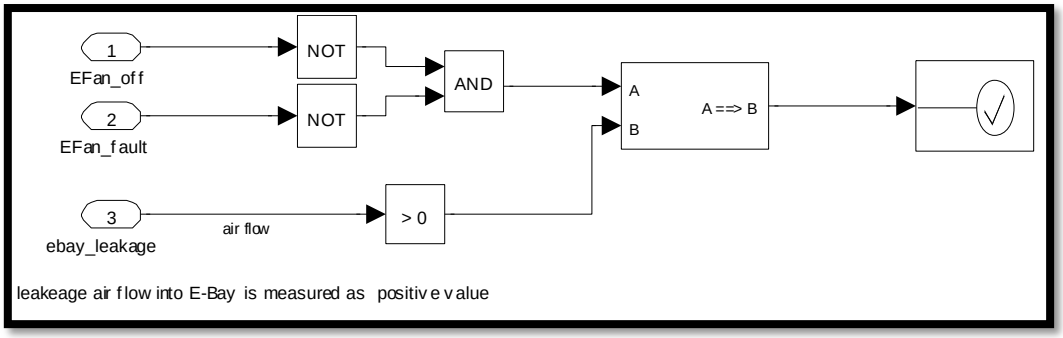

Figure 26 Smoke Propagation Requirement

Blowing Redundancy Loss

Five seconds after the BUV is in SC position and the application has diagnosed the BUV to be in status 2 (SC position), the alert blowing_redundancy_loss shall be latched.

Assumption: BFan is not faulty and BUV is faulty.

Rationale: To indicate decreased redundancy.

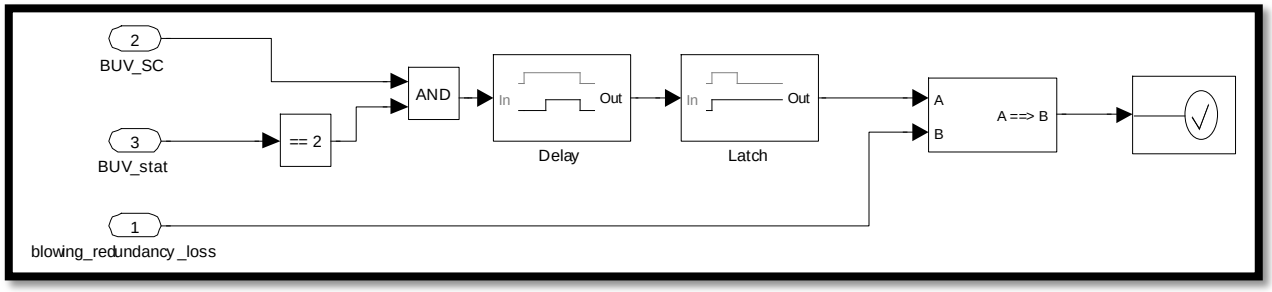

Figure 27 Blowing Redundancy Loss Requirement

Extraction Redundancy Loss

Five seconds after the OVBDV is in SC position and the application has diagnosed the OVBDV to be in status 3 (SC position), the alert extraction_redundancy_loss shall be latched.

Assumption: BUV is not faulty.

Rationale: To indicate decreased redundancy.

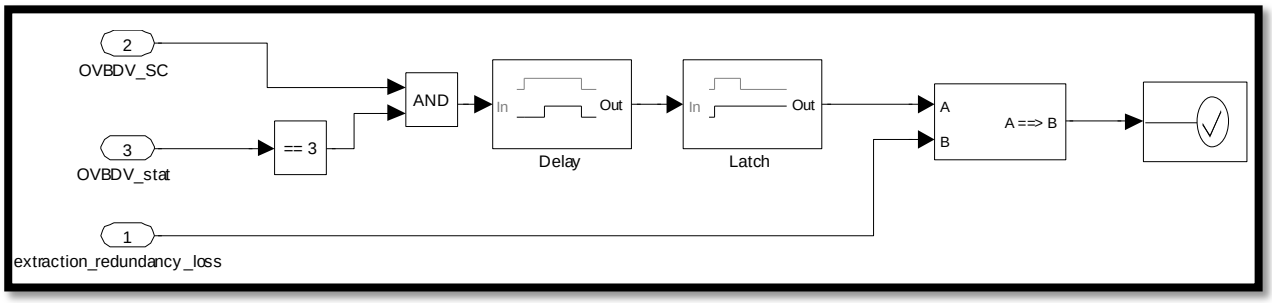

Figure 28 Extraction Redundancy Loss Requirement

Blowing Fault

Five seconds after the BUV is in SC position and the application has diagnosed the BUV to be in status 2 (SC position) and the BFan is faulty, the alert blowing_fault shall be latched.

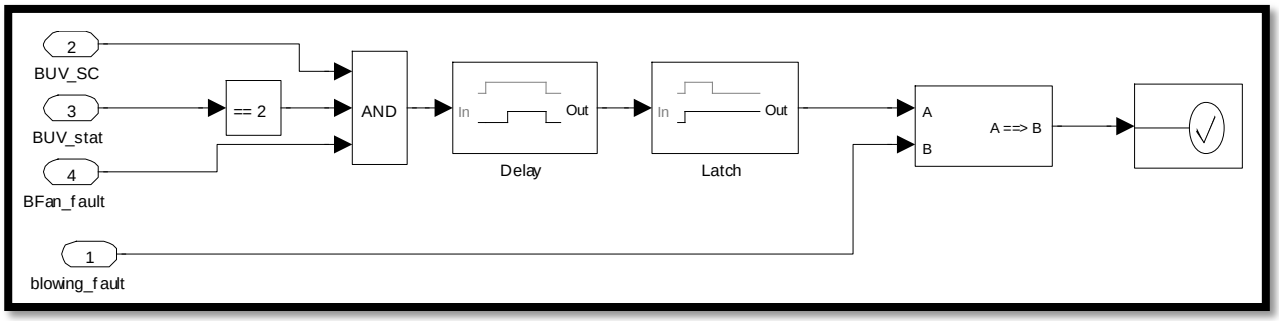

Figure 29 Blowing Fault Requirement

Extraction Fault

Five seconds after the OVBDV is in SC position and the application has diagnosed the OVBDV to be in status 3 (SC position) and the EFan is faulty, the alert extraction_fault shall be latched.

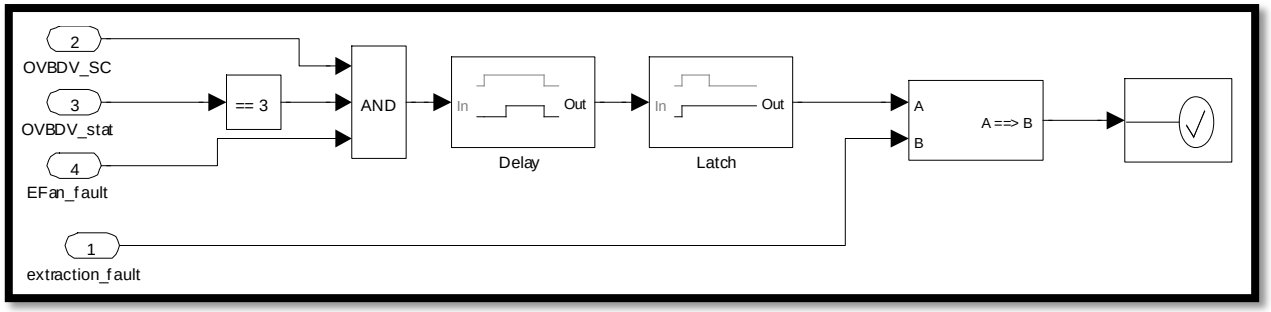

Figure 30 Extraction Fault Requirement

Avionics Ventilation Fault

Five seconds after the BUV is in SC position and the application has diagnosed the BUV to be in status 2 (SC position) and the BFan is faulty and the OVBDV is in SC position and the application has diagnosed the OVBDV to be in status 3 (SC position) and the EFan is fault, the alert blowing_fault shall be latched.

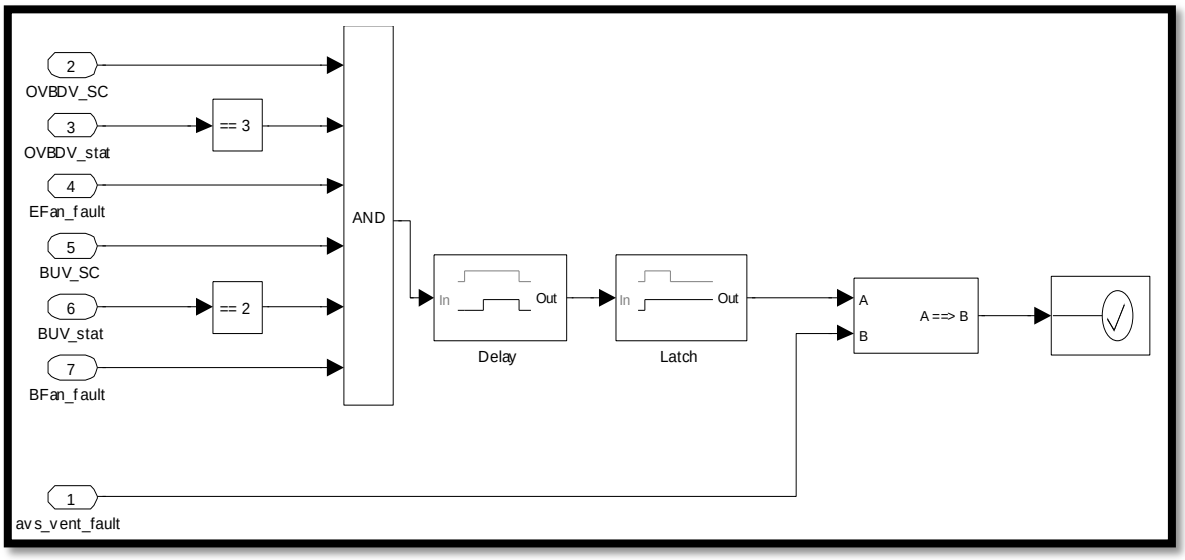

Figure 31 Avionics Ventilation Fault Requirement

3.3.6 Design Verifier

The Design Verifier is a MATLAB/Simulink embedded function upon enabled by dedicated licences. With the Design Verifier it is possible to identify hard to find design errors, generate test cases and verify the design against requirements.

The Design Verifier uses formal methods to discover under which conditions specific dynamic execution scenarios can occur. The following failures can be detected by the Design Verifier: integer overflow, divide by zero, faulty logic and assertion violations.

Because of the quantity of input signals and better overview, the signals can be packed into bus-signals. The design verifier toggles the input signals to find a signal procedure to disprove generated requirements. In order to create a prove objective, the requirements to be checked are integrated in the model by Simulink formalisms – the entire model is executed by the Design Verifier.

To use the Design Verifier it has to be ensured that no SimScape elements are implemented in the model. The Design Verifier is not compatible to that tool.

Implement requirements into the system model could be done by adding different kind of system blocks into the model. These blocks are included into the model in the “Top-Level Requirements”-Subsystem. The needed blocks are:

- Detector-Block
- Implies-Block
- Assertion-Block
- Logic-Blocks

With these four kinds of blocks it is possible to design requirements as shown in Figure 32. An action and the corresponding reaction are defined. The Detector-Block sends the incoming action time-delayed further to the Implies-Block.

The Implies-Block checks if the reaction is true. If the reaction is true, the output of the Implies-Block is logic zero. If the reaction is false, the Implies-Block activates the Assertion-Block and a warning is shown on the desktop.
The Design Verifier looks at the beginning of the simulation for the Assertion-Blocks and tries to falsify the requirements. These Blocks are also usable without the Design Verifier in a normal simulation.

For this example it is expected, that the Backup Valve moves to fully open if a failure of the Blowing Fan is detected.

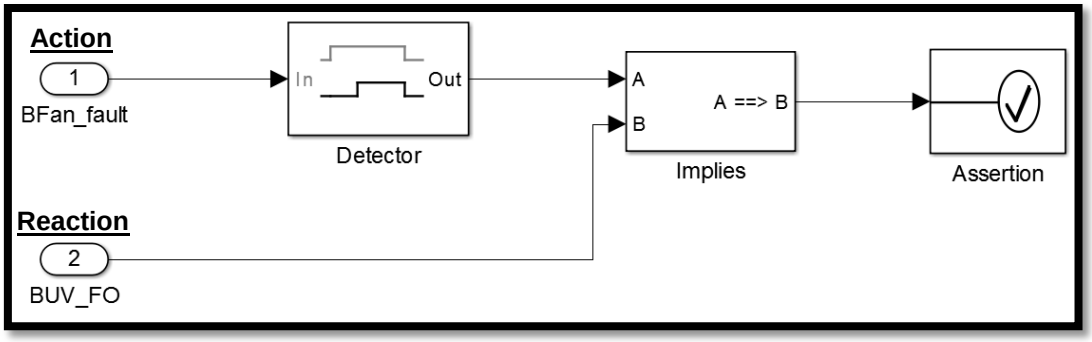

Figure 32 Requirement

In addition to that it is possible to specify values. This can be done by adding Assumption-Blocks to the model to check special conditions or to simplify the model to accelerate the process. The Assumption-Blocks are shown in Figure 33. In this example four assumptions are set to false and one is deactivated. Furthermore it is possible to set specified values.

If there is any design failure detected in the described way, a failure message triggered by the Assertion-Block appears on the desktop. In addition it is possible to re-simulate the procedure that leads to the detected failure and have a detailed look at the behavior of the system. This function supports the localization of design failures.

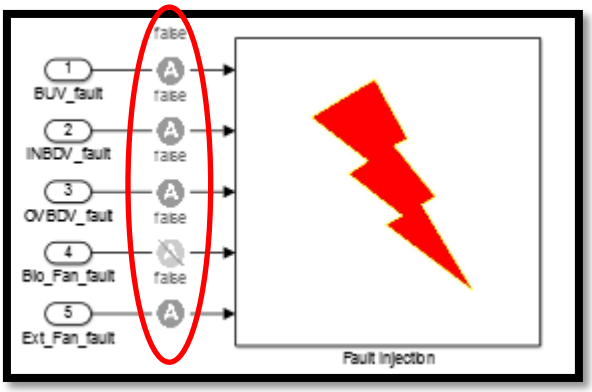

Figure 33 Assumption Blocks

Another design verifier function is the signalbuilder block. It can be generated if a failure procedure is detected. In the signalbuilder block, each input and failure signal can be displayed. This block is shown in Figure 34.

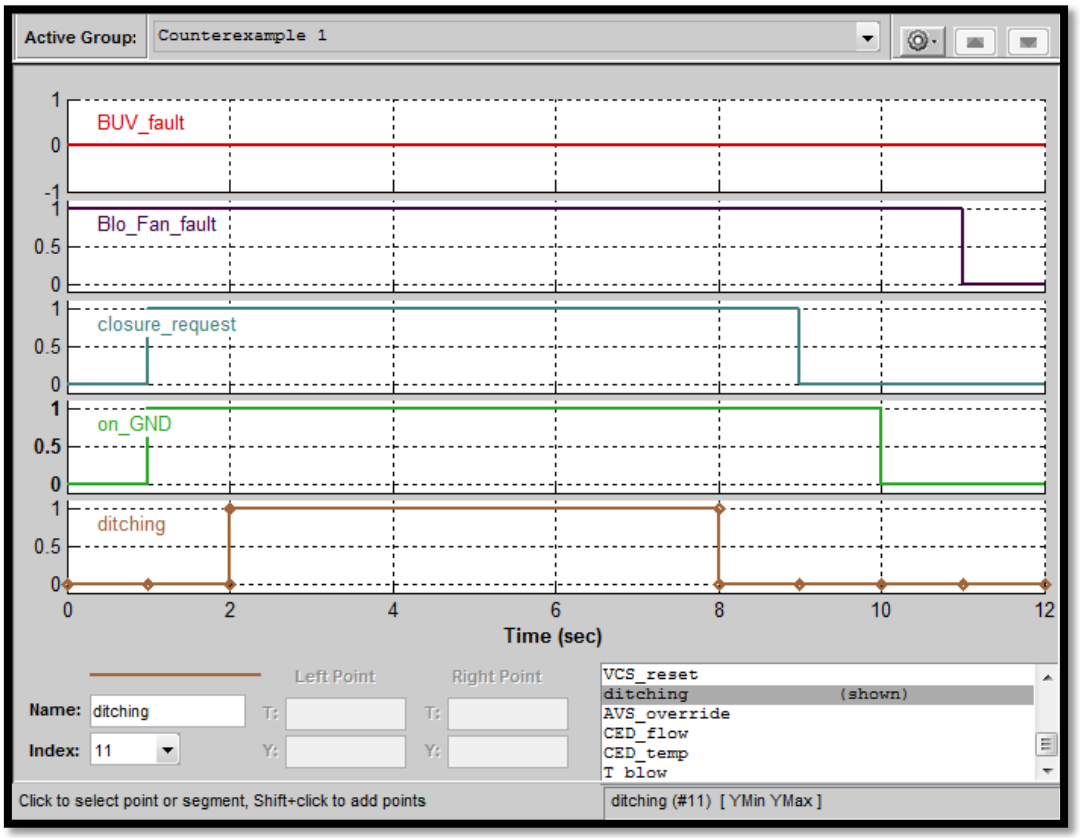

Figure 34 Signalbuilder Block

As you can see, a timing diagram is shown for all selected signals. With this signalbuilder block it is possible to analyze the model by simulating it step by step.

3.4 Stakeholders & Roles

Please identify the stakeholders and their roles with respect to the individual activities.

Stakeholders	**Role**
Safety Engineer	Responsible for the system safety analysis
Design Engineer	Responsible for the system specification
System Supplier	Responsible to implement the system specification
Authorities	Approval for certification

4 Identification of Engineering Methods

The following methods describe in more detail the cross-domain engineering steps of the system design and safety assessment activities. With respect to various models, data for e.g. analysis methods, tools and associated configuration data – engineering methods can be characterized by 5 generic entities depicted in Figure 35.

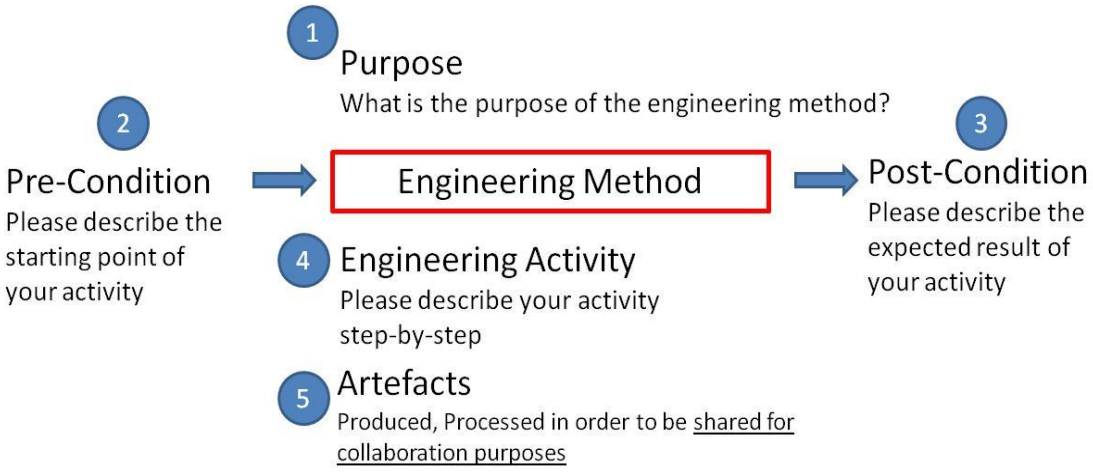

Figure 35 Entities of Engineering Methods

As a first approach in the VCS use case – the entities in Figure 36 are exemplified as the design- and failure models in Matlab/Simulink as well as the Model Checker in Simulink called Design Verifier (Prover Plug-In). The injection of failures at appropriate state transitions or signal flow is performed by an additional underlying framework (presumably based on Eclipse) which also configures the model according to the needs of Design Verifier such as model characteristics and proof objectives. The Design Verifier interrupts at the first counterexample found consequently it has to be re-started or iterated with other failures automatically in order to retrieve all sets of failures. Normally, a system design is robust with respect to single failures, but double or triple failures (with sequence dependency) may violate the proof objective (e.g. safety requirement) resulting in iterative generated interrupts each creating a set of different failure. The lowest number of failures sets violating a proof objective is called a minimal cut set. The safety analysis is completed by computing the probability of the minimal cut set (or a number of minimal cut set) by the safety tool Fault Tree Plus.

The starting point (2) is a catalogue of failures modeled as a Simulink blocks which are selected automatically by the underlying framework also in different sequences. Integration rules ensure in the framework proper failure injection in the design model (2) at preselected insertion points. Various proof objections (2) can be also selected by the framework to reduce model and computing complexity. The Engineering Activity (4) is the concurrent system design with safety assessment within a model based approach. Artefacts (5) to be shared for collaborative purposes are the design models as well as the sets of failures for a dysfunctional behavior model. The

expected results of the engineering activities are improved design models of nominal behavior in Matlab/Simulink as well as probability models of dysfunctions.

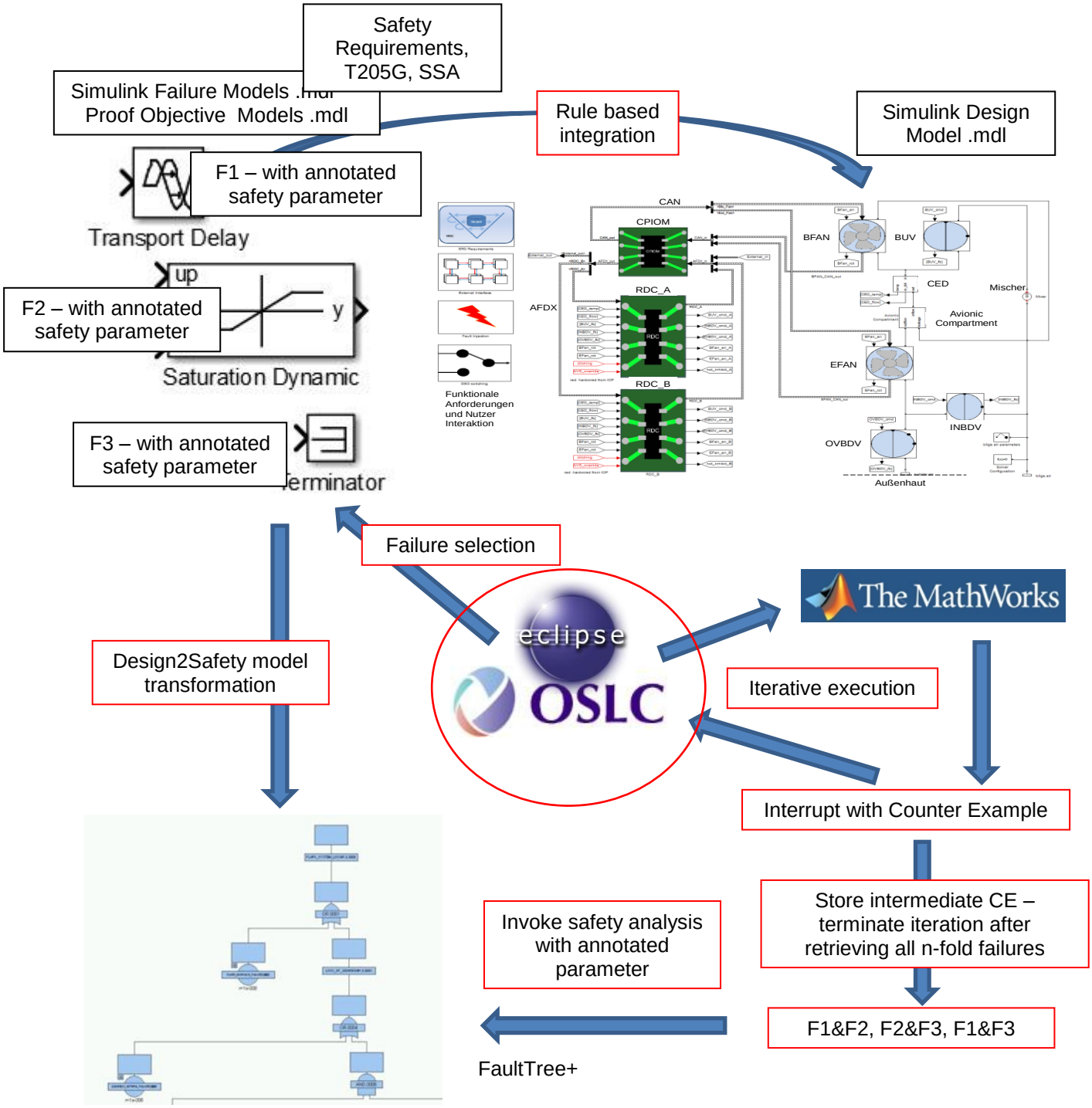

Figure 36 Tool Chain and Interoperability

(Red-marked operations shall be orchestrated by Eclipse and/or OSCL)

4.1 Requirements on Methods and Tools

Dedicated requirements for Interoperability Standards and tool-chain implementations are derived and defined for the next version of this document.

4.2 Requirements for Interoperability Standard

Dedicated requirements for Interoperability Standards and tool-chain implementations are derived and defined for the next version of this document.

5 Glossary

AFDX	Avionics Full Duplex
AVS	Avionics Ventilation System
BFan	Blowing Fan
BUV	Back-Up Valve
CAN	Controller Area Network
CE	Counter Example
CPIOM	Core Processing Input/Output Module
DAL	Design Assurance Level
ECS	Environmental Control System
EFan	Extraction Fan
EFC	Equipment Failure Condition
FC	Failure Condition
FFA	Functional Failure Analysis
FHA	Functional Hazard Analysis
FMEA	Failure Mode & Effect Analysis
FMES	Failure Mode & Effect Summary
FTA	Fault Tree Analysis
IHA	Intrinsic Hazard Analysis
IMA	Integrated Modular Avionics
INBDV	Inboard Valve
MBSA	Model-based Safety Analysis
OVBDV	Overboard Valve
PO	Proof Objective
(P)SSA	(Preliminary) System Safety Assessment
PTS	Purchaser Technical Specification
RDC	Remote Data Concentrator
SIRD	System Installation Requirement Dossier
SRD	System Requirement Dossier
ZSA	Zonal Safety Analysis
VCS	Ventilation Control System