

<h1>Enhanced Safety Assessment For Complex Systems</h1>		
<h2>TECHNICAL REPORT</h2>		
TITLE:	D18 Results of methodology application on cycle #2 case studies	
WORKPACKAGE:	WP 4 (related to Milestone M4)	
TIMESCALE:	M33	
SCOPE:	Task 4.3 “Apply Final Methodology to Case Studies” Task 4.4 “Collection of results and final evaluation of methodology”	
KEYWORDS:	Enhanced Safety Assessment for Complex Systems ESACS	
ISSUE DATE:	28/10/2003	
Compiled:		
	Christian Bognol, Airbus France	
Technical Approval:		
	Antonella Cavallo, Alenia Aeronautica S.p.A.	
Issue Authorisation:		
	Luigi Trotta , Alenia Aeronautica S.p.A.	
© All rights reserved. This document is supplied by the specific ESACS work package quoted above on the express condition that it is treated as confidential to those specifically mentioned on the distribution list. No use may be made thereof other than expressly authorised by that work package leader.		

Named Distribution Only

DISTRIBUTION LIST		
Copy ¹ type	Company and Location	Recipient
@, C, D	Aeronautics Unit - European Commission Rue Joseph II, 79 – Brussels	M. Brusati
M	Alenia Aeronautica S.p.A.(ALA)	Antonella Cavallo, Luigi Trotta
@	Airbus France (Airbus F)	Jean Pierre Heckmann, Christian Bougnol, Sylvain Metge
@	Airbus UK (AUK)	Benita Lawrence, Chris Papadopoulos
@	Airbus Deutschland (Airbus D)	Matthias Bretschneider
@	Saab AB (SAAB)	Per Persson
@	Societa' Italiana Avionica (SIA)	Marco Terrone, Massimo Cifaldi, Laura Valacca
@	Istituto Trentino di Cultura (ITC-IRST)	Alessandro Cimatti, Adolfo Villafiorita
@	Office National d'Etudes et de Recherches Aérospatiales (ONERA)	Christel Seguin, Charles Castel
@	Kuratorium OFFIS e.V. (OFFIS)	Werner Damm, Thomas Peikenkamp, Andreas Luedtke
@	Prover Technology AB (PROVER)	Marten Saflund, Ove Åkerlund

¹ M = Master copy, @ = Email, C = Controlled copy (paper), D = Electronic copy on disk

Named Distribution Only

RECORD OF REVISION		
Issue	Date	Reason for Revision
A	28-10-2003	Issue A

1. TABLE OF CONTENTS

1.	TABLE OF CONTENTS.....	4
2.	RELATED DOCUMENTS AND LIST OF ABBREVIATIONS	4
2.1.	Related Documents	4
2.2.	List of Some Main Abbreviations.....	5
3.	INTRODUCTION	6
4.	TESTING CONTEXT AND INFORMATION.....	7
4.1.	Platforms.....	7
4.2.	Testers.....	8
4.3.	Case studies.....	9
4.4.	Questionnaire	9
4.5.	Test plan.....	9
4.6.	Link between the test plan and the questionnaire	10
5.	COVERAGE OF THE TESTS	12
6.	LESSONS LEARNT FROM THE TESTS.....	14
6.1.	Highlighted points.....	14
6.1.1.	Abstraction on the system model	14
6.1.2.	Failure injection	23
6.1.3.	Functional and Safety Property Checking.....	29
6.1.4.	Fault Tree and Sequence Generation	33
6.1.5.	Architecture assessment using Safety Patterns	38
6.1.6.	Bottom up FMEA	42
6.2.	View on the evaluation criteria questionnaire answers.....	43
6.2.1.	Method: logical issues.....	43
6.2.2.	Software engineering issues.....	43
6.2.3.	User interface issues	44
6.2.4.	New features issues.....	45
6.2.5.	Safety process issues.....	46
6.2.6.	Effectiveness issues	47
6.2.7.	Exploitation issues	47
7.	CONCLUSION.....	48
	APPENDIX 1 : ALA/SIA CYCLE 2 TEST RESULTS.....	49
	APPENDIX 2 : SAAB/PROVER CYCLE 2 TEST RESULTS	50
	APPENDIX 3 : AID CYCLE 2 TEST RESULTS.....	51
	APPENDIX 4 : AIUK CYCLE 2 TEST RESULTS.....	52
	APPENDIX 5 : AIF/ONERA CYCLE 2 TEST RESULTS	53
	APPENDIX 6 : QUESTIONNAIRE ANSWERS	54

2. RELATED DOCUMENTS AND LIST OF ABBREVIATIONS

2.1. Related Documents

1. Use Case document, ESACS/09/000011/A/WP2/RPT dated 24/10/2001
2. Material for case studies (D9), ESACS/04/000020/A/WP3/RPT dated 24/06/02
3. Preliminary methodology and tool-set for improving the safety analysis process of complex systems (D11), ESACS/10/000021/A/WP2/RPT dated 10/07/02
4. Definition of a questionnaire of the effectiveness of the methodology (D10), ESACS/04/000018/A/WP3/RPT dated 30/04/02
5. Test Plan document
6. D16/D17 – Methodology, Techniques and Tools - Refinement after Application in Cycle #1, ESACS/09/000033/A/WP2/RPT dated 28/05/03
7. D14 – Results of methodology application on cycle #1 case studies, ESACS/02/000031/A/WP4/RPT dated 09/04/03

2.2. List of Some Main Abbreviations

BDD	Binary Decision Diagram
CE	Counter Example
CM	Counter Model
ESM	Extended System Model
FM	Failure Mode
FTA	Fault Tree Analysis
FoSaM	Formal Safety Model
MCS	Minimal Cut Set
NuSMV	New Symbolic Model Checker
PCO	Point of Control and Observation
PPI	Prover Plug-In
SAT	Safety Analysis Task
SE	Safety Engineer
SM	System Model
SPPI	SCADE and Prover Plug-In
SR	Safety Requirement
S/R	Safety Requirement and Top Level Event
SSA	StateMate Safety Analysis
STSA	StateMate Safety Analysis
STVE	StateMate Verification Environment
TLE	Top Level Event
SPS	Secondary Power System
SCP	Simulation Control Program

3. INTRODUCTION

This document focuses on an analysis of the final methodology and tools application results within the European research project ESACS (Enhanced Safety Assessment for complex Systems).

The ESACS methodology is based on the use of system modelling tools (e.g. Statemate, Scade,...) and formal verification techniques (e.g. formal languages, model checkers,...), which have been introduced in the industry practices initially for the designers of complex aircraft systems. These tools and techniques allow to propose new links with safety analysis tools and to investigate the improvements they can bring to the safety analysis process. These aspects are included in [3] underlining a preliminary ESACS methodology and tool-set and in [6] describing the refinements of methodology and tools after the 1st application cycle.

The 1st cycle was performed on the preliminary methodology and tool-set and necessitated a test plan (see [5]) which facilitated the collection of the lessons learnt and the feedbacks about the effectiveness. The lessons learnt were based on the preliminary results of the tests which were organised per ESACS use cases defined in [1]. The effectiveness criteria were established by the ESACS partners under a questionnaire form (see [4]). The tests were conducted on system examples proposed to refine the tool candidates and case studies detailed by the industrial partners (see [2]). The results were included in [7].

The feedback from 1st cycle led to the refinement of methodology and tools [6].

This document is relevant to cycle 2 application of the resulted final methodology and tool-set.

This document reminds in chapter 4 the context and information which influence the scope of the testing. Then, the next chapter shows synthetically the coverage of the tests regarding both the use cases and the tool-set separated into different platforms. Then, the chapter 6 summarises the lessons learnt revealed by the testing and gives a view on the evaluation criteria questionnaire answers.

In appendixes, detailed information (test reports and answered questionnaires) can be found for trace-ability purposes.

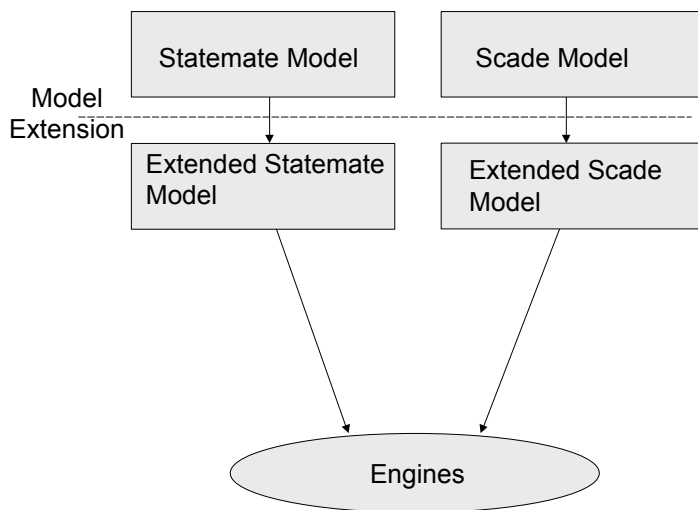
The content of this report will evidence the new results/expectations/conclusions of cycle 2 in respect to cycle 1 and new aspects for future activities and application in the industrial process. When necessary information produced during cycle 1 tests are reminded for a better understanding of the results and also because some of the conclusions are still valid. *When an additionnal information from cycle 2 were interesting to highlight they were written in italic in the relevant place in this document.*

The document is related to project milestone M4.

4. TESTING CONTEXT AND INFORMATION

4.1. Platforms

The document D11 “Preliminary methodology and tool-set for improving the safety analysis process of complex systems” explains the alternative followed by the technology provider partners to mitigate the risks and guarantee the capability to do safety analyses with the ESACS tools within the time limit of M2. This alternative is reminded below and could be generalised depending on the different input model formats:



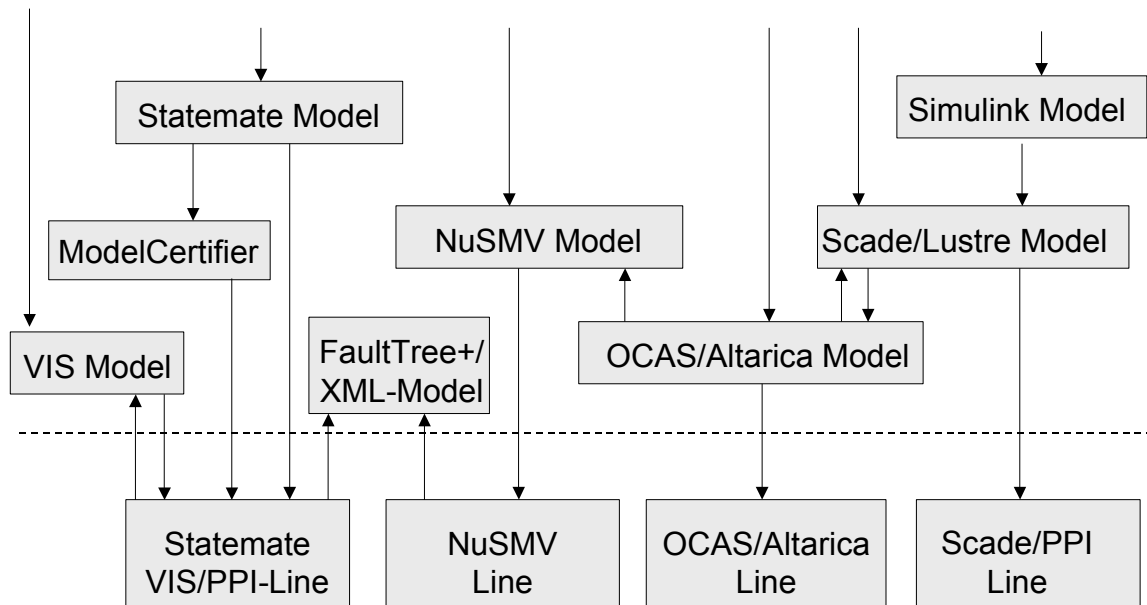
In this alternative all modelling, extensions and SR capturing is done in a same environment. So in this case no translation is needed, neither to a common format nor from a common format to the verification tools, which makes it easier to utilise already existing couplings to the verification tools.

One important aspect of the tools impact on the methodology is the user’s needs regarding the safety activities. In particular, the tool set has to be able to cope with the types and complexities of the case studies defined by the users. This leads to develop 4 platforms, called for simplification purposes:

- Offis platform, based on a Statestate implementation line,
- IRST platform, based on a NuSMV implementation line,
- Prover platform, based on a Scade/PPI implementation line,
- Onera platform, based on an OCAS/Altarica implementation line.

The hereafter drawing illustrates the links between these platforms and the industrial inputs available, considering the existing (or developed within ESACS) converters at the time of cycle 1.

INPUTS AND CONVERTERS



IMPLEMENTATION LINES

It had to be said that, in parallel to the activity of development of new facilities, Technology Partners done some job towards integration for the cycle 2 testing.

Examples are the following:

- *translation from Scade into Altarica possible through the Lustre language*
- *translation from Altarica to NuSMV*
- *import-mechanism for definitions and verification patterns from the ModelCertifier tool implemented in the OFFIS STSA platform*
- *PPI integrated in the OFFIS STSA platform as an alternative verification engine*

4.2. Testers

According to the work plan, the industrial partners managed the testing of the different platforms. Considering the prototyping level of the platforms, the support of the technological provider partners could be recommended.

Basically, the testing was carried out by:

- Alenia and SIA on the NuSMV Line and on the Statestate Line,
- SAAB and Prover on the Scade/PPI Line,
- Airbus Deutschland and Offis on the Statestate Line,
- Airbus Deutschland and Prover on the Scade/PPI Line
- Airbus UK on the Scade/PPI Line,
- Airbus France and Onera on the OCAS/Altarica Line.

4.3. Case studies

The Case Studies used during this different testing are described in the Case Studies document D9, where each industrial proposed a model to be considered because of:

- the heterogeneity of aircraft systems (electrical components, programmable controllers and possibly mechanical components),
- the safety aspects like fault-tolerance (monitors, redundancy management etc.),
- the granularity of model – functional level,
- the complexity: non-trivial model, but not too many components (ESACS workload).

Moreover, the examples used to test the technology candidates during the WP2 (a simple tank example model and more complex Aircraft Hydraulic System models) are also considered as Case Studies. They are especially interesting to detail the assessment of some results:

- either because the model can be also manageable by hand (tank example),
- or because the models for one particular implementation line can be translated in other input models for other implementation lines allowing to compare the different tools used in the different platforms.

As a first conclusion, the compared results, where available, are similar and the differences of presentations are reviewed by the Technology Provider partners to improve the internal functions of their platform for the second cycle of the methodology application.

4.4. Questionnaire

During the Work Package 3, evaluation criteria on the effectiveness of the ESACS methodology and tools were browsed under a questionnaire format in the document D10. This questionnaire was answered by the testers after the cycle 1 methodology application. The different answers are included in appendix 6.

For cycle 2 it was decided to work directly on the synthesis included in the main document (see para 6) proposing what is changed according to the cycle 2 tests conducted. Anyway, for traceability reasons some Partners included the changes also in the questionnaire, updating it for cycle 2.

4.5. Test plan

In cycle #1 before starting the testing, a test plan was proposed as a guideline to assess the different use cases of the ESACS methodology.

In cycle 2 partners could follow the same test plan with some adaptations sometimes:

- *the detailed test numbering could be more difficult to follow and the results could be grouped per ESACS use cases.*
- *for some use cases tests could be done following the evolution of the platforms (platform version).*

In these cases the appendix with the test results is organised according to these adaptations.

The detailed results of the tests are included in Appendix 1 to 5.

4.6. Link between the test plan and the questionnaire

The assessment of the methodology application is driven by the answers to the questionnaire (see chapter 6.2).

As mentioned before, it is also clear that the direct test results collected according to the test plan and the answers of the questionnaire are not independent: the rationales of the answers may be contained amongst the results of these tested use cases.

There is obviously a link between the material from the testing and the answers to the questionnaire.

The matrix below describe these links by recapping trace-ability of the various use case tests of the ESACS platform & methodology with regard to the questions highlighted in the questionnaire.

Named Distribution Only

Question N.	Tests
8.1. Method: Logical Issues	
1	Use Case 5
2	Use Case 5
3	Use Case 5
4	Use Case 5
5	Use Case 4
6	Use Case 4
7	Use Case 1,3,4,5
8.2. Implementation: Software Engineering Issues	
1	Use Case 1,2
2	Use Case 4
3	Use Case 4
4	Use Case 4
8.3. User interface	
1	All tests
2	
A	Use Case 4
B	Use Case 4
C	Use Case 1
3	All Tests
4	Use Case 1,3,4,5
(Only I – II)	Use Case 1
8.4. New features w.r.t. the traditional approach	
1	Not Applicable. These questions require an assessment on traditional safety/reliability analysis approach
2	Use Case 3,4,5
8.5. Safety Process	
1	Use Case 1,5
2	Use Case 3,4,5
3-4	All Tests
8.6. Effectiveness	
1-3	Use Case 4
4-5	Use Case 3,4,5
8.7. Exploitation and 8.8 Comparison with other technical approaches	
1-2	No trace-ability with some particular ESACS platform / methodology tests

Named Distribution Only

5. COVERAGE OF THE TESTS

The following matrix table shows the coverage of the tests carried out after the cycle 2 methodology application on the Use Cases. The use cases titles are reminded after the matrix.

At last, the legend of the table does not show the extension of the tests, e.g. a Use Case done partially by a test (labelled “P”) can lead to an amount of different detailed tests allowing to cover largely a methodological topic. The following chapter 6.1 will translate the lessons learnt from the tests which will underline the importance of these topics.

Use Case	1.1	1.2	1.3	2.1	2.2	3.1	3.2	3.3	3.4	4.1	4.2	4.3	4.4	4.5	4.6	4.7	5.1	
Platform Tested By																		
Offis AL/SIA	Y		Y			I								Y		I	Y	
IRST AL/SIA	Y		Y			I								Y		I	Y	
Prover SA/Prover	Y		Y	P	P	I			P	P	P		P	P	P	I	P	
Offis AD/Offis	Y		Y	P		I							P	I	I	Y	Y	
Prover AD/Prover	Y		Y	P		I							P	I	I	Y	Y	
Prover AUK	Y		Y	P	P	I			P	Y	Y		P	P	P	I	P	
Onera AF/Onera		Y	Y			I	Y	Y	Y					Y		Y	Y	
Total:	Y	Y	Y	P	P	I	Y	Y	Y	Y	Y	Y	N	P	Y	P	Y	Y

Legend:

Y = for a Use Case investigated

N = No, or nothing, for a non tested Use Case

P = Use Case investigated Partially by the test(s)

I = Use Case Included in another Use Case(s)

AL/S: Alenia and SIA

SA/Prover: SAAB and Prover

AD/Offis: Airbus Deutschland and Offis

AD/Prover: Airbus Deutschland and Prover

AF/Onera: Airbus France and Onera

AUK: Airbus UK

Named Distribution Only

Use Case Title:

Paragraph 1: SAT Setup

- 1.1 Build ESM
- 1.2 Build FoSAM
- 1.3 Formalize SRs

Paragraph 2: Change Management

- 2.1 Update SAT
- 2.2 Parallel Design and Safety Analysis

Paragraph 3: System Architecture Validation/Verification Against SRs

- 3.1 Investigate Redundancy
- 3.2 Identify Safety Patterns
- 3.3 Allocate SRs
- 3.4 Assess architecture against SRs

Paragraph 4: Safety Assessment

- 4.1 Interactive Single-Failure Bottom-Up Analysis
- 4.2 Interactive Bottom-Up Analysis w/ Multiple Failures
- 4.3 Automated Bottom-Up Analysis
- 4.4 Interactive Fault Tree Generation
- 4.5 Automatic Fault Tree Generation
- 4.6 Automatic Multiple Fault Tree Generation
- 4.7 Other kind of analysis (Dynamic FT, Ordering Analysis,...)

Paragraph 5: SM Analysis

- 5.1 Simulate SM/FoSaM/ESM - Check Property

Preliminary comments:

Depending on the industrial partner methodology application strategies:

- *the Use Case 3.1 is included in other analyses: either Use Cases 4.4, 4.5, 4.6, or Use Case 4.7, or Use Cases 3.2, 3.3, 3.4, or Use Case 5.1.*
- *the Use Cases 4.4, 4.5, 4.6 are considered either included in the Use Case 4.7 or the opposite: the Use Case 4.7 is included in the Use Cases 4.4, 4.5, 4.6. In any cases, the purpose is the same: the interest to consider the timing aspects in the model and in the produced results to detect the failure scenarios leading to a Top Level Event. It has to be understood that these Use Cases are those that are the most largely and differently tested within ESACS.*

Compared with the cycle 1, the coverage of the tests is also extended: the Use Cases 4.1, 4.2 addressing the safety Bottom-Up methodologies were tested during cycle 2 on the Prover Platform. The other platforms contain the same functions to identify these Bottom-Up results helping to complete a classical FMEA. Nevertheless, the Use Case 4.3 addressing the automated production of a complete FMEA in a classical format is not still implemented.

Considering the depth of the tests carried out, the Use Cases 2.1 and 2.2, that are mentioned as partially investigated ("P"), are only explored, but the management functions required by them are not implemented in the different platforms.

These last improvements will be considered in the ISAAC project. As a preliminary assessment, it can be said that the different topics proposed by ESACS are tested after the cycle 2 methodology application. This coverage allows to complete, highlight or correct the different lessons learnt produced after the cycle 1 methodology application, as showed in the next paragraph.

6. LESSONS LEARNT FROM THE TESTS

6.1. Highlighted points

After cycle 1, this section discussed some conclusions extracted from the different test reports. They are grouped by problem which could help to prepare methodology and tool-set convergence for cycle 2.

After cycle 2, the additional test results were brought in the following paragraphs and for a better self-understanding the preceding information from cycle 1 was also reminded.

6.1.1. Abstraction on the system model

Cycle 1:

The problem before the tests:

This aspect was addressed very early in the ESACS project. For some partners, it was mandatory to start from the original design system model (SM) and then to expand it directly with failures to reach an extended system model (ESM). By this way, the ESM could become the common model shared by designers and safety analysts. For other partners, abstractions of the SM are feasible and are justified by the complexity to manage all the initial inputs plus the ones coming from the failures. For them, a common model between designers and safety analysts is accessible by applying abstractions on the initial System Model. The introduction of failures by the Safety Analysts can be done on this abstracted SM, which becomes by this way a FoSaM (Formal Safety Model). The use of simulations/model checkers help to have a common agreement on the resulting model shared by designers and safety analysts.

As a preliminary conclusion, it was admitted that the FoSaM approach found a big interest in the case where no SM in a formal manner was available, which was generally the case in the preliminary phase of the system design cycle. Moreover, it was expected that the FoSaM approach could be also explored because the problem to link a preliminary formal model with the more detailed ones generated by the designers later in the design life cycle is a valid industrial one and at least, guidelines to produce these detailed SM could be brought by the FoSaM approach. Nevertheless, this long term perspective is out of the scope of ESACS.

At last, it was noticed that in any cases the model is always different of the real system and that it is a representation of the model that the analyst has in mind. This gap is already an abstraction which also has to be addressed. How can be the resulting model less user-dependent?

Starting from the test results, it could be interesting to revisit the ability to avoid or not the abstraction step on an original SM and to progress on the associated topics.

The results of the cycle1 tests:

The first following information reflects the Airbus Deutschland testing:

Named Distribution Only

“The first step in applying the Methodology to the Case Study, i.e. the Slat/Flap-System, is the application of the model checking functionality to the System Model. The reasons for this are:

1. Before a system model is studied under the influence of failures, its normal behaviour needs to be fully understood. In particular it has to be made sure that safety properties are not violated under normal conditions. Violation would indicate the existence of design errors.
2. The technical solution to envisaged ESACS tasks such as Automatic Fault Tree Generation is based on proving (falsifying) properties for an extended system model (ESM) by means of Model Checking. As far as known today, the complexity (in terms of space and time) of Fault Tree Generation is higher than proving properties for the “fault free” system model.
3. Model Checking is applicable to finite models only. In practice, this means that system models (such as the case study) that involve the data type “real numbers” need to be approximated in the following sense:
 - a. Before model checking can start, for each variable an upper and a lower bound have to be fixed.
 - b. A resolution has to be defined. All variables of type “real” are represented by decimal numbers with this resolution. Currently this resolution has to be the same for all real numbers.

These parameters have to be set carefully. Otherwise, the behaviour of the system model is changed and thus the outcomes of model checking are of limited use.

4. The tool STATEMATE used for describing the case study provides two different “Time Models” (Asynchronous and Synchronous Model), i.e. two different semantics. It was considered necessary to study the influence of the semantics on the interpretation of proved (falsified) properties and on the complexity of computation.

As a result, a sub-system of the Slat-/Flap system was tested. This restriction was enforced by large number of state-bits (approximately 15000), which the model checker (the underlying hardware) cannot manage.”

So, according to a reducing of the scope of the system and according to a well managed improvement of the initial SM (by adapting the real number resolution), the model checker can produce results without additional abstractions. Note that this encouraging information does not apply on an ESM, i.e. with the additional complexity of injected failures.

From Alenia/SIA testing, the first tests showed that the SPS Model 2 (model with Real numbers) is, at its initial stage, “not verifiable nor with the platform, neither with the Model Checker for the following reasons:

- model too big
- model with too many points of complexity (real numbers, presence of long timers, write/write races with different values).”

So simplifications needed also to be applied to run the tests.

At an intermediary conclusion, it was also noticed that at least some guidelines should be given to the designers in view to limit the complexity of their models (real number resolution refinement, constant identification, use of timers is not advisable for our “verification” purpose,...).

The other test reports do not mention this difficulty because they used Boolean or Integer values (in the SAAB example also Real type variables was used but only for “comparison”),

Named Distribution Only

which are generally used in preliminary or simplified models of a complex system or in models which are behavioural models using a set of constraints relating the local variables of a component, as those used by Airbus France to build from scratch a FoSaM (these last models are also known by Airbus Deutschland – see Jakob Mauss, Volker May, and Mugur Tatar: Towards Model-based Engineering: Failure Analysis with MDS, ECAI-2000 Workshop on Knowledge-Based Systems for Model-Based Engineering, Berlin, 22.08.2000).

Nevertheless, even in these models using Boolean or Integer values, SAAB recommended a preliminary checking approach similar to the one presented by Airbus Deutschland:

“Before the extension (introduction of failure modes) of the system model is done there is a possibility to check the nominal behaviour of the system using the “ordinary built-in” PPI (model checker) in SCADE. This helps to find any modelling errors or if there are any logical shortcomings in the system. This type of analysis should be mandatory before starting the FM extension and analyses.”

In the Airbus France testing, the check of system behaviour was regularly done with simulations or model checking. For instance, during a particular test (T1b), it was understood (through simulations) “that a real reverse flow was not modelled like it was in the physical system.”

These recommendations can progress on the topic related to the model user-dependency. The model checking techniques appear to be a good candidate to appropriate a SM built by another user. The industrial difficulty to apply this recommendation is that the level of skills does not seem simple and, at least, a special training in Model Checking is a pre-requisite.

Airbus UK testing has the aim of identifying the lowest level of abstraction where the system model can be used in parallel with the safety model. As such Airbus UK followed the following process.

1. A simple FoSaM was constructed consisting of only Type Boolean variables. This model was used to understand the process of using the various tools with the extended capability. The model was of a basic failure logic circuit provided by Prover that had various order failure modes. The circuit contained 10 logic operators. 6 inputs and 1 output. The safety requirement was simply total loss of output. The model was produced in Simulink, then translated into Scade. The Scade representation had to have one assertion added and association of the output as a Point of Control and Observation (PCO). The model was then extended using the FTA_Manager with Fail off failure modes for each of the inputs. The model was then tested for 0 to 3 cut sets. The results were as expected.
2. A second more complex FoSaM was produced for a hydraulic power generation model. Airbus France and Onera specified the model. The model was constructed, again using Simulink and then translating into Scade etc. as in 1. The results agreed to those produced by Onera, IRST and Prover for the same model.
3. A third model was, and is still being used for further testing of the platform. The model is a continuous time Nose Wheel Steering Model that contains all the features that a generic nose wheel steering systems may utilise. The model was extended with basic failure functionality related to the functioning of main components and to advanced failure functionality that looks at signal accuracy in comparison to a reference signals. The model also contains time and sequence dependent behaviour that can be tested for conformity to time and sequence based safety requirements. The use of this model has been staged:
 - A. Initially only basic failure functionality related to the functioning of main components was included. The relationship between components was neglected and all blocked were assumed required so only 'AND' gates were

used. This constructed a model with 32 inputs. It was demonstrated that only 1st order cut sets were present, specifically 32 of them. (computer memory issues were identified at this stage.)

- B. The ESM was modified to contain the correct logic relative to the actual functional model behaviour taking into account redundancy, timing and sequencing issues. This is the current progress of testing so far.

So, from the Airbus UK testing, it was studied to have a functional and symbolic model coexisting that interact with each other.

Lessons learnt from the testing:

The models based on Boolean or Integer values can be used by ESACS a priori without additional abstraction. For the model using Real numbers, the question is no more in words of methodology, but in words of technological limitations. The testing seems to indicate that we arrived at the limit of the technology ability if we want to consider a whole system model.

If this point is confirmed, that means that an abstraction step is required and that the simulations and model checking could be used to consolidate the opinions of the designer and of the safety analyst that they shared a common model. Related to the abstraction step, different strategies can be applied and methodology details are required. Amongst these strategies, which may be combined, we can list the following orientations:

- reducing the scope of the system, which leads to study in detail the problem of interfaces between the delimited scope and the other parts of the system.
- proposing the appropriate adoption of Integer and Boolean values in place of Real values.
- transforming a Real value system model into a behavioural model.

Some of these strategies are also proposed in the “Functional and Safety Property Checking” chapter.

Cycle 2:

The problem before the cycle 2 tests:

The problem dealing with the abstraction of system model used for the analysis is the same described for cycle 1 tests.

The results of the Alenia/SIA cycle 2 tests:

The results, coming from testing activity performed by ALA/SIA during cycle 2 on the OFFIS implementation line, show that the fault tree computation algorithm has been sensitively optimised (about 10 times) starting from ver. 0.193 of the STSA platform with respect to the previous platform releases.

In fact it is now possible to obtain some significant results also from an evolution of the formal model used as case study during cycle #1 of application, that is a SPS model with integer variables assuming values in the range [0, 30] but increased in complexity due to the integration of the failed behavior of the system freewheels that represent the main interfaces among the SPS items (i.e. “shaft – gearbox” and “turbine-gearbox”)

Nevertheless the main limitations relevant to the formal system model to be used for the analyses, (i.e. model too big or with many points of complexity like real variable, long timers, complex data structures as inputs (as vectors, matrix, ..)...) are still applicable also after the cycle #2 of application.

Lessons learnt from the Alenia/SIA cycle 2 testing:

The lesson learnt from the testing is the same coming from cycle 1 tests.

In addition to that, 2 points need to be highlighted. They are related to technology improvements:

- *need to optimise the computation algorithm*
- *need to allow the analysis on models including more complex data structures. To this purpose it has to be reminded that the design engineers is using all the facilities offered by modelling tools (e.g. for Statemate tool: vector, matrix, generic chart, etc.)*

The above activities will be continued, in the future, within the European sponsored project ISAAC (Improvement of Safety Activities on Aeronautical Complex systems – FP6, 1st Call). In particular in the Work Description document at paragraph 7.6.1 “Consolidation of ESACS work – Further development of platform/tools – already started in ESACS”, an “optimisation of computation algorithms to reduce the current limitations and restrictions in the use of the ESACS platform (for instance the handling of models bigger in size, timers, components items, user interface)” is foreseen.

All the above for going towards a more mature “tool-set” to be applied in the industrial process.

The results of the Airbus Deutschland cycle 2 tests and lessons learned:

Two case studies were performed:

1. *Controller of Flap System (one model: Statemate– synchronous time semantics)*
2. *Sidestick Priority Logic (two models Scade and Statemate – asynchronous time semantics)*

The model analyzed in Case study 1 is one part of the case study presented in work package 3. During cycle 1 it turned out that the full model was too big to be handled by the model checker. Therefore, the most interesting part, the Flap Controller, involving intricate decision logics, was analyzed in detail.

The Case Study 2 was added during cycle 2. The reason for this was, first, to have an example where the Statemate/Model Certifier/STSA and the Scade/SPPI approach could be compared and, second, to have models the fault trees of which could be verified manually within a reasonable time.

Level of abstraction: All models were built in the design office. The purpose of the Slat/Flap Model was to validate the textual system specification (completeness, plausibility etc). So, the focus of the models is on describing the functionality of the system (decision logic, performance etc.). Safety aspects are included like other aspects but do not play the key role. The Sidestick Priority Logic is a specification of a controller that is translated into code.

The advantage of using design models is that the design engineer can immediately assess the importance (relevance) of the results of the ESACS analysis. No extra effort is required to translate (interpret) the findings (as required for traditional fault-trees). The possibility to immediately assess the relevance holds in particular for the verification of (safety) requirements. A draw back of using design models is that the functionality is usually quite complex, and therefore the models are quite big. So, a priori it cannot be guaranteed that the analyses (model checking runs) will terminated when applied to the full model. Work is going on to improve "proof engine" capacity to be able to handle large/complex design models.

The results of the Airbus UK cycle 2 tests:

Airbus UK testing has the aim of identifying the lowest level of abstraction where the behavioural model can be used in parallel or with the safety model or as a replacement for the safety model. As such Airbus UK followed the same process than for cycle 1 tests (see cycle 1 paragraph above):

A simple FoSaM was constructed during cycle #1 consisting of only Type Boolean variables. In cycle #2 the model was then tested using the bottom up capability forcing various sets of injected failures and verifying that the top level event was achieved as predicted by the cut-set analysis. So again for the FMEA it behaved as expected.

A second more complex FoSaM was produced during cycle #1 for a hydraulic power generation model. For cycle #2 this model was checked using the normal FTA analysis which again produced the expected results though the speed of analysis was considerably faster and the size of swap space needed to complete the analysis was greatly reduced. When performing the FMEA analysis some incompatibility issues were identified. This is believed to be a platform or software issue, as on the development platforms this issue does not exist. Issues associated with new development of the simulink 2 scade gateway caused some usage problems with the FTA GUI.

A third ESM model was, and is still being used for further testing of the platform. The model is a continuous time Nose Wheel Steering Model that contains all the features that a generic nose wheel steering systems may utilise. The model was extended with basic failure functionality related to the functioning of the main components and to advanced failure functionality that looks at signal accuracy in comparison to reference signals. The model also contains time and sequence dependent behaviour that can be tested for conformity to time and sequence based safety requirements. The use of this model has been separated into stages. In the first cycle the model was used in two ways:

- A. *The initial use was to construct a Boolean skeleton that existed in parallel with the behavioural model. The Boolean skeleton represented the function block interaction dependencies and affected the outputs of both the Boolean and behavioural model. The outputs of the Boolean model represented the safety requirements. This was primarily used in the first cycle. This allowed for an investigation of the system's Boolean skeleton where the behavioural portion could only be observed through simulation.*
 - I. *Initially only basic failure functionality related to the functioning of main components was included. The relationship between components was neglected and all blocks were assumed required so only 'AND' gates were used. This constructed a model with 32 inputs. The analysis results demonstrated that only 1st order cut sets were present, specifically 32 of them. (Computer memory issues were identified at this stage.) For cycle #2 the memory issues were resolved.*
 - II. *The ESM was modified to contain the correct logic relative to the actual functional model behaviour taking into account redundancy, timing and sequencing issues. This was the progress of testing achieved through cycle #1. For cycle 2 this version of the model was not used as the features contained in this model were also available in the Hydraulic power generation model and the nose wheel steering model without the Boolean skeleton.*

Named Distribution Only

- B. *During the second cycle the basic nose wheel steering model without the Boolean skeleton was used. Based on the analysis strategy used with the Prover it is possible to perform different strategies for analyses e.g. static or dynamic analyses. The two aims were to reproduce the same results for the static analysis performed in cycle #1 but without modelling the functional dependencies using a Boolean Skeleton. A further aim was to investigate dynamic safety requirements against the dynamic system subjected to a rate input.*
- I. *Default Analysis with behavioural model: In order to perform a static analysis it was necessary to at least complete a default analysis to show that the nominal model without failures injected produced a valid result. It was found that by placing a range of real values for a few of the inputs, with others set to fixed values, this challenged the capabilities of the proof engine attached to Scade and required somewhere in the order of 25 hours to complete an analysis. The analysis space was reduced whilst the proof engines capability was increased in subsequent steps until it was possible to complete a default analysis. The analysis was investigating the static system. A mixed Real/ Boolean/ Integer version of the model was tested to see if this would help reduce the analysis space but it was found to increase the difficulty of the problem increasing the time to achieve a result to more than 25 hours. The analysis space was reduced drastically so that a default analysis would take up to 30 minutes. From the analysis it was quite obvious that the stand alone prover has a much greater capability than what is available through the prover that is attached to Scade. Esterel and Prover are looking at increasing the capabilities of the Scade Prover..*
 - II. *Zero and 1st Order cut-set analysis: The behavioural model was analysed for 0 order cut sets to confirm the operation of the proof engine using the FTA capability. The result was as expected. The order of the investigation was increased to a 1st order cutset analysis. The tool coupled to the Scade environment produces an error. The same analysis using the stand alone prover (carried out by Prover) produced the expected results.*

So, the Airbus UK testing, investigated the formal analysis of

- *a functional and symbolic model, coexisting, that interact with each other,*
- *two symbolic models with no functional behaviour*
- *a behavioural model containing type “Real, Boolean, and Integer” variables with non-linear characteristics such as Deadzone and Backlash, and dynamic behavioural characteristics.*
 - *It was also possible to identify the excursion of the system behaviour from the dynamic safety envelope as opposed to just a static safety envelope.*
- *The results from the behavioural model testing were incomplete due to the level of maturity of the gateway, the level of constraint on the prover implemented within Scade, incompatibilities between the model file formats generated by the gateway and the FTA GUI.*

Lessons learnt from the Airbus UK cycle 2 testing:

From AUK's testing of the behavioural model, with the help of Prover, it was shown that it is possible to produce some basic results for a model that has all three numeric types. There is a lot of work that still needs to be completed before the tool capability can be used within a design environment by systems specialists as opposed to having the analysis prepared by formal methods specialists such as the team at Prover.

Other tests:

*- In cycle 2, a few real variables were used in the **Saab testing on the Prover platform**. There were a few divisions using constants that were defined as reals, but after this division they were transformed to integers. These reals caused no problems in the analysis. However, testing has proved that real numbers can be used in some cases, at least in not too complicated models.*

*- The **Airbus France/Onera** approach suggested to start with behavioural model of a system. Basically these models always include abstractions which are validated through simulations and model checking .So the Airbus France/Onera cycle 2 tests confirmed their approach without discussing further more this topic.*

6.1.2. Failure injection

Cycle 1:

The problem before the tests:

This aspect is first a user access concern. Then, it becomes a methodological topic. The failure injection into a model to obtain an ESM or a FoSaM is something which is proposed in different manners by the technology provider partners:

- in a very graphical way, by modelling the failures, with “decision” nodes using Boolean logic connected to the signals of the components within the SM model. See the Graphical User Interface (GUI) using libraries of the PROVER platform.
- in a library of generic failure modes (GFML). See the IRST platform.
- in a library of components (COMPL), including their nominal behaviour, failure behaviour and their input/output transformations. See the ONERA platform.
- by directly accessing to the formal language to detail a failure mode to be introduced in the model and ability to include this failure mode into a library. Basically all the platforms could offer this alternative.

The methodological topic is more related to the last aspect. Do we have to limit the failure injection in a user-friendly access (via libraries) to simple failures and is it unrealistic to envisage to extend this way to the complex failures (cascading failures and in particular reverse flow failure propagation; transient failures and in particular unstable failures ; ...)?

The results of the cycle1 tests:

The results cannot allow to conclude but we can summarise the main comments raised in each report.

Firstly related to the user access concern:

All the feasible tests shows that the use of the libraries are very convenient from a user point of view, but remarks are made about the necessity to expand these libraries.

More in details, the GUI from the Prover Platform appears very easy to handle. A slight nuance was brought about the graphical display of the failure injection results (readability of output aspect): ‘little difficult since there are many lines going in and out from each node’.

Related to the injection of new failure modes, it was noted that:

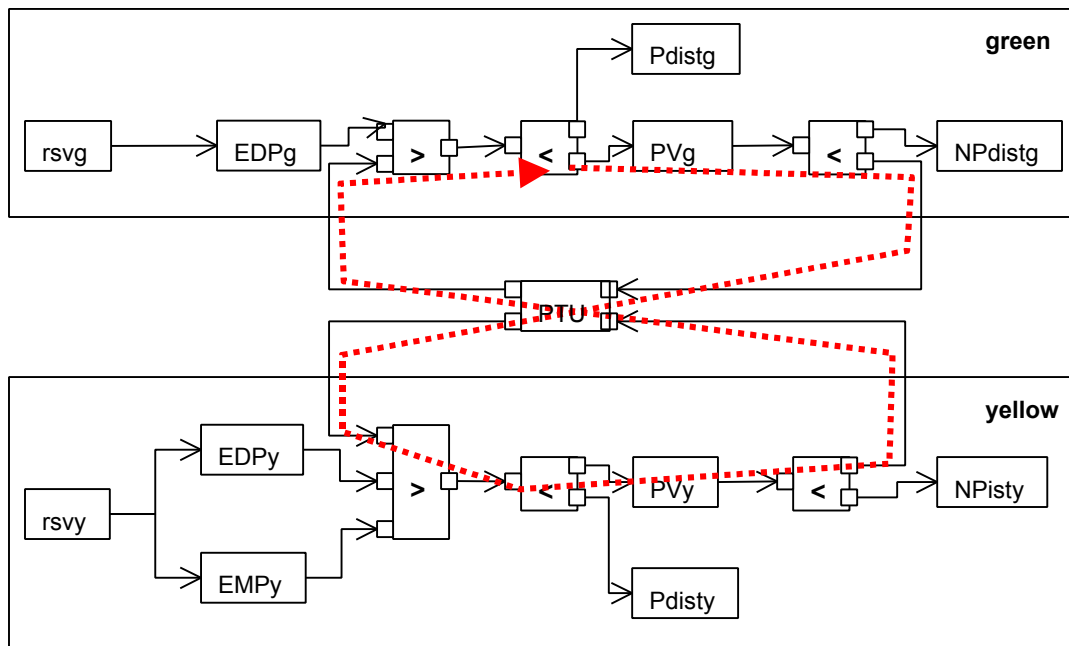
- to enlarge the GFML (Generic Failure Mode Library) with additional failure modes requires a skilled safety engineer who knows the platform formal language.
- This comment is also valid for the extension of a COMPL (Component Library).

Secondly, related to the methodological concern, we could be supported by the different models proposed by Airbus France/Onera on an aircraft hydraulic system. Airbus France and Onera developed an example based on a real system, but where they introduced voluntary a component functioning allowing a cascading failure: the propagation of a failure from one hydraulic line to another one became possible. This example is not a real one but could be.

Named Distribution Only

“Tools associated with data flow languages such as Lustre/Scade, SMV or AltaRica DF subset do not allow circular definitions where the value of a variable X at time t is defined by a function that depends of X at time t (i.e. $X_t = f(\dots, X_t, \dots)$). So loops in the hydraulic system may lead to models that could not be treated by safety assessment tools due to circular definitions.

The following picture shows a loop in the hydraulic system that leads to a circular definition. The input of the green priority valve depends of the output of EDPg and of the yellow to green PTU output but this output may depend on the priority valve input. This situation should not occur as the nominal behaviour of the PTU does not allow simultaneous flows from green to yellow and from yellow to green. But if the activation of the PTU is incorrect, then the loop is physically possible.



A physical Loop

As AltaRica OCAS tool does not exclude circular definitions, we built a first model (HYDBOOLSTATBOUCLE.alt) that follows strictly the hydraulic system structure.

All other models are based on a modification of the structure that avoids physical loops and circular definitions.”

This model aspect was presented in the context of building a model, here a FoSaM. In this particular context, the Airbus France/Onera approach mixing this building and the use of components (including their failures) could find user friendly proposition: the enhancement of the COMPL to illustrate a reverse flow does not raise major model building problem. It does not mean that the tests to automate Fault Tree generation were satisfactory whatever the adopted model. This aspect was discussed later on (see the “Fault Tree and Sequence Generation” chapter).

Coming back to the injection failure topic, and in the case where a SM is proposed, whatever the injection of component failures is in some SM, it could be impossible to detect this cascading failure if your initial system model is not build to represent a reverse flow. The reasons for that could come from other modelling perspectives, for instance, because in the normal functioning, the system designer does not care and makes this choice of modelling.

Named Distribution Only

From this Airbus France Case Study experience, it seems difficult to separate the task of model building and the task of failure injection.

To end this open point, SAAB from his own experimentation can offer a preliminary conclusion: “A difficulty might be to understand that most of the modeling – SM, FM and SR nodes as well as coupling of SM and SR – must be done before start using the GUI for doing the extension and defining the analysis.”

Lessons learnt from the testing:

The methodology application was not developed enough to conclude definitively. The 2 tasks of building a SM and of injection of failures need to be more explored. An obvious prerequisite is to develop new failure modes in the GFML/COMPL/GUI to better assess this point.

A preliminary conclusion after the 1st cycle of application is that two scenarios are applicable:

- The failure has implications/effects only on the output of the block affected by the failure. E.g. if the failure occurs, the output of the block is 0. This case can be dealt with automatic extension facilities by dragging the failure mode (e.g. “stuck at 0”) from a GFML
- The failure has more complex implications: both local on the block and also on other blocks, e.g. reverting a flow of a fluid, etc. This case has to be dealt with “manual” extension of the SM by directly accessing to the model for modifications.

Cycle 2:

The problem before the cycle 2 tests:

The problem dealing with the possible ways for the Failure Modes definition is the same described for cycle 1 tests.

The results of the Alenia/SIA cycle 2 tests:

From ALA/SIA testing activity on the Statemate implementation line, the following final consideration relevant to the methodological problem of FMs definition and injection in the formal model has to be done.

As a matter of fact, the ESACS approach offers two possible solutions for the extension of formal system model to include the failed behavior of its components:

- *the activation of “user-defined” FM, defined by the safety engineer as model inputs to enable the components failed behavior (modeled “ad hoc” by the safety engineer).*
- *the definition of different types of FM on a whatever local/output model variable (selected by the user) inside the STSA platform without changing the formal model*

In the cycle #2 of application a lot of work has been done relevant to the second type of FMs by enlarging the FMs library (including, at the end of cycle #1, only the stuck-at FM) with other types of FM, namely: delay, random and ramp-down FM.

Moreover the possibility of sporadic activation mode for both user-defined and library’s FMs has been implemented

For the IRST implementation line, the main improvements relevant to the FMs definition from the GFML, concern the related GUI that has been sensitively improved through the introduction of a data dictionary, giving a direct link with the SM/ESM modules and variables.

Lessons learnt from the Alenia/SIA cycle 2 testing:

The implementation of the FM library can be considered in general satisfactory and it can in principle cover the main categories of FMs occurring in a real case.

Nevertheless failure modes having more complex implications (e.g. failures having a temporal dependency among them) still deal with a “manual” extension of the SM. This problem will be investigated more in detail in the ISAAC project.

Results and Lessons learnt from the Airbus Deutschland cycle 2 testing:

Mainly Failure Modes of type “stuck-at” were used (this is the simplest type). These relate to inputs and outputs of system components. For the StateMate safety analysis it is a fact that if a system is failure tolerant against a random Failure Mode, it is also failure tolerant against any other simple Failure Mode, like delay or stuck-at errors. So, in a sense, the random failure mode is the most general one. However, a system that tolerates the most general failure might be over-designed. Statistical information about the frequency of failure modes needs to be taken into account for selection of relevant failure modes.

Up to now failure modes that change the structure of the system (generation of new interfaces such as disconnect in a mechanical assembly, or short-to-ground) were not considered.

The results of the Airbus France/Onera cycle 2 tests:

The Airbus France/Onera cycle 2 tests were focused on a civil aircraft Electrical Generation and Distribution System since the cycle 1 tests concerned a civil aircraft Hydraulic System.

In the frame of the Airbus France/Onera cycle 2 tests, the assessment of the platform for another system technology will consolidate the cycle 1 conclusion:

- Related to the Failure Modes, it appeared that the first Failure Modes proposed by the platform library were not adapted to the real propagation of these failures in the system. So it was necessary to redefine them and to introduce their right behaviour in the library. It can be expected that the libraries obtain a certain level of validation to avoid this kind of difficulties.
- Related to the adaptation of the model, it was immediately identified that the ‘loops’, as exposed during cycle 1 on the hydraulic system, were more complex for an electrical technology: during cycle 1, different strategies were explored: the first one considered that the Altarica Ocas tool did not exclude circular definitions (that was not the case with SCADE, SMV, etc.). So for the hydraulic system an analysis to model the loop was done expecting that a model as close as possible of the real behaviour will be better both for the safety results and for the understanding of the model by the designers (models without physical loops were also built for enhancing the conclusions). In the case of the electrical system, the analysis concluded on the major difficulties to produce such circular definition models: not only the number of logical paths to model a loop was more important than in the case of the hydraulic system but also the necessary choice of some paths in the case of the electrical system could lead to ignore some failure propagations via the paths which were excluded. Consequently, the necessity to break the loops appeared. As the available algorithms allowing the automation of breaking the loops gave insufficiently appropriate results (see cycle 1), it was done by hand by placing at the most appropriate place the necessary ‘delays’ in the system model. It was clear at this level that the Fault Tree automation technique was not accurate for these dynamic models (see cycle 1). So the Airbus France/Onera cycle 2 testing focused on the Sequence Generation automation (see paragraph 6.1.4) and on the Model Checking techniques (see paragraph 6.1.3).

Lessons learnt from the Airbus France/Onera cycle 2 testing:

Coming back to the building of the model with its failure modes, even with libraries, it cannot be ignored that the users will have to introduce in the model, and possibly during a certain time in the libraries themselves, some adaptations that will require the knowledge of the language (here Altarica) and of the techniques used by the platform.

Other tests:

- *From the **SAAB/Prover** cycle 2 tests, it should be confirmed that the GUI from the Prover Platform appears very easy to handle, as mentioned after cycle 1. The slight difficulty related to the readability of output aspects is more related to the graphical display of the modelling of requirements rather than the display of the failure injection results. Anyway, the limit should be addressed to SCADE and not to the GUI.*
- *In the **Airbus UK** cycle 2 tests, in AUK's effort the following point was reinforced: to enlarge the GFML (Generic Failure Mode Library) with additional failure modes requires a skilled safety engineer who knows the platform formal language.*

6.1.3. Functional and Safety Property Checking

Cycle 1:

The problem before the tests:

With the term “model checkers” we intend the set of tools used to perform the “model analysis”, that is the verification of different types of system properties (namely functional and safety requirements) to check both the nominal and the degraded behaviour of a given formal system model (FoSaM or SM/ESM). To do this, the model-checking engine must “speak” the same formal language used to build the system model. As a consequence, the problems related to the “Model Analysis” aspect have repercussion on the system modelling activity.

In particular some restrictions are needed during the design activity to generate a formal system model that can be verified by the relevant model checker engine. These restrictions include special rules that have to be followed to reduce the system complexity (for instance by limiting the number of components and variables and/or by using only certain types of variables assuming values in a restricted range) and, in some cases, to avoid inconsistent or inaccurate result.

The results of the cycle 1 tests allow to estimate in which terms the technical limitations of the model checkers engines affect the system analysis and to investigate the possible margins of improvement (optimisation of the theorem proving algorithm and/or use of alternative not exhaustive model checking techniques).

The result of the cycle 1 tests:

The cycle 1 tests performed by ALA/SIA relevant to the verification of system properties on different Secondary Power System models, with two implementation lines (namely Statemate and NuSMV) highlighted that the way to work on “formal models” and relevant “formal” verification techniques allow the representation of the safety properties and also the verification of that requirements.

On the other side it has to be said that the verification algorithms have to be improved in order to can deal with system model bigger in size and with too many points of complexity (real numbers, presence of long timers, write/write races with different values).

So, at this point in time, simplifications needed to run the tests, mainly to avoid too long execution times, out of memory or tool crashes problems.

The intention is to progress in the 2nd cycle with both implementation lines and also enlarging the models for the tests in order to have a larger view for their capabilities.

Lessons learnt from testing:

From testing activity, the main lesson learnt concerns the approach that has to be followed during the design modelling activity in order to be able to obtain the more useful information from model checker elaboration.

First, it was identified the need to follow specific modelling guidelines, then it was identified the need to limit the scope of the analysis. This last implying two different approaches, deriving from two different scenarios:

- in case the design SM is available, to limit the analysis to a subpart of the SM

Named Distribution Only

- in case the design model is not available, to proceed by steps increasingly in complexity in producing the models for the analysis

In this second scenario, it is necessary to start with a simplified model of the system including only the main functional blocks and a restricted number of Boolean variables. The model checker elaboration on this type of system model is very fast and allows highlighting the main characteristics of the design architecture (symmetries, redundancies, controller, vote, monitor).

After that, we focus on a limited part of the system (for example: only one among two rounded lines without any monitor or controller) that is significant to verify some particular categories of system properties.

The model of this limited part of the system will be enriched with respect the initial functional blocks model through the introduction of the HW components and different types of variables assuming different values (either a number discrete of values or all possible values in a restricted range of variability).

The degree of complexity has to be increased, if possible, to reproduce, at least in a qualitative way, the real system behaviour.

Finally we can increase the model complexity by adding other system components/variables and by extending the range of values handled; this process will stop when the system property verification by model checker engine will require too much time of elaboration and/or will incur in out of memory problems.

Cycle 2:

The problem before the tests:

The problem dealing with the functional and safety property checking is the same described for cycle 1 tests.

The results and lessons learnt of the Alenia/SIA cycle 2 tests:

See “The result and lessons learnt of the cycle 2 tests” at para 6.1.1

The results and lessons learnt of the Airbus Deutschland cycle 2 tests:

Regarding formalization of (safety) requirements_experience has shown that in the design office requirements to a system model are not always described as formal as required for application of the ESACS methodology. So, formalization requires training. Some engineers found it difficult to describe a requirement in a declarative way (the “what”). In particular, when the requirements and the model (the “how”) are described in the same language (as in Scade and Prover Engine) there is a risk of having a common mode on a “syntactic level”: that is to say, the same operator (node) is used for the observer (encoding the requirement) and the implementation of this requirement. In this case verification is meaningless. Also there is a risk of having a common mode on a “semantic level”, i.e. a textual requirement is misunderstood by the person who formalizes it in a declarative way (example: a temporal logic formula or pattern in Statemate) and by the person who implements it. As a consequence it has to be validated first, that the formal (safety) requirement “matches” the textual requirement (this is a matter of engineering judgement).

Complexity issues: The above models contain timers and counters. An example of usage would be: upon receipt of a certain external message wait for N steps and then do “XYZ”. The original model contains values $N > 300$. Values like this are still a problem to the model checker as they usually imply a large model diameter. Timers had to be rescaled. Rescaling changes the given model M into a new model M'. Proven properties hold for M'. Justification that these properties hold also for M was done informally. This approach should be improved, as timers are a potential source of errors. Usage of real numbers also contributes to complexity (as mentioned in other sections of this document).

The results and lessons learnt of the Airbus France/Onera cycle 2 tests:

In the frame of the Airbus France / Onera cycle 2 tests, the model checker SMV gave accurate results with the behavioural system model (ESM and FoSaM) of a civil aircraft electrical system (Note: the use of SMV was possible through the translation of the Altarica model into a Cadence SMV model). Moreover, it appeared also that the first industrial expectation that simulations were sufficient to assess a model in the case of a ‘simplified’ ESM or FoSaM was incorrect. As a matter of fact even in the ‘simplest’ models the number of potential system model states are so big that simulations cannot be played by hand and cannot guarantee the completeness of the safety conclusion on a system architecture. Some user premature conclusions using just simulation were denied by the model checker results. Counter example was produced showing the power of the model checking technique.

The results and lessons learnt of the SAAB/Prover:

In the SAAB/Prover cycle 2 tests it was found that to some extent the memory problem revealed by the cycle 1 results can be handled by only calculating one MCS at a time when selecting “show counter model”. If all MCS up to a certain size is required in one execution it is recommended to select “not show counter model” on the Prover platform.

6.1.4. Fault Tree and Sequence Generation

Cycle 1:

The problem before the tests:

The problem to be solved is to find out causes (fault tree or sequences) leading to a top level event starting from a formal model (ESM or FoSaM).

This aspect is first a methodological topic (algorithms to implement considering the limits of the models), then it becomes a user/platform interface concern (way to display a fault tree and/or fault tree results).

The results of the cycle1 tests:

The 1st result is that in principle it is possible to automatically derive the FT from the ESM.

Some issues arose:

ONERA developed five models of an hydraulic model to test whether fault tree generators would be able to handle a non data flow model, a data flow model without reverse flows and with static activation, a data flow model without reverse flows and with dynamic activation, a data flow model with reverse flow and static activation and a data flow model with reverse flows and dynamic activation.

ONERA and AIRBUS France used these models to test two fault tree generators : one included in OCAS/Altarica and the other included in AltaRica Toolbox. The main positive result is that both fault tree generators give good results (i.e. efficient generation and correct minimal cuts) on data flow model with static activation, the main limitations we have encountered when testing the tools are that OCAS/Altarica Fault tree generator is able to generate fault trees for dynamic models but they are not satisfactory as they generally do not take into account the various states that can be reached by the system, AltaRica Toolbox Fault tree generator was not able to deal with dynamic models (even with limited dynamic behaviour as using update events). We also tested the sequence generator of AltaRica Toolbox, it gave good results on data flows models with reverse flows but without dynamic activation.

Prover and IRST tested their tools on the data flow model with static activation and found similar results. These tests were made possible thanks to converters from AltaRica language to Lustre/SCADE and to NuSMV language.

In the Prover approach no FT was generated explicitly. Instead Prover utilise the analysis result – being a counter model – to identify the MCS; given that we have a static model this will lead to the same qualitative result as running a FTA tool. In case of possible dynamic system behaviour, when identifying MCS, it is necessary to interpret the sequence of system evolvment in combination with the identified MCS to get a complete understanding of way the TLE is reached.

Relevant to the Statemate implementation line it was evidenced that some methodological issues has to be approached for obtaining useful results. The main issues related to the generation of the FT are the following:

Named Distribution Only

- To include the possibility to skip the start-up phase, i.e. making the analysis starts only after that the system has reached a specific configuration
- To insert patterns (e.g. “output = 0 for more that a certain time”) in the property to be verified in doing the FTA
- To give the user the possibility to select if the failures can occur simultaneously (e.g. two failures in the same time instant) or not
- To structure the obtained flat FT in a new FT that is more related to the structure/architecture of the system

Moreover, in order to make possible the FTA for more complex models the following are needed:

- the computation algorithm has to be improved
- the possibility to add some “freezing” facilities for the input variables has to be included. This in order to reduce the complexity in the calculation

Lessons learnt from the testing:

From ONERA point of view, tool improvement is not the most urgent work to be performed. We first need to know when it is appropriate to use a tool or not according to the type of model and safety requirement that we have to assess. For instance, if it is necessary not only to consider what failures causes a top level event but also the order in with that failures must occur, then the use of a classical fault tree presentation could be misleading. Tools that take into account the order of failures should be used in such a case. On the contrary, the use of such tools to generate the minimal cuts of a non-temporal requirement on a static model seems to be overkill and might lead to avoidable bad performances.

We propose to elaborate a methodology that takes into account the strengths and weaknesses of the various tools. This methodology should provide guidelines to the safety engineers when they have to perform safety assessments. The methodology could recommend the use of one tool based on the form of the model and the safety requirement. The methodology could give hints on how to decompose the models in parts that could be checked efficiently with different tools. Finally the methodology could provide approaches to synthesise the results obtained with various tools.

For ALA point of view what is important to highlight at the end of cycle 1 is that the original ESACS requirement “to generate FT in some automatic manner from the model” was verified.

It is also possible to interface the ESACS FT output with some commercial FT tools.

Some requirements for implementation were identified for cycle 2.

As methodological aspect it was identified that sequences and counterexamples are needed for having more useful details on the dynamic path leading to the TLE.

Cycle 2:

The problem before the Alenia/SIA cycle 2 tests:

The problem described before cycle 1 tests of finding out causes (fault tree or sequences) leading to a top level event starting from a formal model (ESM or FoSaM) was solved at the end of cycle 1.

Then the new aspects to be approached during cycle 2 tests are those methodological issues highlighted at the end of cycle 1 tests performed by Alenia/SIA i.e.:

- To include the possibility to skip the start-up phase, i.e. making the analysis starts only after that the system has reached a specific configuration
- To insert patterns (e.g. "output = 0 for more that a certain time") in the property to be verified in doing the FTA
- To give the user the possibility to select if the failures can occur simultaneously (e.g. two failures in the same time instant) or not
- To structure the obtained flat FT in a new FT that is more related to the structure/architecture of the system

Moreover, in order to make possible the FTA for more complex models the following are needed:

- the computation algorithm has to be improved
- the possibility to add some "freezing" facilities for the input variables has to be included. This in order to reduce the complexity in the calculation

Furthermore from lesson learnt from cycle 1 tests, some requirements for implementation were identified for cycle 2.

In particular as methodological aspect it was identified that sequences and counterexamples are needed for having more useful details on the dynamic path leading to the TLE.

The results of the Alenia/SIA cycle 2 tests:

From Alenia/SIA cycle 2 tests, the following new facilities have been implemented on STSA to improve the failure modes handling and to give to the user the possibility to perform more specific and refined safety analyses:

- the definition of the bounding conditions for a FTA relevant to a TLE defined inside the STSA platform like:
 - the indication of minimum duration of the TLE. This facility allows avoiding the temporary occurrence of the TLE (if out of the analysis scope)
 - the definition of the starting point for the analysis. This facility gives to the user the possibility to skip the initialization phase.
- the possibility to define FTA by importing proof from ModelCertifier. This facility makes available the complete set of pattern proofs predefined in the Statemate ModelCertifier commercial tool.
- the possibility to freeze the inputs model to predefined values. This facility allows the user focusing on a particular system scenario

Named Distribution Only

- *the control on the number of failure modes that can happen in a given temporal window. This facility allows avoiding simultaneous failures*
- *the possibility to generate the simulation paths (SCPs) associate with each MCS found. This facility allows debugging the results of a FTA in terms of counter-examples found by the Model checking engine.*

At the end of the tests performed by Alenia/SIA on the IRST Platform during cycle #2, the main improvements, with respect the problems and limitation encountered during cycle #1 of application, relevant to the model analysis task are:

- *the GUI has been sensitively improved through the introduction of data dictionaries, giving a direct link with the SM/ESM modules and variables, and of SMV keypad / pattern library, supporting the user in the system property writing.*
- *From a methodological point of view, the introduction of the possibility to define some hypotheses / assumptions for each analysis task. At present, only the freezing on FMs and input variables has been properly implemented.*

Lessons learnt from the Alenia/SIA cycle 2 testing:

For Alenia/SIA point of view what is important to highlight at the end of cycle 2 is that not only the original ESACS requirement “to generate FT in some automatic manner from the model” has been satisfied but also new requirements risen up from the first phase of test activity were implemented.

The following methodological improvements are still necessary:

- *relevant the SCP generation, to eliminate some limitation and malfunctioning (risen up under given conditions like FTA with restriction on simultaneous failures, asynchronous model, ...) highlighted during the test activity.*
- *to optimize the Model Checking engine to deal with more complex system formal models;*
- *to improve the treatment of no-simultaneous failures by optimizing the relevant FT computation algorithm;*

All these improvements are within the scope of the ISAAC project.

Finally the main lessons learnt from the testing are:

- *the implementation of new facilities, required at the end of cycle 1 tests, effectively allows the user obtaining more useful results from the safety analyses because they give to the user the possibility to perform more specific and refined FTA and, at the same time, to increase the number and the typologies of analyses. This guaranties a more complete investigation of the system behaviour with respect the traditional approach.*
- *Moreover, the SCP generation allows debugging unexpected FTA results induced by the system dynamic and not foreseen in the analyses, usually performed in the safety industrial process, coming from an only static view of the system.*

Lessons learnt from the Airbus Deutschland cycle 2 testing:

The Slat/Flap-System (Case Study 1) is a redundant system, that is to say, certain failures of its components are masked. Redundancy can be achieved by duplication of components. In particular, the Flap Controller is duplicated. However, due to complexity model checking and failure analysis could be applied only to one channel at a time. So, essential redundancy was outside the scope of the formal analysis. As a result the Flap Controller was a good example for model checking, i.e. for proving certain safety requirements. But it was less useful for generation of "interesting" fault trees having higher order failure combinations (minimal cut sets).

The same holds for the Sidestick Priority Logic: It is part of a redundant system. However, no special redundancies were implemented in the part being analysed. As a result, the fault-tree analysis revealed a lot of single failures.

Assumptions and Abstractions: Both model checkers allow the definition of additional assumptions (and abstraction for Statemate) when proving properties or doing failure analysis (assertions, freezing, simple assumptions etc.). These may significantly affect the output of the analysis. Currently, for Statemate fault tree generation these assumptions are not mentioned in the output. This prevents traceability of results.

Display of counterexamples: Counterexamples can provide deep insight into (un-intended) model behaviour. However, a better user interface is required to allow the user to decide which information he/she needs and which information he does not need.

The results of the Airbus France/Onera cycle 2 tests:

As mentioned before (see paragraph 6.1.2), for the Airbus France / Onera cycle 2 tests, the Fault Tree automation was not adequate considering the dynamic of the electrical model. So the cycle 2 results concerned only the sequence generation techniques and it was not feasible to progress on the cycle 1 suggestion to analyse when it was more appropriate to use either an Automated Fault Tree or a Sequence Generator.

Lessons learnt from the Airbus France/Onera testing:

As for cycle 1, the sequence generator of Altarica Toolbox is not perfect enough for the dynamic model (either the models that contain their own dynamic, or the static models which include a model of their dynamic activation): the detailed results showed some sequences that were not minimal and few sequences that did not lead to the TLE. So the maturity of this technique alone is not reached with the Altarica/Ocas platform, even if the large abilities to simulate the model can compensate easily this difficulty.

6.1.5. Architecture assessment using Safety Patterns

Cycle 1:

The problem before the tests:

Based on the capability of formal modelling tools and of the model checkers, Airbus France and Onera proposed a new concept to assess qualitatively a system architecture under a safety point of view:

Instead of building a fault tree which allows to identify the scenarios leading to a top level event, the analyst uses safety patterns to match the safety requirements with the items of the architecture. Basically these safety patterns have safety properties (redundancies, command/monitoring,...) ensuring locally in an architecture the qualitative fulfilment of safety requirements. For a considered aircraft system, the assembling of the safety patterns becomes an architecture which underlines the safety requirements. Then, what is locally fulfilled can be assessed more globally on the whole system by using the simulations and the model checking.

In a long term, this view of an architecture can be used directly, not for assessing an existing system architecture, but to build architectures. As a matter of fact, their final interest is not to compete with the traditional approaches as the Fault Tree method, but it is in this last capability to consider directly the safety requirements when building an architecture, particularly during the preliminary architecture modelling.

Within the ESACS project, the investigation on the safety patterns were done to check the equivalence of their results under an architecture assessment point of view compared with the traditional approaches (automated or not) as the Fault Tree method. If this verification cannot find major objections, it is of a big interest to explore also these safety pattern concepts to ensure that the final objectives of architecture building can be met.

In other words, it can be interesting to assess the complexity of behaviour required within a safety pattern. Is it sufficient to start with “simplified” safety patterns to describe a preliminary architecture ?

The results of the cycle1 tests:

Considering a certain model of an aircraft system (here a FoSaM), implementing:

- the normal functioning behaviour,
- the failures of components,
- the system behaviour when failures occur (additional reverse flow implementation when necessary, identification of the temporal and/or functional impact of failure on the system behaviour e.g. leakage progression, etc.),

it was easy from an analyst point of view to identify which kind of safety patterns should be used to underline the qualitative safety requirements:

A qualitative safety requirement (including a quantitative requirement derived into qualitative requirement considering the used technology reliability) could be illustrated by the minimum number of failures which were necessary to loose a function (the function loss should be understood in an extended manner. It had to cover:

Named Distribution Only

- the unexpected functioning, i.e. the functioning in a wrong time,
- and the erroneous functioning, i.e. the functioning with wrong output function values).

A qualitative characteristic of a safety pattern is the number of failures necessary to lose the safety pattern (also in an extended manner, but for simplification reasons, the implementation was done only with the loss of the safety pattern).

So the choice of a safety pattern seemed firstly easy: identification of simplex, duplex, triplex patterns, or more specifically identification of command/monitoring patterns to be matched with the FoSaM components placed to meet the safety requirements.

In an other hand, a safety pattern needs also to include the way it propagates normal functioning and failures, so in a similar concern than, at model level, the system behaviour when failures occur. In other words, the analyst had the choice between safety patterns implementing or not a reverse flow, i.e. the capability to propagate an information (signal, failure,...) in the nominal functioning sense, but also in the reverse sense (bi-directional flow).

So rapidly, the library of safety patterns was enhanced with these different ways: for instance, the initial library was double to consider first the behaviour without reverse flow (case: propagation of the functioning and failure according to a specified nominal way) and secondly the behaviour in case of reverse flow (worst case where potentially all the failures could be propagated since a path existed). As a result, the library was enlarged and could translate potentially a very large number of behaviours.

In the testing, if the starting point was a FoSaM, the choice of the relevant patterns to find the same results than in an automated fault tree generation could be also guided by the choice applied during the building of the FoSaM (if a reverse flow were implemented in the FoSaM, it would be recommended to use safety patterns with reverse flow capability).

This practical rule could not be applied if we imagined the building of an architecture without a FosaM and based only on the use of safety patterns and of safety requirements (which is the long term objective).

In this case, and after the testing, the conclusion appeared clearly that the choice of simplified safety patterns (e.g. with no reverse flow) when no requirement was specified could limit the identification of significant points in an aircraft system, and that the analyst had to start with the worst possible case when nothing was specified. That leads to define safety patterns implementing complex behaviours (reverse flow, unexpected functioning, erroneous functioning,...) and to detail the way to manage/control this complexity (for instance by adding patterns representing derived requirements to constraint the propagation of the failures).

Lessons learnt from the testing:

The lessons learnt from the testing limited to the ESACS scope are good considering the equality of the results between:

- the analysis by simulations and model checking of a system architecture using safety patterns,
- and the (automated or not) traditional safety analyses identifying failure scenarios in a system.

Named Distribution Only

Considering now the long term objective to propose architecture based only on the use of safety patterns and of safety requirements, other methodology concepts should be added to handle the derived safety requirements applying on the items and links within the safety patterns.

In this context, a user friendly platform interface is not our first priority. The opportunity of the second cycle of the methodology application will be used to confirm the above lessons learnt by exploring other parts of the Airbus France case study.

Cycle 2:

The problem before the tests:

The problem dealing with the ability of the current safety patterns to meet the final objectives of building a safe architecture of an aircraft system is the same described for cycle 1 tests.

The results of the Airbus France/Onera cycle 2 tests and lessons learnt from the testing::

The understanding of the failure modes and of their way to be propagated in the system allowed to avoid the safety pattern model problem raised during the cycle 1 tests.

To be more explicit, the current safety patterns including:

- *failures,*
- *the way to propagate normal and abnormal functioning, and more generally a logic of activation in the full respect of a safety property,*

were easier to be represented for the electrical system than for the hydraulic system, where the adverse effect of the environmental conditions of some components was difficult to predict (loss of a hydraulic circuit, leading to an over-speed of a pump, leading to an overheat of another hydraulic circuit). This electrical system complexity was not in the definition of failures and their propagations at a component or a safety pattern level, but much more in the assembly of all the detailed items. It was a good candidate for highlighting the interest of a safety pattern approach.

Consequently, the results were really fine and the reasoning based on the safety patterns for modelling the electrical systems appeared relevant: the current architecture could be reviewed and an approach of construction/analysis using increasingly more and more safety patterns for matching with the safety requirements allowed to identify the safety significant items and architecture pieces of the electrical system. Each step of the analysis was treated firstly by simulation. But it appeared, as explained in the paragraph 6.1.3, that this step was not sufficient because of the number of simulations that it would have been necessary to carry out considering the number of potential states in these models. So the model checking technique was a necessary feature in this safety pattern approach.

New topics of exploration were identified during the cycle 2 tests:

- *can safety patterns be defined and used for modelling the activation logic, that is currently supposed to have no failure/error ?*
- *can the construction of some complex safety patterns (grouping for instance 'back-up' and 'redundancies') be done by aggregation of basic safety patterns or is the construction from scratch is the single alternative ?*
- *can the experimentation of alternative safety patterns to propose new system architectures be replaced by a construction logic to be more formalized ?*

It will be developed inside the ISAAC project.

Other Tests:

For Alenia/SIA, in the context of the first system model development phase, i.e. when the “architecture” of the system is established, it is important to have the possibility to build some “simple” models on the basis of a “component library”. This problem can be summarized as the Building an architectural model using a “component library”.

E.g. on the Statemate line this is related to the facility to build models with “generic charts”. Then, for STSA was asked the enlargement of the STSA scope to include the possibility to do the analysis on models including “generic charts”.

The proposed facility was not implemented due to time constrains. It will be developed inside the ISAAC project.

6.1.6. Bottom up FMEA

The results of the Airbus UK cycle2 tests:

Using the Prover capability it was possible to take an ESM and force certain sets of failures and then test only those sets to see if a top level event could be achieved for each set. This was only possible for the symbolic models used in the evaluation by AUK. Prover were able to produce a very basic result from the Behavioural model.

Lessons learnt from the Airbus UK cycle 2 testing:

The basic testing produces a useful capability. The presentation of results could be improved from the FTA cut-set analysis to aid in preparing the FMEA tables.

The time to complete an analysis is heavily dependent on model size and strategy. AUK is still waiting for the behavioural model analysis to complete. The proof engine is now considerably stable. Unfortunately the strategy has to be applied taking into consideration the duration the analysis will take to reach a result.

6.2. View on the evaluation criteria questionnaire answers

After the focusing on specific points of the test reports, it becomes necessary to present the impact of the testing on the evaluation criteria proposed following the questionnaire organisation.

After Cycle 1, the answers to this questionnaire were synthesized in D14.

As mentioned before it was decided to work directly on this synthesis proposing what is changed according to the cycle 2 tests conducted. The repetitive topics with 6.1 cannot be avoided and are necessary for the readers who skip the preceding chapters.

6.2.1. Method: logical issues

This chapter of the questionnaire addresses the limitations brought by the Model Checker technologies, the comparisons between ESACS methodologies and the traditional approaches like FTA, and the novelties brought by the ESACS approach.

- Model Checker Limitations were reported by the testers using a complex System Model with “real” number values. As a consequence, difficulties were arrived about memory capacity and/or computation time. For the other testers, no major limitations were identified. Restrictions on Model Style were not perceived with the same intensity, but everyone agreed on the fact that special rules had to be followed to reduce the complexity of the system model and avoid inconsistent or inaccurate results.

- ESACS approach provided unquestionable advantages in term of development of formal safety model, interactive simulation (both normal and failed behaviour) and static simulation to assess the system model against safety requirements. Everyone agreed on the improvement of the interaction between design and safety engineers. Automatic generation of FTA or the use of Model Checkers to deal with the safety concerns was considered as a plus. All the partners pointed out that ESACS methodologies suffered from a lack of maturity: few industrial feedback did not allow giving a validated opinion on the difficulty of its implementation in an industrial context. It has been stressed that the today ESACS methodologies required a high skill and training.

- Each partner gave examples of analyses that could be performed thanks to ESACS approach. Especially, all partners pointed out that both the static and dynamic properties could be studied. Examples were provided to justify their position.

Moreover during cycle 2 tests, new features, required at the end of cycle 1 tests, were implemented to allow the user obtaining more useful results from the safety analyses. These new methodological issues are described in detail at paragraph 6.1.4 above.

6.2.2. Software engineering issues

These properties are expressed in term of interoperability with the tools used during the test. Comments on the possibility to integrate additional tools into the platform are given as well.

Named Distribution Only

It should be noted that the answers were limited due to the fact that each tester was concentrated on maximum 2 platforms.

Generally speaking the interoperability was judged satisfactory.

Nevertheless the conversions between different design models (so the interoperability between modelling tools) were not tested by the industrial partners (the translated data were prepared directly by the technology provider partners) and it is worth noting that the definition of a common format will help the integration of new tools within the ESACS platforms. In another hand, the common format cannot be the unique answer: it was reported the industrial concern that a particular safety tool having been upgraded, its interoperability with the rest of the ESACS platform tools was lost.

In any case it had to be said that, during cycle 2, in parallel to the activity of development of new facilities, Technology Partners done some job towards integration.

Examples are the following:

- *translation from Scade into Altarica possible through the Lustre language*
- *Altarica to NuSMV*
- *PPI integrated in OFFIS STSA platform.*

A divergent view appeared related to the backward annotation from safety tools to the modelling tools (e.g. the ability to “evidence” in the system model the “paths” found in doing the FTA). *At the end of cycle 2 tests, this divergence is now reduced: the ESACS platforms, relevant to both the IRST and Statemate implementation lines, implement this backward function. The implementation of this new facility was judged, in general, satisfactory by the testers (Alenia and SIA)*

6.2.3. User interface issues

A synthesis can be found if we consider the level of novelty introduced by a particular ESACS methodology.

So we can classify the methodologies as below:

- a new technique was tested to assess its capacity to identify some safety items. These methodologies concern for instance the use of model checkers either directly to verify a safety property, or indirectly to help to produce minimal cut sets or sequence generations as the results of a classical FTA.
- a new concept was tested to assess its capacity to identify some safety items. For instance, these methodologies concern the use of safety patterns.
- Automation methodology is required to retrieve the final or intermediate results and illustrations obtained with a classical safety analyses. These methodologies are for instance those related to the handling of failure modes, the Fault Tree building and the Fault Tree resolution.

The limits between this classification are fuzzy and a tested activity can concern several classes, but they can support the explanations of the divergent answers.

Named Distribution Only

For the “new” methodologies (new techniques or new concepts), it was noted that a better user interface is not a priority at this level of R&T activity but should be improved for an industrial application.

A preliminary improvement of the user interface has been, in any case, already performed by the Technology Providers during cycle 2 of application of the ESACS project and will be one of the priorities of the new ISAAC project, more addressed towards an integration within the industrial process of the ESACS methodology and platforms.

Related to the automated methodology, the level of maturity of the different tools in each platform (used to answer to each different industrial inputs) influenced the answers and could give the impression that the “new” methods had also a good user interface. Or, at the opposite, focusing on a new technique, trying to present the results as if it was a technique which implemented a classical FTA could seem hard to convince a safety tool user. For instance, the implementation of the classical means of abstraction (filter, hierarchy) in a FTA seems now unsatisfactory in a majority of answers.

It is important to note that, being ESACS, a research project, the priority is to check the validity and applicability of the methodological issues. After having done this, the logical continuation is to proceed in the improvement of the user interface related issues. *As already stated before, this logical continuation is one of the priorities of the ISAAC project, since the user interface is an industrial priority before any deployments of tools inside an organisation.*

6.2.4. New features issues

This section is focusing on weakness and benefits of today’s FTA analyses. This section is completed by the list of benefits of the new methodology.

The main weakness of FTA that was mentioned is the possible misunderstanding and incompleteness of information about the system functionality, leading consequently to the incompleteness of the cut sets involved in the Top Level Event. It is also difficult and time-consuming to keep FTA analysis up-to-date with design changes.

Benefits of such traditional methodologies and associated tools mainly rely on the simplicity to handle them, they are self-contained, computations of the Fault Tree allow to check whether quantitative objectives are met.

Benefits of the ESACS methodology were expressed in term of safety/reliability validation and verification improvement. They concern the architecture assessment, simulation, FMEA/FTA, and timing properties. The safety/reliability validation and verification process implying the safety and reliability requirement trace-ability, it is clear that the preceding positive answers have been confirmed for this second topic also.

In conclusion it can be affirmed that, at the end of cycle 2 of application, the main methodological basis for an integrated and automated safety evaluation on complex systems, have been built.

Then, at this point in time, the main limitation / improvements required by the users concern technological aspects, like the optimisation of the computation algorithm and the improvement of integration among the different tools available.

All these technological issues will be investigated within the ISAAC project and could involve, in the future, the generation of new methodological problems.

6.2.5. Safety process issues

Safety process efficiency was analysed through several point of views:

- The usability of the design model as well as the integration of the design model and safety analysis. The abstraction aspect of the design model was already discussed in paragraph 6.1.1. Related to the integration of system design and safety analysis, everybody recognized that was improved mainly thanks to ESACS, which relied on the principle of safety assessment based on a unique view of the system. Consequently, exchange of information is facilitated between design and safety engineers. This benefit can be taken as soon as the early phases of the system/safety development (FHA, PSSA, SSA).
- The impact of workload both for design and safety engineer. The workload is decreased in the whole process (including modifications arising later in the system life cycle). Everybody believes that there is an increase of the workload for the design engineer (but presumably a reduction in long term) while the workload of the safety engineer will probably decrease. One hypothesis given to explain the increase of the design engineer workload is that early significant points are expected to be discovered by the safety engineer; this will lead to more frequent changes in the design.
- The easiness or difficulty to integrate the ESACS safety process into a current industrial process. It seemed that the integration into current industrial process would take time (the answer given by all partners to the question “short term applicability” was medium and the most optimistic answer is within 1,5 years). The reason is based not only on the necessary improvements revealed by the tests about the maturity of ESACS, but by the knowledge about the patience required to inform and convince users of two processes at a time: the design process and the safety process.

The maturity of the platform and the integration of the ESACS methodology within the industrial process are two of the main objectives of the ISAAC project.

6.2.6. Effectiveness issues

This section mainly concerns the effectiveness of the platform with respect to the use of FTA. This aspect was already developed in paragraph 6.1.3 and 6.1.4.

No gain is expected in term of time to perform FTA the first time. However, the gain could become to the ESACS advantage when considering the time or redo a FTA and would seem easier compared with the traditional approach.

Another key issue is the possibility for ESACS platform to allow hazards or problems to be identified with a better efficiency than using a traditional approach.

It is worthwhile noting that in the ESACS approach there is a chance to find the hazards earlier in the system development process:

- *the automatic verification of the system safety requirements (either by automatic Fault Tree results generation or by the use of safety patterns) would appear to be an efficient way to reach this goal and, by this way, would be good candidates for validation and verification methods.*
- *the undesired output of a system, especially related to timing aspects, could be pre-identified before the system simulations and tests activities would be launched."*

6.2.7. Exploitation issues

Comments were provided with regard to the maturity of the platform as well as the possibility to introduce the ESACS methodology in the industrial process. The partners agreed on the fact that ESACS platform will probably be interesting for the tool vendors provided they will invest important efforts in term of R&T because the platform is currently not mature enough for an industrial use. They yet underlined that the formal methods and techniques are already in use in the industry. ESACS appears as a first continuation of this way, *that will be consolidated by the ISAAC project in view of a partial or, possibly, full integration of the new methodology within the industrial process.*

7. CONCLUSION

At the end of the 2nd WP4 Cycle of application it can be said that the ESACS results are promising in the sense that:

- it was experimented that the way to work on “formal models” and relevant “formal” verification techniques allow the representation of the safety properties and also the verification of that requirements
- it was also experimented that it is possible to derive the “fault tree” in automatic manner for the “undesired event”
- it was checked on varied system technologies the ability to apply at least one of the ESACS approaches (FTA, Sequence Generation, Model Checking or Safety Pattern identification) to ‘safety assess’ a system.
- it was also experimented that it is possible to link the platform result to other tool, e.g. Isograph FaultTree+ in order to do quantitative calculation in that
- the safety process issues are improved, mainly in terms of possibility to detect the hazards earlier in the development process and in the improved exchange of information between the safety and design worlds.
- Moreover, the new facilities, implemented in the second cycle of ESACS methodology application, relevant to the definition of more refined safety analysis and to the generation of backward notation starting from a FT automatically generated, allow the user to find and to debug unexpected FTA results induced by the system dynamic and not foreseen by the safety analyses, coming from an only static view of the system. These potential new failure scenarios will be able to be proposed for the simulations and tests activities to check their consistency on the real system. So a better coverage of the failure scenarios leading to the identified hazards can be expected.

On the other side it has to be said that, at this point in time, the main limitation / improvements required by the users concern technological aspects, like:

- the verification algorithms have to be improved in order to can deal with system model bigger in size and including complex data structure (see paragraph 6.1.1)
- something else has to be done in the direction of the tool integration either for what that concern the conversions between different design models (so the interoperability between modelling tools) and relevant to the definition of a common format to support the integration of new tools within the ESACS platforms.

Moreover:

- some methodological improvements are still necessary relevant mainly to the FTA definition in the ESACS platforms (analysis with restriction on simultaneous failures, fixing of maximum number of failures in a MCS, formalized process to propose new architectures based on different safety patterns...)
- the user interface has to be improved
- the platforms have to be improved to include other facilities use cases related e.g. SAT management or FMEA automation (see Use Cases document).

The intention is to progress in the ISAAC research project with a deeper investigation on both these technological and methodological issues to achieve the goal of a final integration of the ESACS methodology and platforms within the current industrial safety process.

Named Distribution Only

APPENDIX 1 : ALA/SIA CYCLE 2 TEST RESULTS

[Test_Result_2_ALA_SIA_031023.pdf](#)

Named Distribution Only

APPENDIX 2 : SAAB/PROVER CYCLE 2 TEST RESULTS

[Test_Result_Cycle2_SAAB_PROVER.DOC](#)

Named Distribution Only

APPENDIX 3 : AID CYCLE 2 TEST RESULTS

[Test Results Cycle 2 AirbusD.DOC](#)

APPENDIX 4 : AIUK CYCLE 2 TEST RESULTS

[Test-Result-Cycle2-AUK_Simulink-PROVER20031006a.DOC](#)

APPENDIX 5 : AIF/ONERA CYCLE 2 TEST RESULTS

[Test_Result_Cycle2_AIF_ONERA.doc](#)

APPENDIX 6 : QUESTIONNAIRE ANSWERS

AL/SIA

(2 questionnaires: 1 on OFFIS platform, 1 on IRST platform)

[Questionnaire-on-OFFIS-Platform_Cycle2_ALA_SIA_mod031008.doc](#)

[Questionnaire-on-IRST-Platform_Cycle2_ALA_SIA.doc](#)

SAAB/PROVER

(Questionnaire on PROVER platform)

[Questionnaire_Saab_cycle2.doc](#)

AID

(Questionnaire on OFFIS platform)

[Questionnaire_on_Prover_Platform_Cycle2_AirbusD.DOC](#)

[Questionnaire_on_OFFIS_Platform_Cycle2_AirbusD_OFFIS.DOC](#)

AIUK

(Questionnaire on PROVER platform)

[Questionnaire-on-Simulink-Prover-Platform_Cycle2_AUK031010.doc](#)

AIF/ONERA

(Questionnaire on ONERA platform)

[Directly completed on the chapter 6.2 of D18](#)