CONSISTENT SAFETY ANALYSES IN MODEL-BASED SYSTEM ENGINEERING: CONCEPTS AND TOOLS

Authors: Jonathan Dumont, Franck Sadmi, Frédérique Vallée (All4tec)

Keywords: Safety, Dependability, Model-Based System Engineering, FMECA (Failure Mode Effects and Criticity Analysis, FTA (Fault Tree Analysis)

Summary:
Model-based system engineering is nowadays more and more considered as essential to help defining complex systems. Thus it is part of the recommended approaches for implementing system engineering processes.

Model-based safety assessment is also today a way more and more considered in order to improve the safety analyses of complex system.

In this paper we propose an approach that combines both trends and we describe a tool named Safety Architect© that supports the proposed approach.

1. INTRODUCTION

This paper addresses successively the following aspects:

- system engineering and system modeling: the four main categories of technical engineering processes,
- safety analyses of the successive models : how safety requirements can be refined through safety analyses at each modeling iteration,
- a description of ALL4TEC tool Safety Architect©.

2. SYSTEM ENGINEERING AND SYSTEM MODELING

2.1. Transversal concepts and definitions

2.1.1. Block-system

The block-system is the basic decomposition technique used for system engineering. It decomposes a complex system into several hierarchical sub-systems. Each block-system is a complete entity on which the whole engineering processes must be applied.

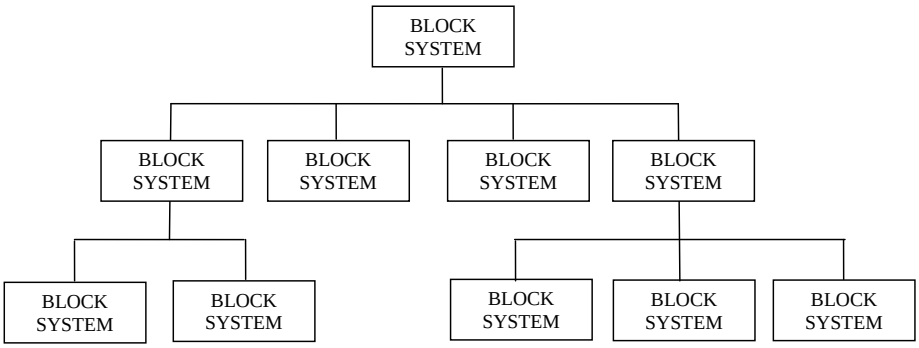


Figure 1: Block systems decomposition

Each block-system has its own complete set of documents produced by the system engineering processes: stakeholders needs document, system requirements document, design document, component needs documents, integration and validation test plans, integration and validation test reports, justification file. At the last level, block-systems also generate component needs documents but the system engineering processes are no more applicable to these components: component design and technological processes must be used at this level.

A block-system is a generic object: in a specific block system the first level is always called “system” and elements of the next decomposed level are always called “component”. The component becomes a “system” when comes the turn to study its corresponding block-system.

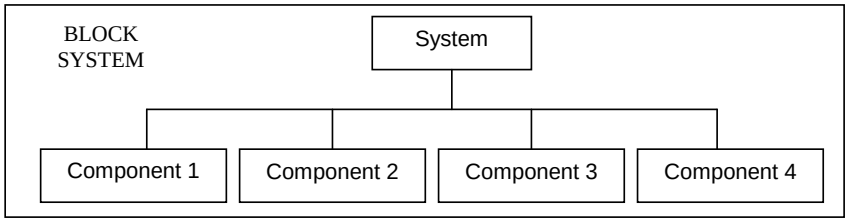


Figure 2 : Block systems and components

2.1.2. Exchanges between block-systems

During the descending phase of the V life-cycle, the only documents transmitted between block-systems are the customer needs documents (what the upper block-system needs) and the system requirements document (what the lower block-system can do). So, each block-system may use its own tools and processes: the only requirement is that the customer needs and the system requirements are the same for the upper and for the lower block-systems. In particular, the traceability has to be managed only inside a single block-system: the only traceability problem between block-systems is the identification of customer needs on one hand and of system requirements on the other hand.

During the ascending phase of the V life-cycle, the physical objects and their validation files rise from the underneath levels.

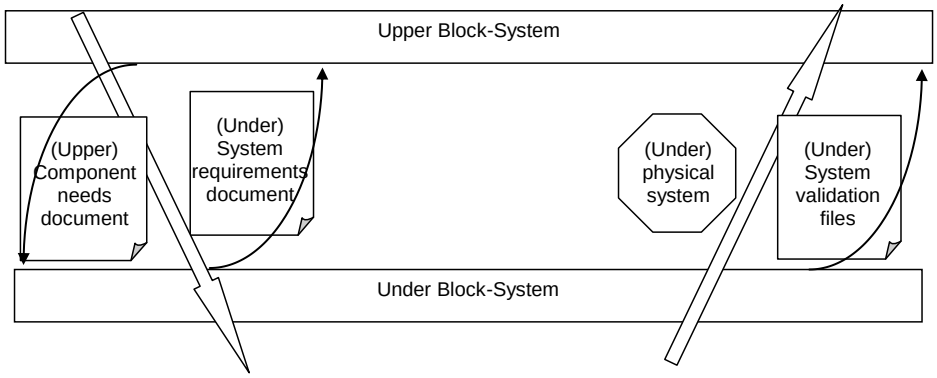


Figure 3: Links between block-systems

2.1.3. Work by iteration

For the block-system development, the system engineering processes are not performed sequentially. Of course, the requirements are always first analyzed, then, the functional architecture and physical architecture definitions are performed. But the work is done through numerous iterations:

- The first iteration covers the nominal operational mode, then the following ones cover the degraded modes, the dependability requirements, etc. until the block-system is completely described.
- A complete and consistent set of documents is established, verified and validated at the end of each iteration. This set is the basis for mock-up or prototyping development, is known and accepted by all stakeholders, and allows the transition to the next iteration with confidence.

2.1.4. Multi-disciplinary teams

By definition, a system team addresses multiple disciplines, otherwise it is a mono discipline team and the system engineering processes are no longer applicable. The person in charge of developing a block-system must rely on diverse disciplines, skills, and knowledge (mechanical, software, hardware, electrical, dependability, quality…). Representatives of all involved disciplines must be represented in the system team.

The system documents must address all multi-disciplinary aspects and must be approved by all system team members (e.g. a specific electrical document can not exist and only be approved by the electrician of the system team).

2.1.5. Documents elaboration

Each engineering document is first a way to communicate information. Thus, the following rules have to be fulfilled:

• Each document must precise its object in order for the reader to make sure that he will find what he needs in the document.

• The document must contain an introduction part to present the context and the main content that will facilitate the document appropriation by any new incomer. A glossary is required for all acronyms and discipline or application domain specific vocabulary.

• The document must be structured to improve the usability: the reader must find quickly what he is looking for.

• All modeling slides or programming lines without comments must not be inserted in the main parts of the document but inserted in appendixes with appropriate annotations.

• An object or a concept must always be identified with the same name in all documents of the same block-system. Do not use synonyms.

2.2. System engineering processes

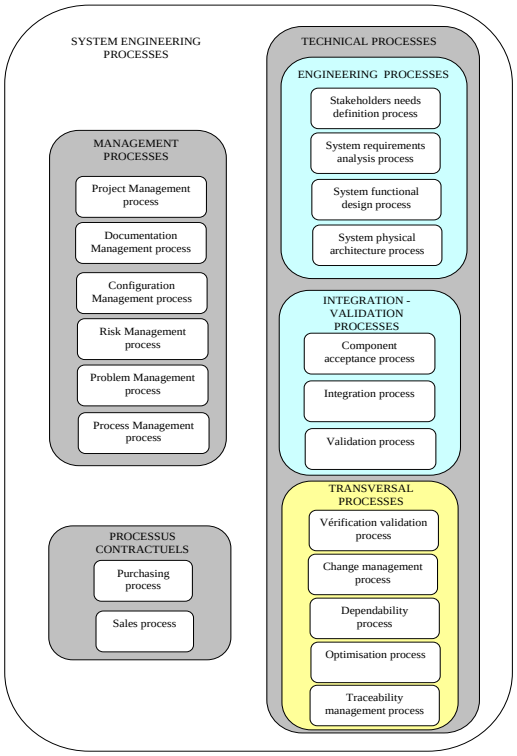


Figure 4: System engineering processes

System engineering processes are useful for the development of complex systems with a lot of functions, flows and interactions between components. They are heavy processes: it is not recommended to use them for a simple system with only two functions.

The system engineering processes cover several application fields: management field, contractual field, technical field. The management field pertains to all project management activities (project planning, project monitoring and control, decision analysis and resolution) and also to all support management activities (documentation management, configuration management, problem management, process management, risk management, etc.). The contractual field pertains to purchasing, sales and supplier management activities. The technical field pertains to all engineering activities.

This paper focuses on the technical engineering processes which are divided into four main categories:

- processes related to the stakeholders needs definition and system requirements analysis activities,
- processes related to the system functional design and the physical architecture definition activities,
- processes related to the component acceptance, system integration and validation activities,
- transversal processes (verification, change management, dependability activities, optimization activities) used as support by the three other categories.

Those technical processes follow a logical order (needs followed by implementation, integration and finally validation) but they are not strictly sequential because many iterations and rework may be necessary in a same project. The processes of the first two categories are defined in more details hereafter.

2.2.1. Stakeholders needs definition process

This process collects and elicits the needs and constraints from all stakeholders which are affected or accountable to the system throughout the phases of the product lifecycle. The needs are identified, selected and classified before being allocated to the appropriate block-systems with the appropriate traceability reference. This process encompasses the following activities:

- Identify the system mission and scope
- Identify the boundaries of the system
- Identify the stakeholders
- Collect the needs
- Structure the input documentation

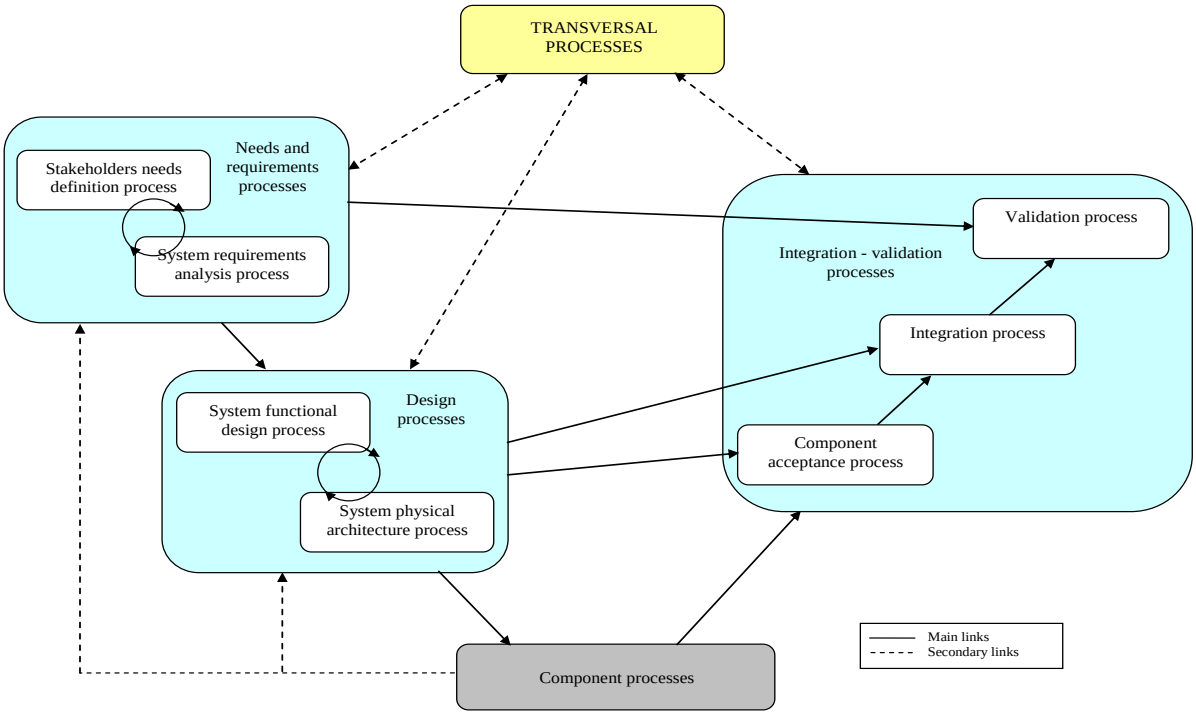


Figure 5: Interactions between technical processes

- Establish the stakeholders needs document
- Trace the needs origin
- Verify and validate the stakeholders needs document

2.2.2. System requirements analysis process

Stakeholder needs are analyzed and translated into system requirements until sufficient details are available to enable design and testing of the system to proceed. This analysis can be instrumented with modelling techniques. System requirement analysis produces functional requirements, effectiveness requirements, external interfaces requirements, operational requirements and constraints applicable at system level. All requirements are not systematically issued from a client need (e.g. re-use constraint or derived requirement issued from other block-systems). Specific attributes are associated to each requirement in order to ease its management (e.g. version, priority, history…). Each requirement is identified and classified thanks to pre-defined criteria. Each requirement is traced from its origin. The system requirements document is submitted to a formal review.

This process encompasses the following activities:

- Build the scenarios
- Identify the operational modes
- Identify the interfaces
- Analyze the documents to find the applicable requirements
- Establish the system requirements
- Classify the system requirements
- Allocate attributes to the system requirements
- Trace the system requirements origin
- Verify and validate the system requirements

2.2.3. System functional design process

The system functional design process is performed according to the two following steps:

- Before the physical architecture definition phase: A functional architecture independent of the physical architecture is defined or an already existing one is reused.
- After the physical architecture definition phase: The functional architecture is updated in order to take into account the physical one.

In this phase, the technical functions required by the system components (e.g. electrical supplies) are integrated and, what is more, the whole functional architecture is reorganized in a way to separate and minimize interfaces between functions allocated to different system components. This decoupling is the first step for the system components detailed design and identify in a precise way all exchanges between components.

The traceability between functional requirements and the associated functional elements (scenarios, operational modes, main functions or elementary ones) is established. The origin of the new functions introduced during the design phase (such as technical constraints functions) is also recorded.

The functional model (architecture and hierarchy definition) is documented in the functional part of the design document and submitted to a formal review.

This process encompasses the following activities:

- Identify the elementary functions
- Describe the operational modes
- Establish the functional architecture
- Establish the functional hierarchy
- Trace the used requirements with the functional elements (modes, scenarios, main or elementary functions)
- Verify and validate the functional design

2.2.4. System physical design process

The system physical architecture process consists in identifying and organizing all components that implement elementary functions. A recommended practice for design is to identify several potential technical solutions and select the most appropriate one thanks to an optimization process. Functions are allocated to components and functional flows are allocated to physical links. The physical model (architecture and hierarchy definition) is documented in the physical part of the design document and submitted to a formal review.

This process encompasses the following activities:

- Identify the candidate physical architectures
- Choose the more adapted physical architecture
- Allocate the functional and the non functional requirements to the components
- Verify and validate the chosen physical architecture

2.3. Modeling approaches

The first two processes may rely on different ways of modelling. Depending mainly on the nature of the system that is tackled, the following modelling approaches and tools may be used: SADT, SART, SysML, Statemate, CORE, UML, Matlab/Simulink, etc.

ALL4TEC aims are to use these models, when they exist, as a basis to perform the safety analyses of the dependability process and thus to both take advantage of a collaborative work and enforce consistency between system engineering and safety engineering.

3. SAFETY ANALYSIS OF THE SUCCESSIVE MODELS

The dependability process is performed in parallel with all other engineering processes and includes safety analyses of the system under design.

During the system requirements analysis phase, all system potential or possible hazards and accidents are analyzed. The potential sources or sequences of events leading to hazards or accidents and their impacts are identified. From this risk analysis, dependability requirements are deduced to define which risk levels are acceptable and which high level safety tests must be performed.

During the design definition phase, the functional and physical architectures are analyzed in order to:

- determine the hazard or accident propagation through the data flow communication links or interfaces,
- evaluate the likelihood and consequence of hazards for all system components and allocate them to safety requirements.

The chosen implementation methods or protection techniques are defined with the corresponding tests to be performed. They are traced as dependability requirements too.

At the end of the engineering process, the "system Feared Events (FE)" have been refined into "Components Feared Events".

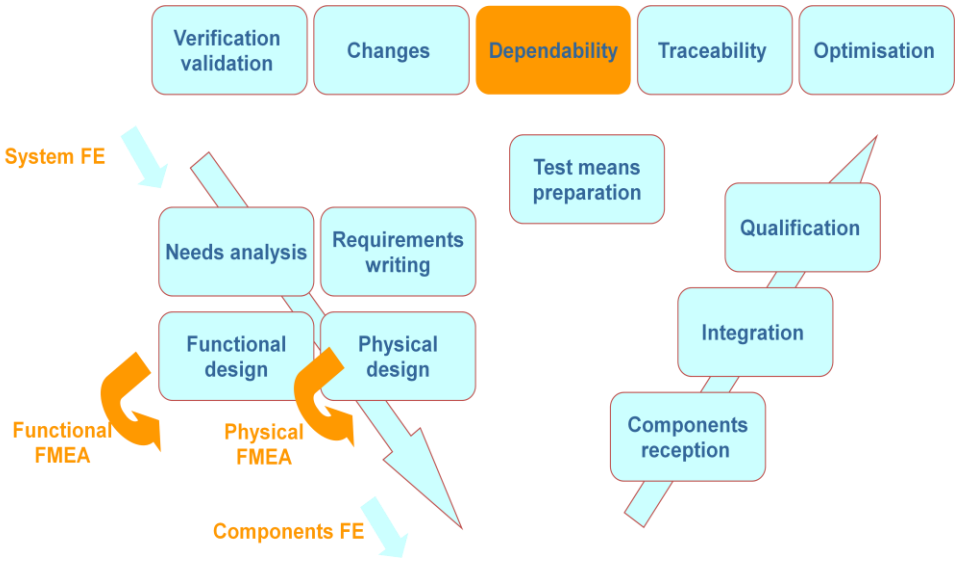


Figure 6: Safety analysis in the engineering process

4. THE TOOL: SAFETY ARCHITECT©

Safety Architect© is a tool dedicated to the safety analysis of preexisting models whatever the nature or the description degree of the model is. In Safety Architect©, safety analysis is handled via a method similar to FMECA (Failure Modes, Effects and Criticality Analysis) and the analysis results are shown in a fault-tree. More details on the tool main principles and functionalities are given in the following paragraphs.

4.1. Main principles

Remind that FMEA aims are to determine, for a given system, components failures that can bring the system into a non-desired state defined as a feared event (FE). Each FE must be identified as a failure mode of one output of the system. The paths formed by failures and their spread in the system leading to a FE are called critical paths. The main principles of using Safety Architect© are summarized in the following synoptic:

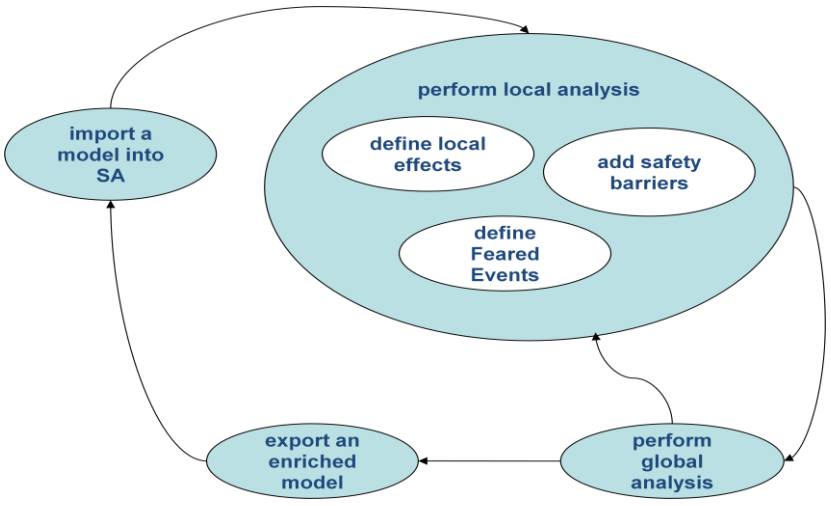


Figure 7: Main functionalities of Safety Architect©

The user must first have a system functional design or its physical architecture model. A **Modeler**, outside Safety Architect©, allows him to construct such a model consisting of blocks connected among themselves and with the environment of the system by oriented streams (in the broad sense, it can be energy, information or dependencies/constraints).

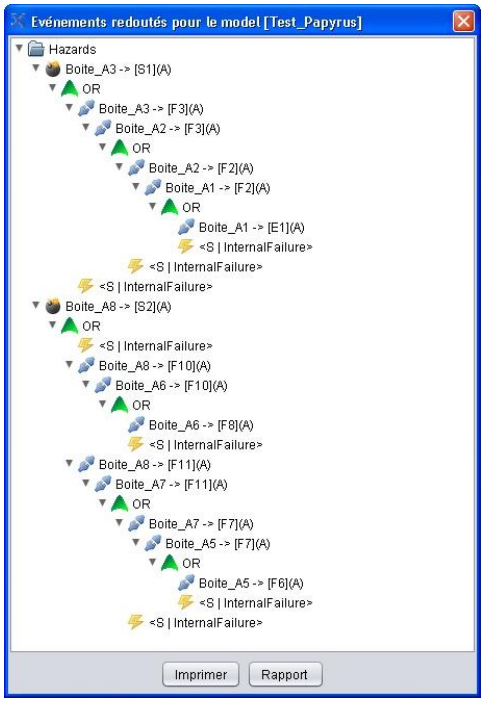
Then, the user must **import this model into Safety Architect©** to perform the safety analysis. The first step consists in performing a "**local analysis**". Therefore, the tool helps the user to define the local effects of each leave block of the model. This consists in linking failure modes of the outputs of the block to the linked failure modes identified on the block inputs. Failure events can also exist on the block itself, and then can be linked to block output failure modes. Safety Architect© performs such a local analysis only on leaf-blocks.

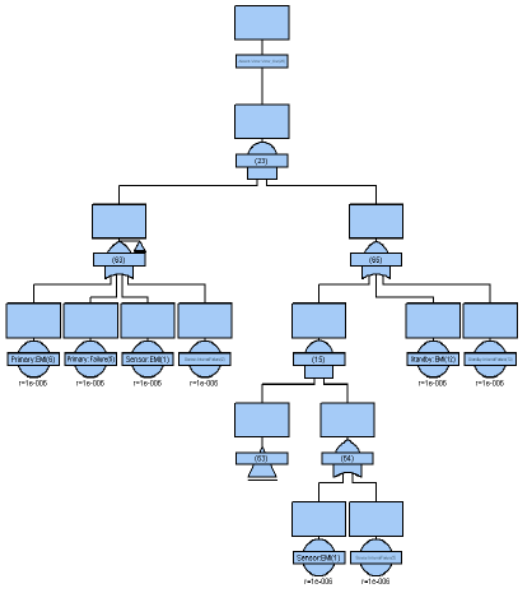
In parallel with local analysis, the user can also **implement safety barriers**. Remind that a safety barrier is a part of a system participating to the safety objectives compliance. This element cut the critical paths and thus prevents the attainment of identified FE. At the physical level, it can be for example a redundancy, and at the functional level, the addition of conditions detecting or correcting errors. With Safety Architect©, two approaches are possible to implement safety barriers: explicitly by the addition of blocks "barriers" in the original system model or implicitly by modifying the local analysis equations.

Then, before launching the overall analysis, the user must specify the failure modes corresponding to the feared events (FE) which will be studied by this analysis.

At this step, the user may request Safety Architect© to achieve the **overall analysis** stage. This step, fully automatic, consists in spreading in the system all the identified failure modes, and to trace those (or combinations of them) that reach a FE.

At the end of this analysis, the failure modes (or their combinations) reaching each FE as well as the paths used by the failure modes to attain the FE are known. When the results are not acceptable, the user can go back to the local analysis and add implicit safety barriers. When the results are acceptable, the user can **export to the Modeler an enriched model**. We call an "enriched model" a system model with dysfunctional data issued from Safety Architect©. The enriched model contains not only all assumptions made during local analysis but it can also summarize the results of the global analysis (for example, non-editable views of the generated fault-tree). This enriched model can be modified in the Modeler and re-imported into Safety Architect© in order to retrieve information reusable in the event of changes.
Integrity checks are run during the import phase.





4.2. Summary of Safety Architect©'s Features

In summary, Safety Architect©'s main features are:

- Importation of the system model
  - With "Papyrus" UML modeler
  - With RSA modeler (Rational System Architect)
  - Through the importation of SysML or UML models
- Local analysis
  - Definition of the feared events
  - Edition of failure modes and their local effects
  - Addition of barriers
- Global analysis
  - Propagation of failure modes
  - FMEA results in ".html" or ".csv" file
  - Minimal fault-tree (non editable)
  - Exportation to a fault-tree tool

4.3. Examples of applications

Safety Architect© has been used for around one year by ALL4TEC and its Clients to:

- Elaborate the safety case of an automotive embedded system,
- Perform the safety analyses of different complex systems or embedded software components used for Defense applications.

For all these studies, the commonly recognized advantage was that the tool is of great help in case of re-work (changes in the requirements or design) and is well adapted to the safety analysis of complex models.

These studies have also given the opportunity to highlight the improvement that should be implemented in the next releases and have been reported to the development team. Some MMI facilities and new functionalities have been added to the tool roadmap.

1. CONCLUDING REMARKS

Provided that model-driven engineering is employed, Safety Architect© is a tool that brings noticeable gain on Safety Analysis quality and costs. For example, ALL4TEC has experimented that the average gain made on FMEA effort was of 50% for an initial FMEA and of around 80% when there was a need of rework of FMEA.

Moreover Safety Architect© can be used in every engineering loop, is independent from any engineering tool, and is able to interface with many engineering tools. It is compliant with usual safety standards such as ISO/CEI 61508, EN 5012x, ESARRs, Future ISO 26262 …It provides an elegant way to ensure that safety analysis is fully compliant with system or software engineering activities and offers a real collaborative work between system or software engineers and safety analysts.