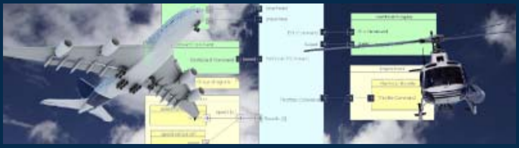


Methodology Handbook

Efficient Avionics Systems Engineering with ARP-4754A Objectives Using SCADE System®

First Edition

CONTACTS

Headquarters

Esterel Technologies SA
Parc Euclide - 8, rue Blaise Pascal
78990 Elancourt
FRANCE
Phone: +33 1 30 68 61 60
Fax: +33 1 30 68 61 61

Submit questions to Technical Support at scade-support@esterel-technologies.com.

Contact one of our Sales representatives at scade-sales@esterel-technologies.com.

Direct general questions about Esterel Technologies to scade-info@esterel-technologies.com.

Discover latest news on our products and technology at www.esterel-technologies.com.

LOCAL SUPPORT SITES

United States

1082 North Alafaya Trail
Suite 124
FL 32826 Orlando
United States
Phone: +1 724-514-2997

Southern Europe

Esterel Technologies SA
Park Avenue - 9, rue Michel Labrousse
31100 Toulouse
FRANCE
Phone: +33 5 34 60 90 50

Central Europe

Esterel Technologies GmbH
Otto-Hahn-Strasse 13b
Ottobrunn - Riemerling
D- 85521 München
GERMANY
Phone: +49 89 608 75530

China

Esterel Technologies
1303, Jiaxing Mansion
877, Dongfang Road
200122, Shanghai
P.R. CHINA
Phone: +86 21 61050287

Shipping date: June 2013 Revision: SSY-HB-ARP4754 - DOC/rev/26003-07

Abstract

This handbook addresses the issues of process, structuring, and productivity in systems engineering for avionics applications. Such projects, driven by the ARP-4754A guidelines, traditionally require iterative multi-disciplinary tasks, incurring high verification efforts and high attention to the preservation of system definition integrity. The handbook reviews the regulatory guidelines and then presents the optimization of the development and verification processes that can be achieved with the SCADE System® methodology and tools, coming with the whole software development capabilities of SCADE Suite® and SCADE Display®, the lifecycle management capabilities of SCADE LifeCycle®, and the multi-domain simulation capabilities of ANSYS® Simplorer®. SCADE System enables the correct allocation of requirements to a given development item and produces architecture, interfaces, and data dictionaries for the various development teams that together build an aircraft system.

Table of Contents

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Figures

List of Tables

1. Introduction

1.1 Background, Objectives, and Scope

Before all professions involved in the Systems Engineering process come on board, a small integrated team gets the extremely sensitive duty to establish what the system should look like at the end.

This team must produce all the necessary artifacts for all the interdisciplinary teams that will work on the System development. Systems Engineering is about creating a system and a plan for it. It is not only a question of conception but also a question of maintaining the integrity of the system throughout its life cycle.

In this handbook, Model-Based Systems Engineering (MBSE) is described as an efficient deployed technique that sets modeling as the primary source of data all along the system development life cycle to improve communication with stakeholders and within engineering project teams, to identify early system requirements issues, to enhance design rigor and integrity, to increase productivity from impact analysis of requirement changes to the reuse of registered models, to reduce risks by enabling requirements validation prior to implementations.

SCADE MBSE solutions are organized around SCADE System and its SysML-based notation.

Demonstration is made that the carefully selected subset of SysML supported by SCADE System allows system engineers to stay focused on their functional and architectural designs, avoiding the complexity of the UML/SysML language.

The handbook shows how MBSE handles the key objective of validating the requirements and highlights how the features of SCADE System facilitate the daily engineering duties, such as integrity checking, traceability management, allocation management, integrating changes, and reporting.

The handbook also explains how SCADE MBSE solutions allow for carefully and safely meeting the objectives of the tables in the ARP-4754A standard [ARP-4754A].

Modeling allows simulation: ANSYS / Esterel Technologies Simulation-Driven Product Development (SDPD) solution is described as the unique optimized solution able to interrelate functional, 0-D and 3-D simulation tools. Virtual simulation is admittedly able to reduce the number of errors found during physical integration and verification activities.

1.2 Presentation

This document provides a careful explanation of the system life cycles as described in the ARP-4754A guidelines. It provides explanations on how model-based techniques enable safe, secure, productive systems engineering practices.

The document is organized as follows:

Section 2. This section provides an overview of the systems engineering life cycle and commonly used best practices

Section 3. This section carefully summarizes the major statements, recommendations, and rules described in the AR-4754A standard: "*Guidelines for Development of Civil Aircraft and Systems*".

Section 4. This section shows the SCADE Model-Based Systems Engineering (MBSE) approach, focusing particularly on the SCADE MBSE V-cycle and on Model-Based tools, SCADE System being at the center.

Section 5. This section thoroughly develops the processes and methodology expounded in the SCADE MBSE V-cycle. For each process, the application of MBSE methodology and tools is detailed. Attention is paid to the elaboration of System Requirements, which is the primary objective of the Systems Engineering process. Integration, Verification, and Validation activities are handled as well.

Section 6. This section draws conclusions that sum up the major benefits of using the SCADE Model-Based Systems Engineering approach.

Appendix A lists and substantiates the ARP-4754A tables of objectives data according to the Integral Processes determined by the standard.

Appendix B provides the list of references.

Appendix C lists all acronyms in use in this document and defines key terminology in a glossary.

Appendix D briefly describes SCADE Suite, SCADE Display, and ANSYS Simplorer tools.

Note that the content of this document applies to the following products:

- SCADE System 2.0
- SCADE LifeCycle 6.4
- SCADE Suite 6.4
- SCADE Display 6.4
- Simplorer 11.0

2. Systems Engineering

2.1 What is a System

Broadly, a system is a set of interacting components that achieves an objective (Figure 2.1). While achieving its objective, a system provides a service to a user (or services to several users), it interfaces with operators, and it works within its environment that is regulating its operation.

The objective of the system is to process exactly and correctly the inputs, the actualization of the functional requirements, and the expression of outputs. To fulfill the objective, the system has to overcome outside forces like Environment, Boundary Crossing, and Interacting Systems.

A system has a boundary that depends on the focus defined by its objective. For example, if the objective of the system is "*to provide runway centerline guidance to the aircraft when landing*", the system is a localizer. But, if the system is meant "*to provide precision guidance to an aircraft approaching and landing on a runway*", the system is an Instrumented Landing System (ILS), that is basically made of two localizers (one lateral, one vertical).

When defining the boundary of a system, one defines what crosses it, including what is expected (defined inputs and outputs of the system, resources, etc.) and what is not (*e.g.*, events that can damage the achievement of the objective such as interferences, shocks, etc.).

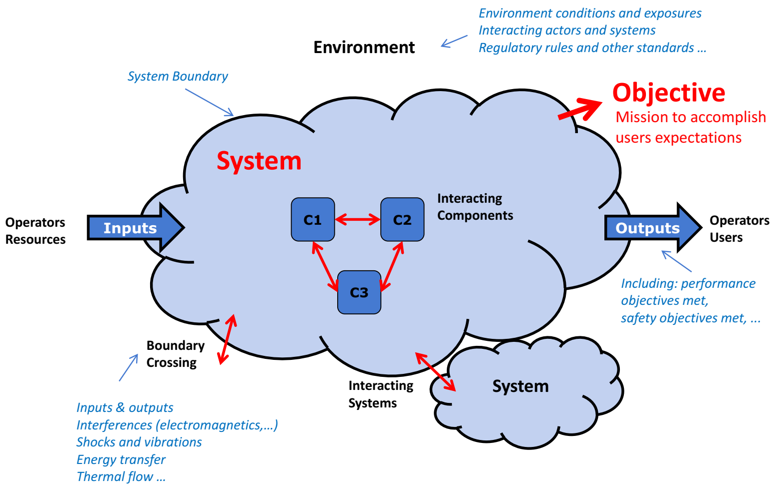


Figure 2.1: What is a system

So a system is defined by what it is doing. That is why Systems Engineering best practices are primarily relying on requirements, scenarios of use, and function decomposition, as described in the next chapters.

2.2 Definition of Systems Engineering

"*Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal. Systems engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.*" [INCOSE]

Systems engineering is… "A*n interdisciplinary, collaborative approach that derives, evolves, and verifies a life-cycle balanced system solution which satisfies customer expectations and meets public acceptability.*" [IEEE P1220]

Systems engineering is primarily a global view that is managing complexity.

Systems engineering aims at implementing a top-down approach but also making a provision for reuse in a bottom-up approach.

Systems Engineering requires contribution from extremely diverse technical disciplines. Thus, decomposing the whole into components and characterizing the interactions among these components is a key to success of Systems engineering.

Systems engineering is both a technical and a management process, together iterative and incremental.

Technical process: develop physical systems that satisfy the customers' needs. It encompasses the design, development, production and operation of these systems.

Management process: organize the technical efforts in the appropriate lifecycle. Its nature is mainly oriented towards problem solving. Systems engineering is constantly looking to increase the probability of success, reducing risks, managing safety and reliability, and optimizing the global life cycle cost.

2.3 Systems Engineering Management

As illustrated in Figure 2.2 below, three major activities contribute to Systems Engineering Management.

Development Planning and Control is the management process that governs the technical processes and provides steps and progressive baselining, from initial concept to the implemented, integrated, verified, and validated system.

The project must be defined and plans issued. Project Plans encompass Statements of Work, Development Plan, Verification & Validation Plan, Quality Assurance, Safety and Certification Plans, Configuration Management and Project Schedule.

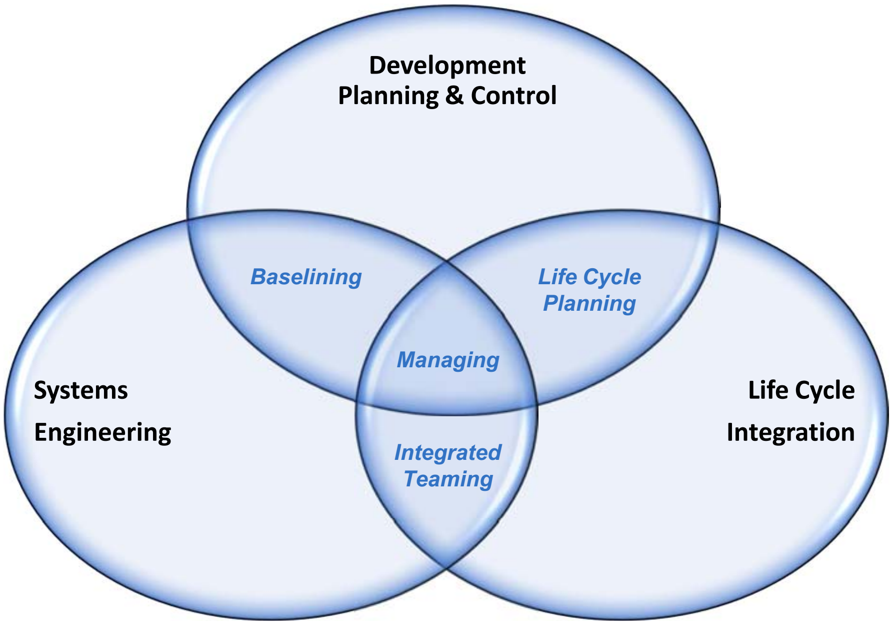


Figure 2.2: Systems engineering management

The project must be carefully controlled (regular assessments and corrective actions) as Systems Engineering is basically a problem solving method. It requires both agility and formal management. Decision making is an essential part of the process, must be precisely defined and planned, and must be based on trustable data from the interdisciplinary teams.

Lifecycle Integration is the necessary process that ensures that the developed system is viable all along its life cycle. It involves the multidisciplinary teams and the customers/ users. The objective of this management process is to streamline the interactions among teams, to ensure the incorporation of inputs from specialized areas such as safety, risk management, business, resource allocation, and finance in the system development project management.

The Systems Engineering Process represents the technical core process. This is the main subject of this document that is detailed in the following chapters.

Basically, the Systems Engineering process is mainly a top-down, comprehensive problem-solving process. But it is as well iterative and recursive as described in the next sections.

2.4 Systems Engineering Process

There are multiple ways to show a Systems Engineering process. Figure 2.3 below shows a V-Cycle representation.

The V-Cycle is a natural representation of the design process that firstly shows a top to bottom path, from the establishment of the system requirements to the design of components, followed by a bottom to top recovery path that enables step by step integrations and verification and validation activities.

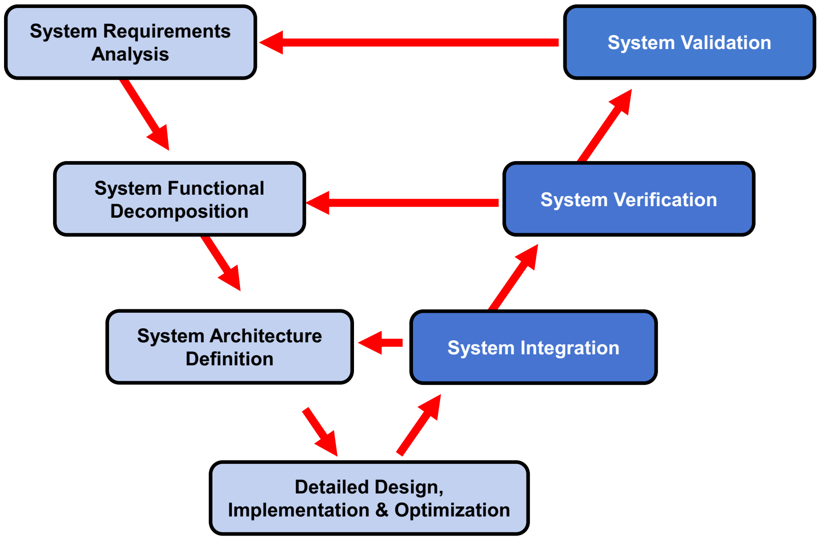

System Requirements Analysis
System Validation
System Functional Decomposition
System Verification
System Architecture Definition
System Integration
Detailed Design, Implementation & Optimization

Figure 2.3: Typical systems engineering process

2.5 From System Requirements to Architecture Definition

This section covers the System Requirement Analysis and System Functional & Architecture Design processes of the above V-cycle (Figure 2.3).

Requirements are the primary focus of the systems engineering process (systems engineering process shortened purpose is to transform requirements into designs).

As shown in Figure 2.4 below, this portion of the Systems Engineering process covers the following main connected activities:

- Requirements Analysis
- Functional Decomposition
- Architectures Design
- Synthesis and Selection

These activities are both technically-driven and management-driven.

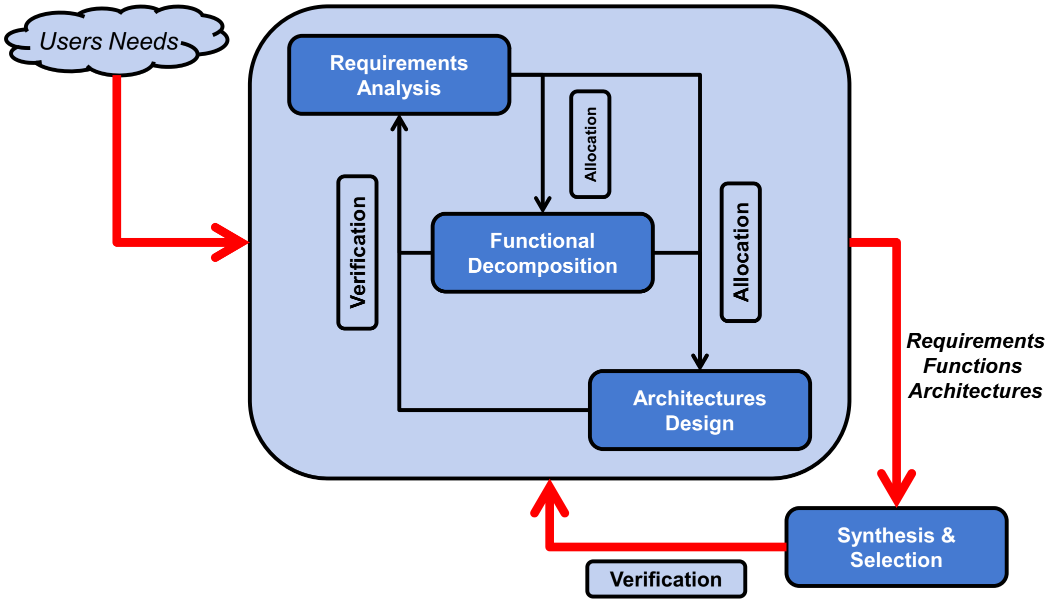


Figure 2.4: Technical activities from Requirement to Architecture selection

A set of top-level stakeholders needs, objectives and requirements, sometimes informal, are translated into product functional and non-functional requirements. An initial representation of a future system is built that does not imply any specific implementation.

These requirements can lead to a few alternate architectures and designs for the end product.

Each requirement is iteratively examined for validity, consistency, desirability and attainability. With these examinations, or evaluations, a decision can be made on the design.

The selection of the preferred System Architecture is essentially a trade-off among the various architecture options. The selected baseline System Architecture should be robust and satisfactory close to the theoretical optimum in meeting requirements, with acceptable risk and within available resources and budget.

Process inputs include:

- Users' needs, objectives and requirements
- Technology base
- Standards
- Requirements from prior developments
- Program decision requirements

Requirements Analysis activities include:

- Elicit the requirements, creating the set of requirements from the needs expressed by users and other stakeholders
- Analyze the concept of operations (missions of the system)
- Identify the environmental constraints and standards that are limiting the solutions
- Identify the functional and non-functional requirements
- Validate the requirements, determining their completeness and correctness

Functional Decomposition activities include:

- Decompose requirements to a hierarchy of functions
- Allocate performance and other constraints to all functional levels
- Define functional architecture
- Manage traceability between Requirements and Functions
- Define functional interfaces

Architectures Design activities include:

- Define physical architecture implementing the functions
- Define alternative architecture concepts
- Allocate Functions to Components of the architecture
- Manage traceability between Requirements and Components
- Define physical interfaces

Synthesis and Selection activities include:

- Study trade-offs
- Manage risks, safety and performance
- Select preferred architecture
- Make and report decision
- Set up configuration, baselines, traceability and all necessary data
- Verify the selected architecture: costs analysis, COTS, reuse of in-house or COTS components, impact of organization are taken into consideration for the selection of the appropriate architecture

Process Outputs include:

- Decision database
- Selected architecture of components
- Specifications and baselines

2.6 Detailed Design and Implementation

This section covers the System Design and detailed Design & Optimization processes of the above V-cycle (Figure 2.3).

This is the Implementation part of the overall Systems Engineering process.

"*During the Implementation Process, engineers follow the requirements allocated to the system element to design, fabricate, code, or build each individual element using specified materials, processes, physical or logical arrangements, standards, technologies, and/or information flows outlined in detailed drawings or other design documentation. Requirements are verified and stakeholder requirements are validated. If subsequent configuration audits reveal discrepancies, recursive interactions occur with predecessor activities or processes, as required, to correct them.*" [INCOSE]

The hardware (electromechanical, hydraulic, etc.) and software components (also called: elements or items) are developed according to the appropriate process standards.

Please consult our Model-Based Software Development and Verification handbooks for software components that are developed with SCADE Suite and SCADE Display.

2.7 Integration, Verification, and Validation

This section handles the Component Integration & Verification, System Integration & Verification, and System Validation processes.

"*The purpose of the Integration Process is to assemble a system that is consistent with the architectural design. This process combines system elements to form complete or partial system configurations in order to create a product specified in the system requirements. [...] The purpose of the Verification Process is to confirm that the specified design requirements are fulfilled by the system. This process provides the information required to effect the remedial actions that correct non-conformances in the realized system or the processes that act on it. [...] The purpose of the Validation Process is to provide objective evidence that the services provided by a system when in use comply with stakeholders' requirements, achieving its intended use in its intended operational environment. This process performs a comparative assessment and confirms that the stakeholders' requirements are correctly defined. [...]. System validation is ratified by stakeholders.*" [ISO/IEC15288]

As shown in Figure 2.5 below, integration is iterative and progressive. Each level of integration is built on the top of the previous level of tested integration.

It is essential that interfaces between components, between sub-systems, and between the system and other systems and/or its environment are well known.

Defining and maintaining those interfaces is a primary responsibility of Systems Engineering.

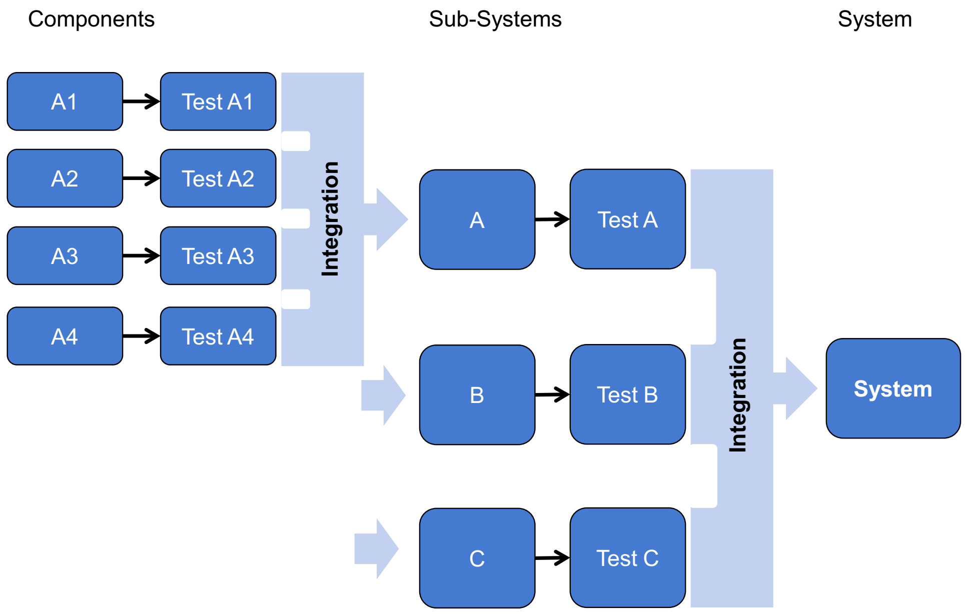


Figure 2.5: Progressive nature of integration

Figure 2.6 below shows the difference between Verification and Validation and how testing activities are doing both.

System Verification ensures that the system has been built correctly.

Verification techniques involve:

- **Inspection**: engineer performs an examination of the system against the requirements (useful for observable properties such as: color, weight, etc.)
- **Analysis**: it is used when real conditions of the verification cannot be established. Analytical data are used or simulation in very precisely defined conditions is performed
- **Demonstration**: a set of inputs that is sufficiently covering the expected operation is used to demonstrate that the required functions are performed
- **Test**: tests are conducted with respect to the requirements and are meant to obtain accurate quantitative results that can be analyzed against expected accurate results

System Validation ensures that the built system is the right system. This is why users and other project stakeholders are involved. In particular, the built system must achieve the business goal which determines that the initial concept has been transformed into an operational system.

Validation is concerned with checking that the system meets the customer's actual needs. Validation is a rather subjective process. It involves making assessments of how well the (proposed) system addresses a real-world need.

Validation starts early in the process as it includes activities, such as requirements modeling (with model analysis) and / or simulation / prototyping for user evaluation.

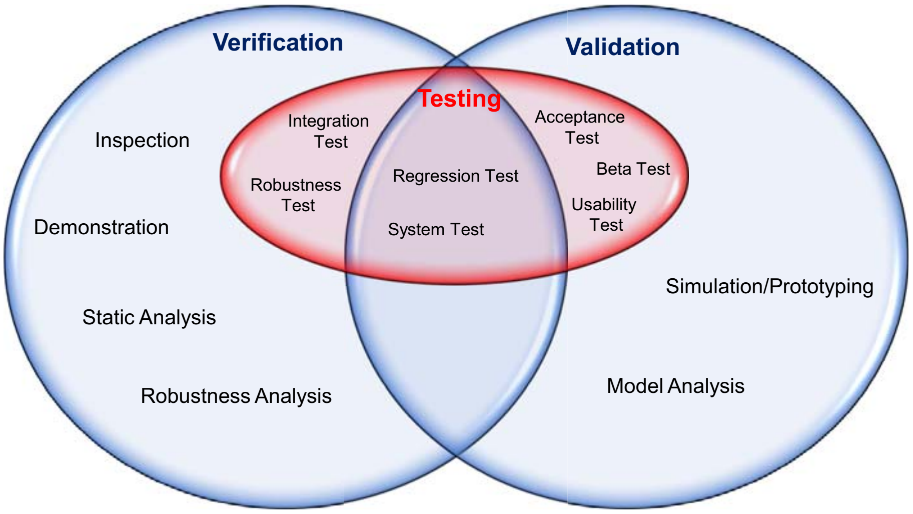


Figure 2.6: System Verification and Validation

3. ARP-4754A

3.1 The Standard

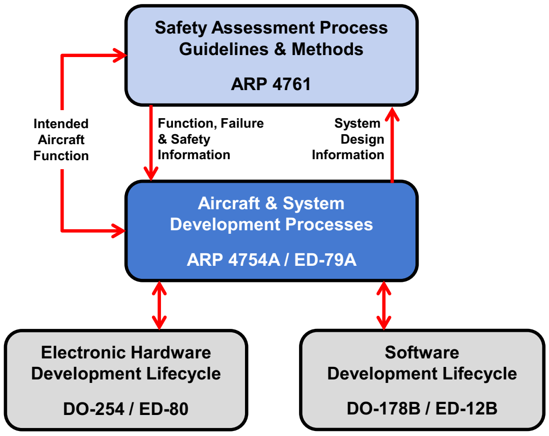

Safety Assessment Process
Guidelines & Methods
ARP 4761
Intended Aircraft Function
Function, Failure & Safety Information
System Design Information
Aircraft & System Development Processes
ARP 4754A / ED-79A
Electronic Hardware Development Lifecycle
DO-254 / ED-80
Software Development Lifecycle
DO-178B / ED-12B

Figure 3.1: Global view of the ARP-4754 positioning

The assurance of ARP-4754A is two-fold:

- Validate that the requirements are correct and complete
- Allocate those requirements to software and electronic hardware items ("item" is the term used for component)

While the DO-178B/C standard drives the development of software items, the DO-254 drives the development of electronic hardware items.

These standards assume that the requirements are correct and complete, while their assurance is two-fold as well:

- Develop the software and hardware electronics
- Verify that the software and hardware meet their requirements

3.2 Aircraft and System Development Process

ARP-4754A standard primarily focuses on establishing good requirements. These requirements are elaborated so that they can be provided to the teams in charge of the development of items.

An item is a component of a system that can be developed by a team that manages a given discipline: software, electronic hardware.

Figure 3.2 below shows how the ARP 4754A standard represents the global System Development Process.

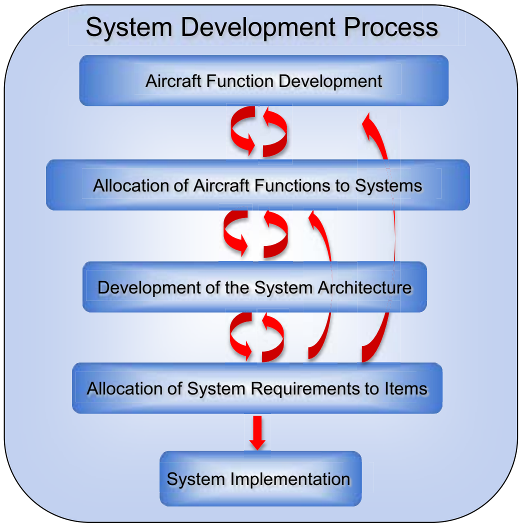

System Development Process
Aircraft Function Development
Allocation of Aircraft Functions to Systems
Development of the System Architecture
Allocation of System Requirements to Items
System Implementation

Figure 3.2: System Development Process according to ARP-4754A

3.2.1 Aircraft Function Development

Aircraft Function Development is the process of identifying aircraft-level functions, function requirements, and interfaces.

"*Iteration*" is the key word of the System Development Process. For adding functions to an aircraft, the entry point may occur in the context of changes to a particular item.

"*Most actual system developments processes involve many iterative cycles, making the experience appear more cyclic than sequential. The entry point for aircraft function implementation may occur at any point in the cycle. For a new aircraft-level function, the process begins with the top-level definition of functions. […] In practice many of the development activities shown […] are concurrent and may involve interactive dependencies that lead to alteration of previously established requirements.*" [ARP-4754A]

3.2.2 Allocation of Aircraft Functions to Systems

Allocation of Aircraft Functions to Systems consists of grouping aircraft functions and assigning them to main "systems". This activity includes functional decomposition. The output of the process is a set of requirements for each defined system.

The standard pinpoints that it is a very important and complex activity:

"*The functional groupings interact with the aircraft architecture and are the basis for system architecture development. While it is not necessary to know in detail how a system will be implemented to accomplish the necessary functional groupings, implementation constraints, failure effects and lifecycle support may all play a significant role in selecting the most appropriate groupings.*" [ARP-4754A]

3.2.3 Development of the System Architecture

Development of the System Architecture is about building the structure of the interconnected items and boundaries that meet the above defined requirements, including safety requirements and performance requirements.

"*More than one candidate system architecture may be considered for implementation. These candidates system architecture may be evaluated using such factors as technology readiness, implementation schedules, producibility, contractual obligations, economics, prior experience, and industry precedence.*" [ARP-4754A]

3.2.4 Allocation of System Requirements to Items

Allocation of System Requirements to Items is done concurrently to the development of the system architecture.

Several iterations are necessary to solve derived requirements (*i.e.*, requirements that appear during design) and make the allocation of requirements to items as clear as possible. Iterations can evolve up to refine the system functions development and the associated allocation of functions to the system.

"*The process is complete when all requirements can be accommodated within the final architecture. The decomposition and allocation of requirements to items should also ensure that the item can be shown to fully implement the allocated requirements.*" [ARP-4754A]

Thus, the accomplishment of this allocation means that the appropriate architecture of items has been selected; this is the architecture that meets requirements while being as close as possible to the other project requirements, constraints, and opportunities.

3.2.5 System Implementation

System Implementation is about developing the items that form the system. Only electronic hardware and software items are considered. This implementation is driven by other standards:

"*The point where requirements are allocated to hardware and software items is also the point where the guidelines of this document (ARP 4754A) transition to the guidance of DO-178B/ED-12B (for software), DO-254/ED-80 (for electronic hardware), and other industry guidelines. This document provides guidelines for architecture, development assurance level, and functional decomposition including redundancy management. This means that the requirement allocation to hardware and/or software has been reached at the point when architecture, redundancy management and requirement decomposition are complete.*" [ARP-4754A]

It is important to notice that System Implementation, as defined by ARP-4754A, encompasses the hardware and software design/build, the electronic hardware/software integration activities, and the aircraft/system integration activities.

HARDWARE AND SOFTWARE DESIGN/BUILD

"*The outputs of this phase include electronic hardware-software integration procedures, released hardware drawings, software source code together with related life cycle data, applicable development assurance data, breadboard or prototype hardware if applicable, and lab/ flight test articles.*" [ARP-4754A]

ELECTRONIC HARDWARE/SOFTWARE INTEGRATION

Mutually prototyping one part for the other, virtual simulations, and simulation platforms are integration methods considered by the standard.

"*The output is equipment under configuration control together with development assurance data and/or software life cycle data.*" [ARP-4754A]

AIRCRAFT/SYSTEM INTEGRATION

System integration is the step-by-step process of integrating and testing the integration of the items of a system.

Aircraft Integration's main duty is to ensure that all systems that have been proved to separately operate correctly are still working well together.

3.3 Integral Process

While the overall System Development Process described in Section 3.2 represents the path from a concept to implementation, the Integral Process described in this section represents the concurrent processes that have interactions at any stage of the system development.

Figure 3.3 below shows all the process elements that constitute the Integral Process. It shows at which stage of the engineering development they have interaction: item level, system level, aircraft level. These processes are described in the following sections.

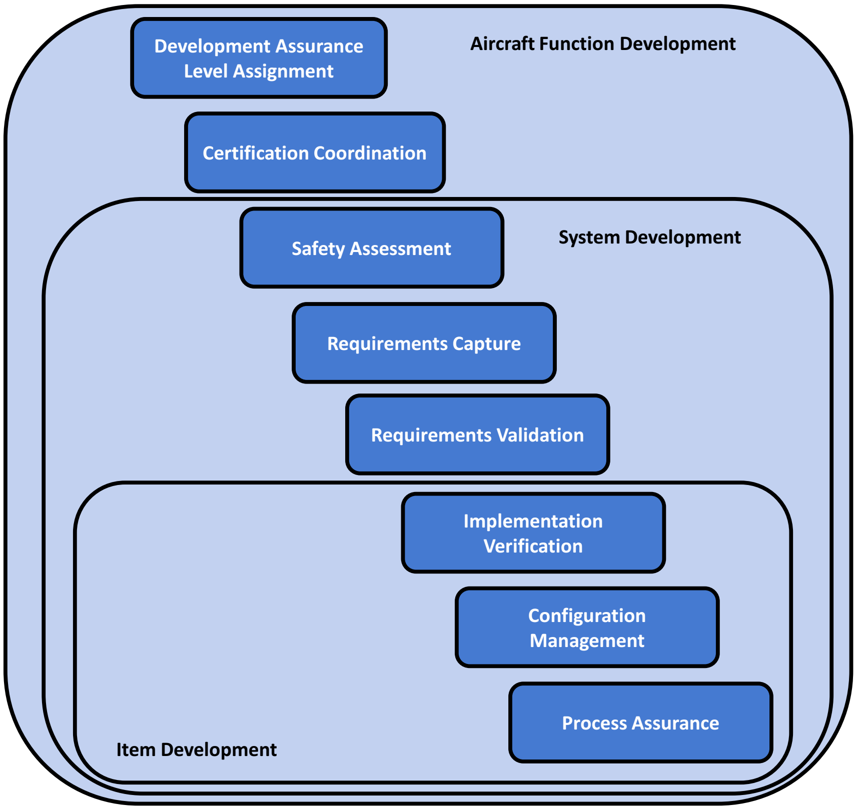


Figure 3.3: ARP-4754A Integral Process

3.3.1 Safety Assessment

This process encompasses a large part of the ARP-4754A objectives. It contributes to the standard’s main focus: establish good requirements.

The Safety Assessment process ensures that requirements elaborated for safety reasons are correctly and completely considered.

Details on the Safety Assessment activities are provided in the ARP-4761 standard [ARP-4761].

Figure 3.4 below shows how the Safety Assessment activities are interacting with the System Development Process.

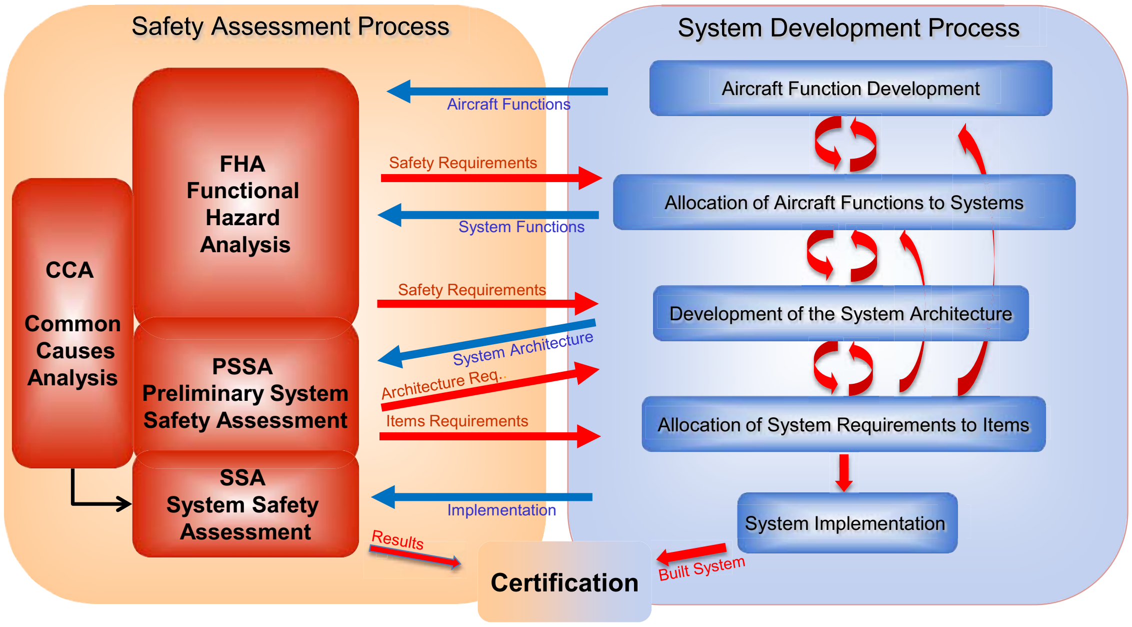


Figure 3.4: System Engineering Process and Safety compliant with ARP-4754A

At Aircraft /System levels, a Functional Hazard Analysis (FHA) is primarily performed. The process enables the identification of potential functional failures (of Aircraft and System Functions) and classifies failure conditions based on the identified effects.

At Aircraft / Systems levels as well, Preliminary Safety Assessments are elaborated (PASA and PSSA). PSSA is about determining how failures could cause the Failure Conditions of the above FHA. PSSA establishes the system or item safety requirements and provides preliminary indication that the system architectures can meet those safety requirements.

It is at this level that protective strategies can be incorporated in the development process, such as: partitioning, redundancy, built-in tests, monitoring, independence, etc.

The System Safety Assessment (SSA) collects, analyzes, and documents the verification artifacts that demonstrate the system, as implemented, meets the safety requirements established by the PSSA.

Fault Tree Analysis (FTA: deductive top-down failure condition analysis) and Failure Mode end Effect Analysis (FMEA: inductive reviewing of items and systems to identify failure modes, and their causes and effects) are implemented at this stage.

Common Cause Analysis (CCA) establishes and verifies physical and functional separation and isolation and independence requirements between systems and items, and verifies that these requirements have been met.

CCA relies on all Safety Assessment processes. In particular, a Common Mode Analysis is performed. It verifies that all failures identified by the SSA are independent from each other in the System architecture under evaluation.

3.3.2 Development Assurance Level Assignment

This is the process of assigning a Functional Development Assurance Level (FDAL) at Aircraft/System level, and consequently attributing an Item Development Assurance Level (IDAL) to each item that constitutes the Aircraft Function.

The table below shows the DAL assignment levels that must be performed for each function in the aircraft.

These assignments are creating the necessary rigorous basis for the assurance process applicable to the development of functions/ systems/items.

Table 3.1: DAL assignment levels by aircraft function

Top-level failure condition severity classification	Associated top-level function FDAL assignment
Catastrophic	A
Hazardous/Severe Major	B
Major	C
Minor	D
No Safety Effect	E

3.3.3 Requirements Capture

Requirements capture can be textual or graphical.

“*When graphical requirement capture is planned, the following topics should also be included:*

- *Identify the use of models/modeling*
- *Identify the intended tools and their usage during the development*
- *Define modeling standards and libraries in order to establish a common understanding of the use of models*

Models used to capture requirements and then directly used to produce embedded code (Software or HDL) come with the scope of DO-178B/ED-12B and DO-254/ED-80, from the time that certification credit is to be taken until the software or hardware is returned to the system processes for system verification.” [ARP-4754A]

Requirements can be:

- Functional requirements: those that describe the desired functions of the system. They are constrained by operation conditions, regulatory restrictions, and real life.
- Customer requirements: those that drive variants of the aircraft
- Operational requirements: those that define the interface between the various crews and the aircraft (flight crew, maintenance crew, etc.)
- Safety requirements: they are coming from the Safety Assessment Process
- Performance requirements: define how the functions of the system are providing good service
- Installation requirements, maintenance requirements, physical interface requirements
- Certification requirements: they may be required to demonstrate compliance with airworthiness regulations

3.3.4 Requirements Validation

A validated requirement is good and complete, and correct.

A good and complete requirement is:

- Necessary: it is not a simple remark, redundancy of requirements is avoided
- Implementation-independent: it specifies the “what”, not the “how”
- Clear: it is not ambiguous, meant to be communicated
- Thorough: no amplification is needed
- Consistent: it is compliant with the applicable standards
- Achievable: it can be implemented by the designers
- Traceable: it can be allocated to a higher level requirement (*i.e.*, to the stakeholders’ needs)
- Verifiable: by inspection, analysis, demonstration, or test

A correct requirement is a requirement that is well understood and feasible.

Among usual techniques, such as checklists, traceability matrices, etc., the ARP-4754A standard mentions that prototyping can be an appropriate means for validating requirements.

“*Prototypes are models of the desired system that may be hardware and/or software based, and may or may not development versions of the system. Prototypes permit users of a system to interact with a proposed model of the* *system to uncover missing requirements, behaviors of the system that should be prohibited, and potential problems with user interaction.*" [ARP-4754A]

The requirement validation rigor (including independence in the validation process) is driven by the determined FDAL (see tables in Appendix A).

3.3.5 Implementation Verification

The verification process confirms at system level that the required functions have been correctly implemented, so that the requirements have been satisfied, and ensures that the results of the safety analysis are still valid after the implementation.

Verification method, techniques and data are those described in the section 2.6

3.3.6 Configuration Management

"*Configuration management is both a system development and a certification activity. A configuration baseline should be established at the time in the system development process when requirements compliance substanciation is first desired. The traceability of the final proposed configuration to, that configuration baseline is a necessary element of demonstrating development assurance.*" [ARP-4754A]

Configuration includes:

- System and items requirements
- Applicable certification data listed in Section 3.3.8
- Tools and utilities
- Any other data that contributes to define the system or items

3.3.7 Process Assurance

The Process Assurance verifies that the plans and standards are developed, verifies that the development is performed in accordance with those plans, and establishes reports of these verifications.

3.3.8 Certification Coordination

This process starts with the Certification planning and has the objective to provide evidence that the aircraft, system, item satisfy airworthiness requirements.

At item level, DO-178B/C or DO-254 are applicable.

Table 3.2: List of certification data at system level

Certification Plan	summary of the main subjects and plans to manage for certification
Configuration index	table that identifies all the items and interfaces of the system
Development Plan	identify the overall process that leads the system development
Design Description	describe the operations and capabilities of the system
Validation Plan	ensure that the specified requirements are sufficiently correct and complete
Verification Plan	ensure that the requirements have been correctly and completely satisfied (including safety requirements)
Configuration Management Plan	set up the method to manage the configuration of elements that make up the system and its items
Process Assurance Plan	describe the means to verify that guidelines and procedures are correctly applied
Functional Hazard Assessment (FHA): described below	(described in Section 3.3.1)
Preliminary Aircraft/System Safety Assessment (PASA/PSSA)	(described in Section 3.3.1)
Aircraft / System Safety Assessment (ASA/SSA)	(described in Section 3.3.1)
Common Cause Analysis (CCA)	(described in Section 3.3.1)
Validation Data	results of the Validation Plan
Verification Data	results of the Verification Plan
Evidence of the Configuration Management	results of the Configuration Management Plan
Evidence of the Process Assurance	results of the Process Assurance Plan
Certification Summary / Compliance Report	results of all the Certification Plan (summary of results of all plans)

4. SCADE Model-Based Systems Engineering

4.1 Definition of MBSE

"*Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.*" [INCOSE]

"*A model is an approximation, representation, or idealization of selected aspects of the structure, behavior, operation, or other characteristics of a real-world process, concept, or system.*" [IEEE 610.12-1990], *i.e.*, an abstraction.

"*A model usually offers different views in order to serve different purposes. A view is a representation of a system from the perspective of related concerns or issues.*" [IEEE 1471-2000].

MBSE formalizes the practices of systems engineering through the use of models and organizes models-centric activities instead of documents-centric activities.

As shown in Figure 4.1 below, the main risk throughout the development process of a product is the risk of inconsistencies, in particular between the system that has been specified by system engineers and the system that has been realized by the interdisciplinary technical team. In addition, support processes, such as Safety Assessment, Process Assurance, and Certification have impacts throughout the development process: addition of iterations and increase of risks.

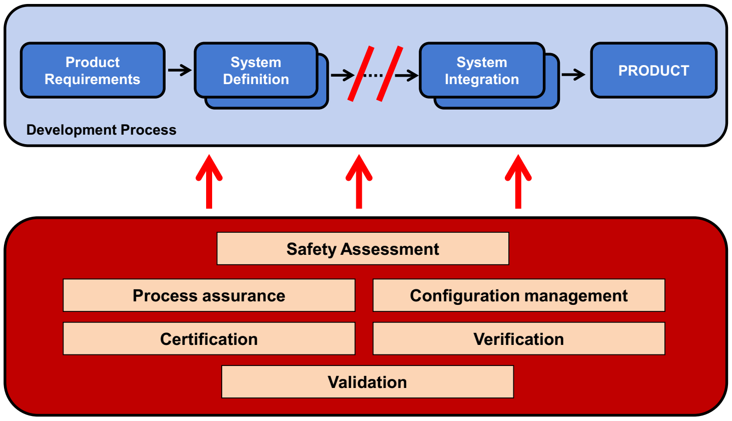


Figure 4.1: Problem of consistency along the development process

Most of the early information that governs the system development can be collected into models and tables that can be shared among teams, encouraging cooperation over various technical disciplines.

Model-based approaches are made for communication, integration of changes, and safe management of key artifacts all along the systems engineering life cycle.

Model-Based Systems Engineering improves quality and productivity, ensures systems engineering artifact integrity throughout the system development life cycle and, finally, optimizes the management of the development, reducing risks and facilitating decision making (Figure 4.2).

Improve quality and productivity:

- Enable through simulation early discovering of operations and requirements
- Enable early identification of requirements issues and validate the correctness of the requirements
- Improve the allocation of requirements to items
- Establish rigorous requirements traceability, facilitating early verification of requirements completeness
- Speed up safe impact analysis of requirements changes

Ensure integrity and reliable communication:

- Create a single reference point that aggregates views and key data of the project
- Enable early design verification
- Auto-generate documentation

Optimize Systems Engineering management activities:

- Facilitate the reuse of existing valid models
- Enable optimized iterations and valuable increments
- Allow early problem solving activities
- Improve cost estimates

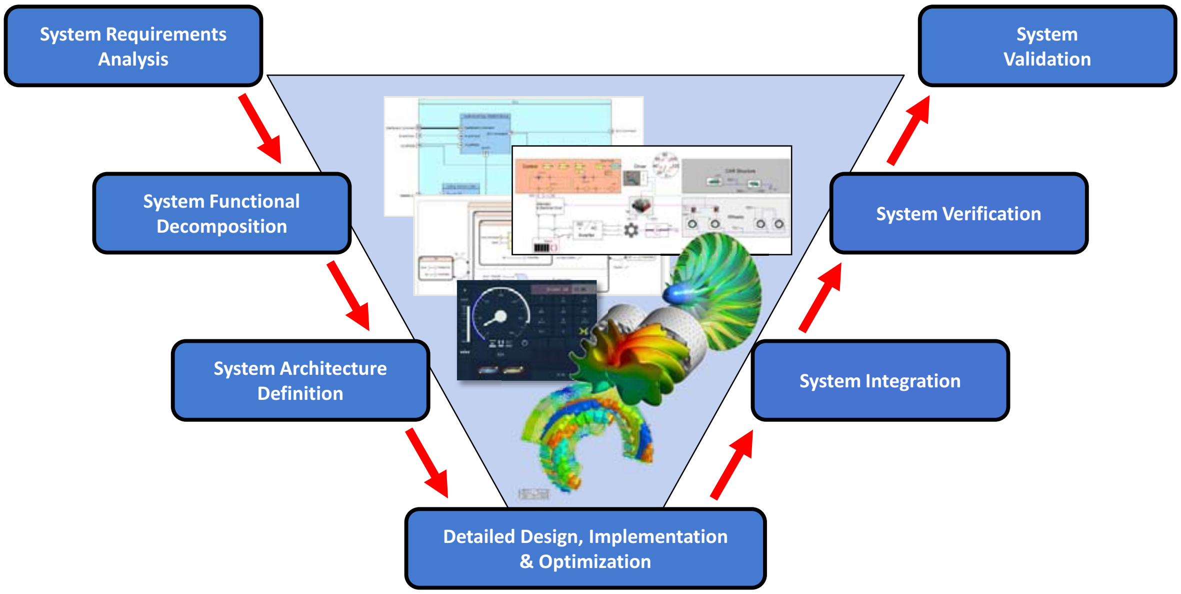

System Requirements Analysis
System Functional Decomposition
System Architecture Definition
Detailed Design, Implementation & Optimization
System Integration
System Verification
System Validation

Figure 4.2: Driven by models

Utilizing a Model-Based Systems Engineering approach is even more beneficial.

Systems Engineers recognize that elaborating and maintaining the Interface Control Documentation (ICD) is such an important and difficult task that it is a cost driver of the whole system development process. The successful integration of components into a system depends on an early dependable ICD.

In the MBSE approach, integrating views from different disciplines into models naturally forces focus on the system and components boundaries. In this chapter, we will present the SCADE System tool and, more specifically, how SCADE System supports this focus on interfaces. From the primary architecture model that directs the MBSE approach, SCADE System initiates the management and the export of key data for the ICDs.

Another key advantage of MBSE is simulation.

Simulation is much more than animating a model. Simulation is a very efficient means for early validation of concepts, and securing and optimizing the further development of “physical” components.

We introduce the Simulation Driven Product Development™ concept (SDPD) in the following section.

4.2 Simulation Driven Product Development

“Modeling, simulation, and prototyping used during architecture design can significantly reduce the risk of failure in the finished system […]. These techniques enable the development of complex and costly enabling systems. Systems engineers use modeling and simulation on large complex projects to manage the risk of failure to meet system mission and performance requirements.” [INCOSE]

Organizations are still facing late design failures when they heavily rely on physical integration to verify up to system requirements.

On the opposite, relying on the appropriate modeling and simulation techniques at each stage of the development life cycle allows validation of design decisions assigned to a given stage before going to the next.

As illustrated in Figure 4.3, Systems Architects, Engineering Groups, and Integration and V&V Groups are all requiring appropriate Modeling and Simulation solutions.

SYSTEMS ARCHITECTS:

Functional Simulation and 0D-Simulation are used to verify the feasibility of concepts, to evaluate operations, to aid the understanding of users’ needs, to validate system requirements, and to explore architecture opportunities before detailed components design.

ENGINEERING GROUPS:

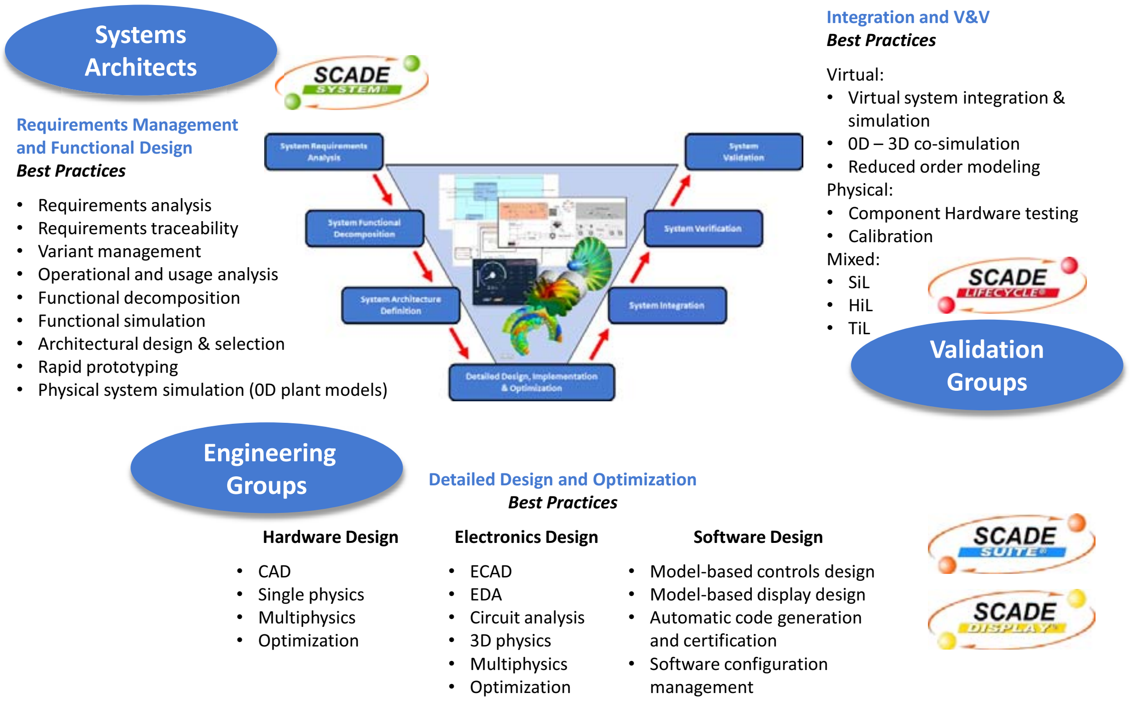
High-fidelity 3D Simulation practices are implemented to obtain design adjustment data and to tune optimization factors before hardware and electronics physical components are designed.

Model-Based Software Development best practices specifically include simulation and code generation. It means that the model is considered here both as an executable spec and as executable code, which is a paramount optimized process.

INTEGRATION AND VERIFICATION & VALIDATION GROUPS:

Various Co-Simulation practices are implemented in order to verify a component in a high-fidelity simulation environment prior to its integration into the system.

- 0D / 3D Co-simulation
- Reduced Order Modeling (ROM)
- Model-in-the-Loop / Software-in-the-Loop / Hardware-in-the-Loop (MIL / SIL / HIL)



Requirements Management and Functional Design
Best Practices

- Requirements analysis
- Requirements traceability
- Variant management
- Operational and usage analysis
- Functional decomposition
- Functional simulation
- Architectural design & selection
- Rapid prototyping
- Physical system simulation (0D plant models)

Integration and V&V
Best Practices

Virtual:
- Virtual system integration & simulation
- 0D – 3D co-simulation
- Reduced order modeling

Physical:
- Component Hardware testing
- Calibration

Mixed:
- SiL
- HiL
- TiL

Detailed Design and Optimization
Best Practices

Hardware Design
- CAD
- Single physics
- Multiphysics
- Optimization

Electronics Design
- ECAD
- EDA
- Circuit analysis
- 3D physics
- Multiphysics
- Optimization

Software Design
- Model-based controls design
- Model-based display design
- Automatic code generation and certification
- Software configuration management

Figure 4.3: Engineering practices and owners

Figure 4.4 below shows the positioning of ANSYS and Esterel Technologies tools suites that span the entire range of physics, electronics, and software so users can assess how their design will behave in the real world and can verify their actual design in a high-fidelity virtual world.

Modeling and Simulation foster "what-if thinking" and exploration of design alternatives for optimal solutions at each stage of the product development life cycle.

ANSYS calls this approach: **Simulation Driven Product Development** (SDPD).

SCADE System, together with SCADE LifeCycle Requirements Management Gateway, SCADE LifeCycle Rapid Prototyper, and ANSYS Simplorer are the tools that implement SDPD at the early stages of the Model-Based Systems Engineering process:

- SCADE System is a systems modeling tool that was developed specifically for modeling functional decomposition as well as systems architecture with high dependability requirements managed using SCADE LifeCycle Requirements Management Gateway, and providing full support of industrial systems engineering processes, such as ARP 4754A, ISO 26262 and EN 50126.

- SCADE LifeCycle Rapid Prototyper enables fast modeling of human machine interfaces, thus facilitates the elicitation of the system operations.
- ANSYS Simplorer, utilizing multi-domain 0D simulation techniques, enables analysis of many aspects of systems, from electronic components to electromechanical components and mechatronics.

Prior to implementation, modeling of mechatronics components provides the ability to simulate and analyze inner structures and dynamics, as well as impact of deformations, stresses, fluids, thermal flows, and high- and low-frequency electromagnetic:

- ANSYS Mechanical offers simulation solutions for the analysis of linear and nonlinear structures, while simulation of motion and stability of mechanical systems is handled by ANSYS Rigid Body Dynamics add-on.
- ANSYS Maxwell allows for analyzing electromechanical devices while ANSY HFSS can simulate full-wave electromagnetic fields.
- ANSYS Fluent gives the capability to model fluid dynamics including turbulences and heat transfers.
- Apache Design Solutions enable engineers to design and simulate low-power high-performance integrated circuits.
- SCADE Suite and SCADE Display enable engineers to model, simulate, and generate certified code for embedded software, respectively embedded control software and human machine interfaces.

While physical implementations and verifications of electronic and hardware components are processed, their initial 0D and/or 3D models can be updated accordingly in order to allow virtual simulation verifying late design choices.

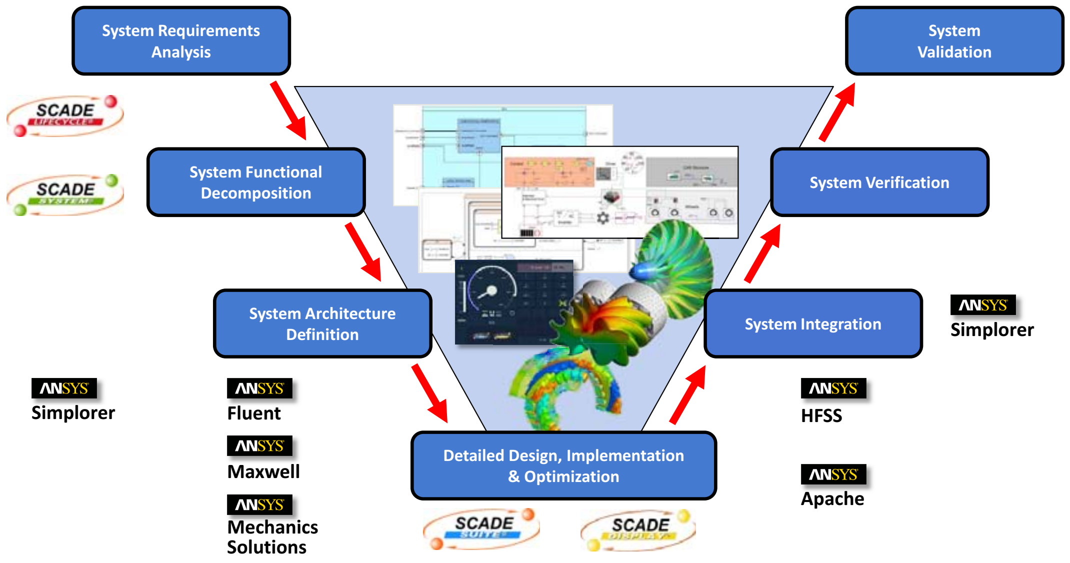


Figure 4.4: ANSYS / Esterel Technologies Systems Engineering solution map

ANSYS Simplorer uses the IEEE standardized language VHDL-AMS to develop plant models, as well as the capability to integrate Reduced Order Models from 3D physics based solvers for higher fidelity. These plant models can be used in verifying software applications generated from SCADE Suite and SCADE Display prior to their integration with the physical components. Thus, Integration, which is usually a tense and hazardous phase, can be very progressive (Figure 4.5 below).

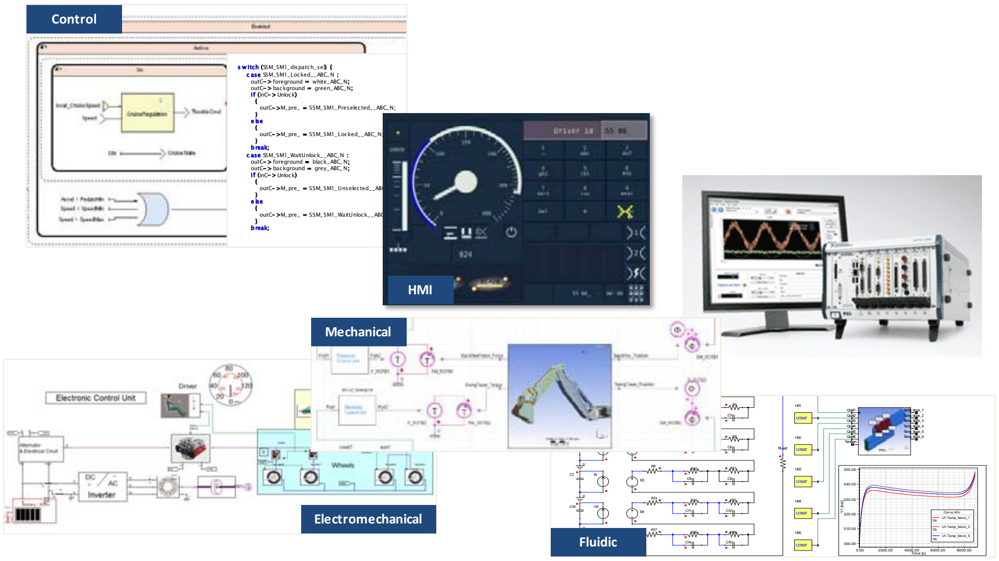


Figure 4.5: Virtual system integration and V&V with ANSYS tools

The following section is handling the major features of SCADE System, which is at the origin of the SDPD approach, like an architect is at the origin of a construction project.

In addition, Appendix D briefly describes SCADE Suite, SCADE Display, SCADE LifeCycle, and ANSYS Simplorer.

4.3 SCADE System Features Overview

SCADE System is a system design tool suite for modeling safety-critical and mission-critical systems that is based on the SysML and Eclipse standards.

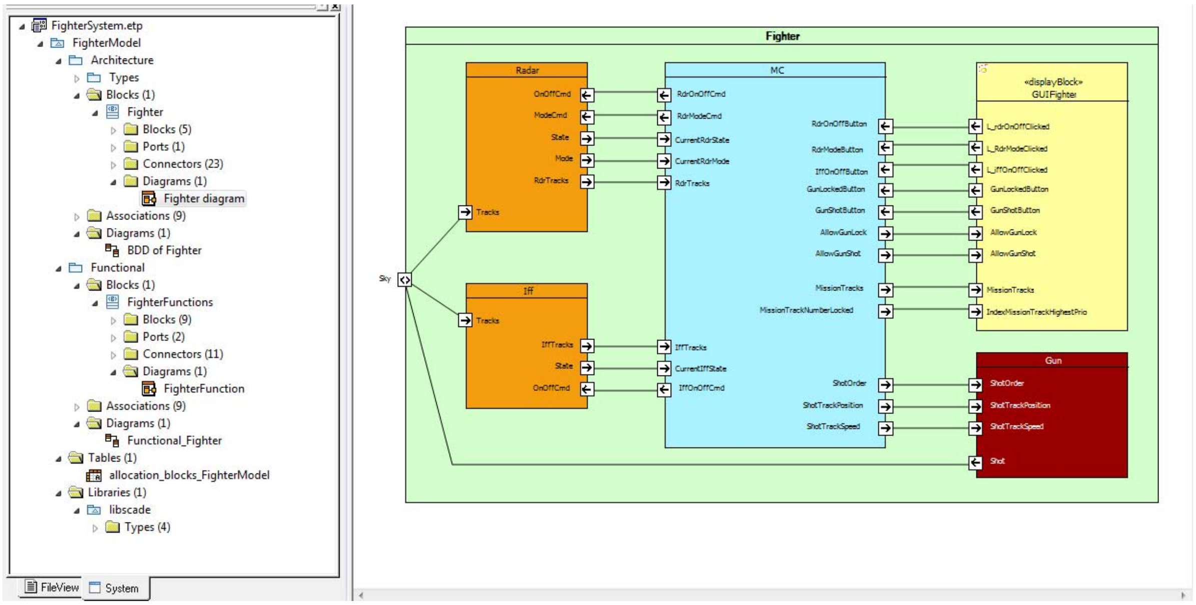


Figure 4.6: Designing with SCADE System

SCADE System features structural system modeling, data dictionary, allocation tables, and architecture verification.

It allows critical systems designers to reduce duplication of efforts and to avoid inconsistencies between system architectural descriptions and software/hardware/mechanical descriptions. It allows, as well, systems engineers to safely manage large amount of data related to naming, interfaces, and all artifacts that should be shared among the whole team.

Figure 4.6 shows the SCADE System Graphical User Interface.

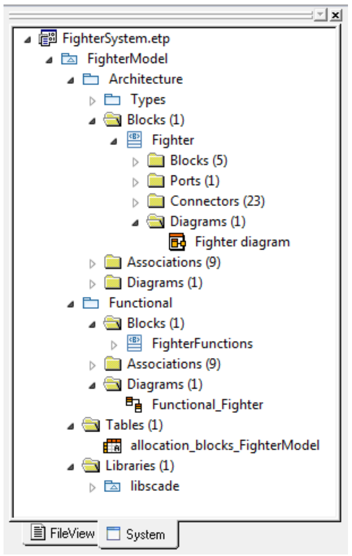
4.3.1 Model and Package

In SCADE System, a model is a graphically represented tree of packages of blocks that constitutes a system (Figure 4.7 below).

The model gathers:

- views of interconnected blocks (typically used to define functional view and architectural view)
- the data dictionaries (data dictionaries are gathering additional data, for example refining information on the interfaces of the blocks)
- the allocation tables (allocation tables are expressing the relationship between blocks or data)
- the Libraries

Packages are structuring the different views.



```
FighterSystem.etp
  FighterModel
    Architecture
      Types
      Blocks (1)
        Fighter
          Blocks (5)
          Ports (1)
          Connectors (23)
          Diagrams (1)
            Fighter diagram
      Associations (9)
      Diagrams (1)
    Functional
      Blocks (1)
        FighterFunctions
      Associations (9)
      Diagrams (1)
        Functional_Fighter
    Tables (1)
      allocation_blocks_FighterModel
    Libraries (1)
      libscade
```

Figure 4.7: Model and package in SCADE System

4.3.2 Block Diagram

A Block Diagram (Figure 4.8 below) is a basic graphical construct of the model. A block has ports. Ports are input or/and output gates of the block. Connectors connect the different blocks through the ports. Blocks, ports, and connectors enable the representation of functional breakdown of the system, as well as its architectural construction.

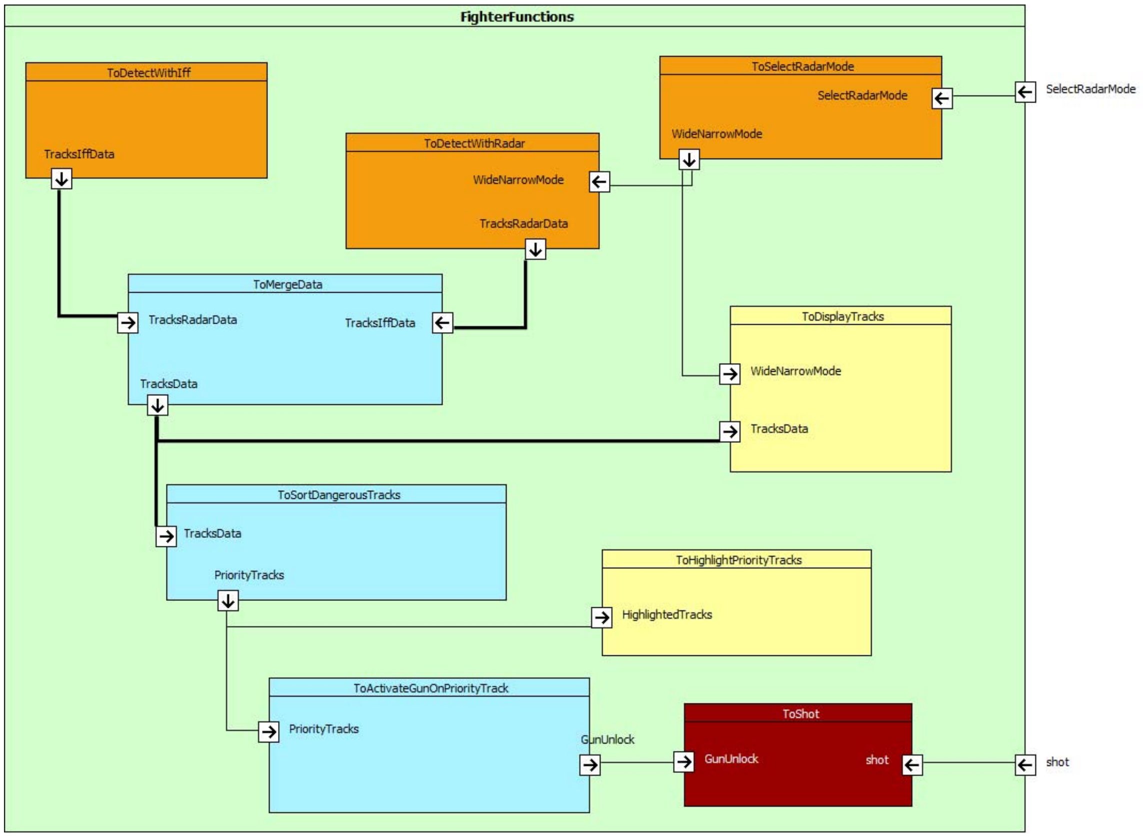
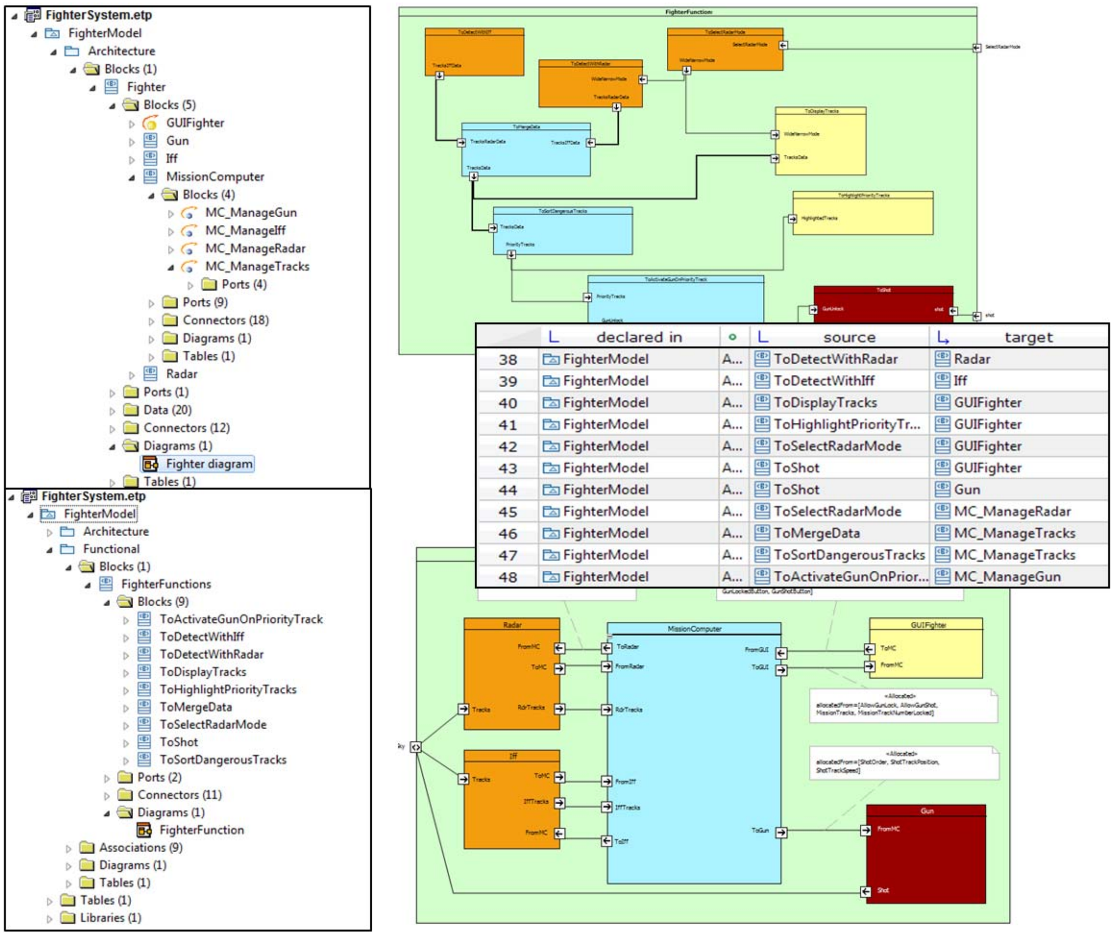


Figure 4.8: Block Diagram in SCADE System

4.3.3 Allocation Tables

Using simple drag & drop actions, Allocation Tables (Figure 4.9 below) link objects of a package to objects of another package.

For example, functional blocks can be allocated to component architectural blocks, and same for the corresponding interfaces (Ports).



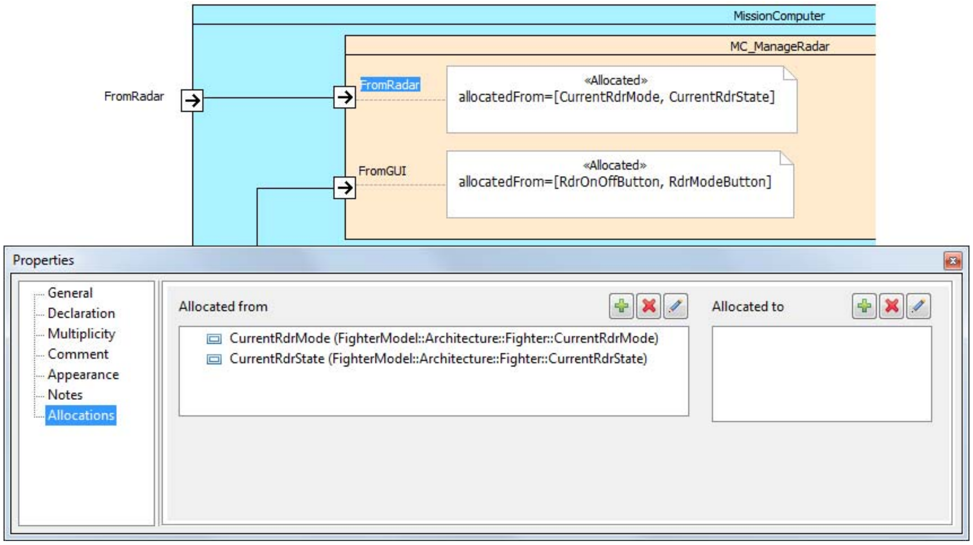
```
FighterSystem.etp
  FighterModel
    Architecture
      Blocks (1)
        Fighter
          Blocks (5)
            GUIFighter
            Gun
            Iff
            MissionComputer
              Blocks (4)
                MC_ManageGun
                MC_ManageIff
                MC_ManageRadar
                MC_ManageTracks
                  Ports (4)
              Ports (9)
              Connectors (18)
              Diagrams (1)
              Tables (1)
            Radar
          Ports (1)
          Data (20)
          Connectors (12)
          Diagrams (1)
            Fighter diagram
          Tables (1)
```

```
FighterSystem.etp
  FighterModel
    Architecture
    Functional
      Blocks (1)
        FighterFunctions
          Blocks (9)
            ToActivateGunOnPriorityTrack
            ToDetectWithIff
            ToDetectWithRadar
            ToDisplayTracks
            ToHighlightPriorityTracks
            ToMergeData
            ToSelectRadarMode
            ToShot
            ToSortDangerousTracks
          Ports (2)
          Connectors (11)
          Diagrams (1)
            FighterFunction
      Associations (9)
      Diagrams (1)
      Tables (1)
    Tables (1)
    Libraries (1)
```

	declared in		source	target
38	FighterModel	A...	ToDetectWithRadar	Radar
39	FighterModel	A...	ToDetectWithIff	Iff
40	FighterModel	A...	ToDisplayTracks	GUIFighter
41	FighterModel	A...	ToHighlightPriorityTr...	GUIFighter
42	FighterModel	A...	ToSelectRadarMode	GUIFighter
43	FighterModel	A...	ToShot	GUIFighter
44	FighterModel	A...	ToShot	Gun
45	FighterModel	A...	ToSelectRadarMode	MC_ManageRadar
46	FighterModel	A...	ToMergeData	MC_ManageTracks
47	FighterModel	A...	ToSortDangerousTracks	MC_ManageTracks
48	FighterModel	A...	ToActivateGunOnPrior...	MC_ManageGun

Figure 4.9: Allocation tables in SCADE System

For any SCADE System object, the list of objects it is allocated to and the list of objects allocated to it ("Allocated from") can also be displayed and edited from the object Properties view.



«Allocated»
allocatedFrom=[CurrentRdrMode, CurrentRdrState]

«Allocated»
allocatedFrom=[RdrOnOffButton, RdrModeButton]

CurrentRdrMode (FighterModel::Architecture::Fighter::CurrentRdrMode)
CurrentRdrState (FighterModel::Architecture::Fighter::CurrentRdrState)

Figure 4.10: Allocation in Properties

4.3.4 Data Dictionary

Tables of data are usual means used by systems engineers to represent systems artifacts. In particular, the definition of interfaces in ICDs necessitates a large amount of data to be established, distributed among teams, and maintained.

Data created in a SCADE System project can be particularized using predefined properties and structured annotations.

These properties and annotations can be viewed in tables.

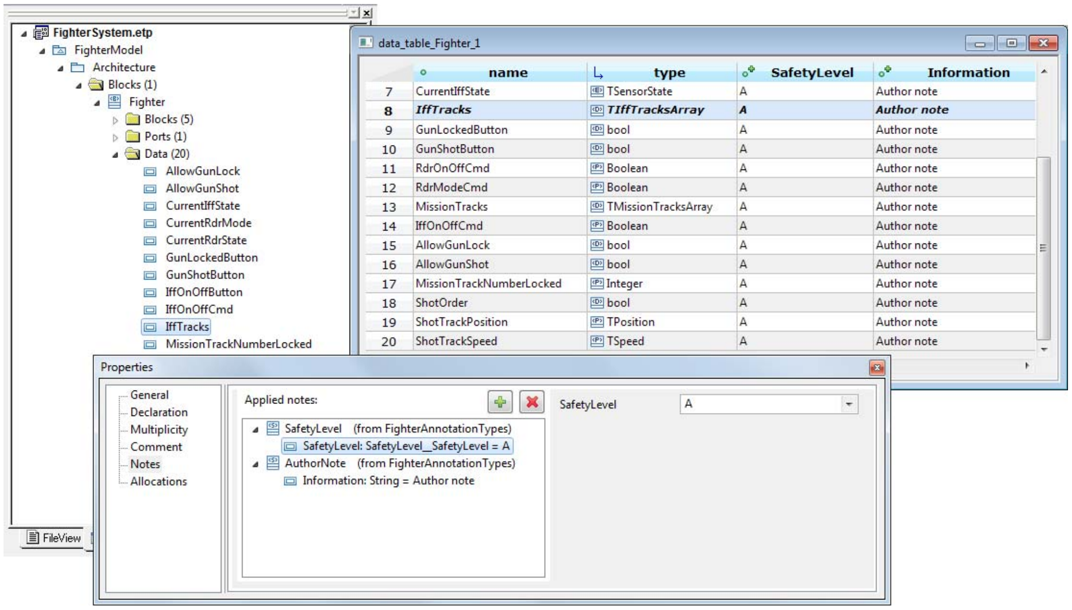
The tables, as a whole, make up the Data Dictionary, in SCADE System.

The support of data dictionaries in SCADE System features:

- Import and export of data dictionaries (copy/paste from/to Excel spreadsheets)
- Link data with the functional and architecture models
- Data management is benefiting from the whole SCADE System engineering capabilities: graphical views, traceability, diff, report, automated verifications, etc.

As shown in Figure 4.11 below:

- SCADE System Data Dictionary is implemented as a block viewed in the model tree
- Data are gathered from structured information via the annotation mechanism
- Data properties can be shown and edited in customizable Tables



```
FighterSystem.etp
  FighterModel
    Architecture
      Blocks (1)
        Fighter
          Blocks (5)
          Ports (1)
          Data (20)
            AllowGunLock
            AllowGunShot
            CurrentIffState
            CurrentRdrMode
            CurrentRdrState
            GunLockedButton
            GunShotButton
            IffOnOffButton
            IffOnOffCmd
            IffTracks
            MissionTrackNumberLocked
```

data_table_Fighter_1

	name	type	SafetyLevel	Information
7	CurrentIffState	TSensorState	A	Author note
8	***IffTracks***	***TIffTracksArray***	***A***	***Author note***
9	GunLockedButton	bool	A	Author note
10	GunShotButton	bool	A	Author note
11	RdrOnOffCmd	Boolean	A	Author note
12	RdrModeCmd	Boolean	A	Author note
13	MissionTracks	TMissionTracksArray	A	Author note
14	IffOnOffCmd	Boolean	A	Author note
15	AllowGunLock	bool	A	Author note
16	AllowGunShot	bool	A	Author note
17	MissionTrackNumberLocked	Integer	A	Author note
18	ShotOrder	bool	A	Author note
19	ShotTrackPosition	TPosition	A	Author note
20	ShotTrackSpeed	TSpeed	A	Author note

Properties

- General
- Declaration
- Multiplicity
- Comment
- Notes
- Allocations

Applied notes:

```
SafetyLevel    (from FighterAnnotationTypes)
  SafetyLevel: SafetyLevel_SafetyLevel = A
AuthorNote    (from FighterAnnotationTypes)
  Information: String = Author note
```

SafetyLevel: A

Figure 4.11: Data dictionary in SCADE System

As shown in Figure 4.12 below, data allocations can also be graphically displayed in diagrams.

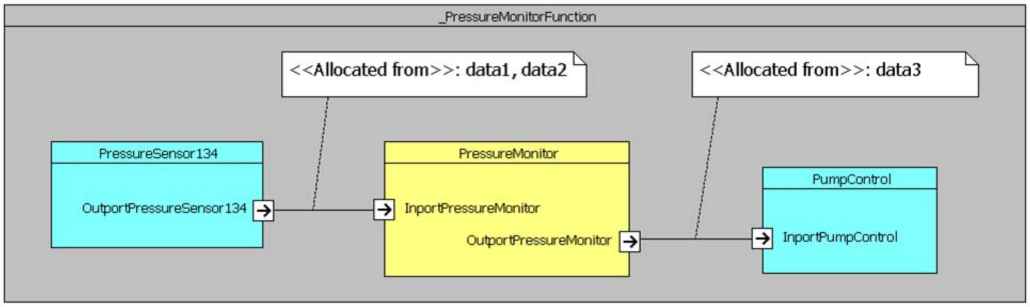
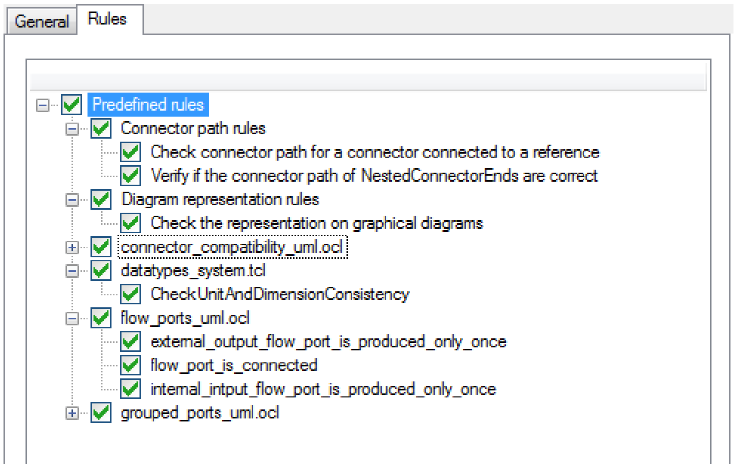


Figure 4.12: Allocation of additional data to connections between blocks

4.3.5 Model Static Checking

SCADE System Checker is a tool which checks models for consistency and for other project-dependent constraints.

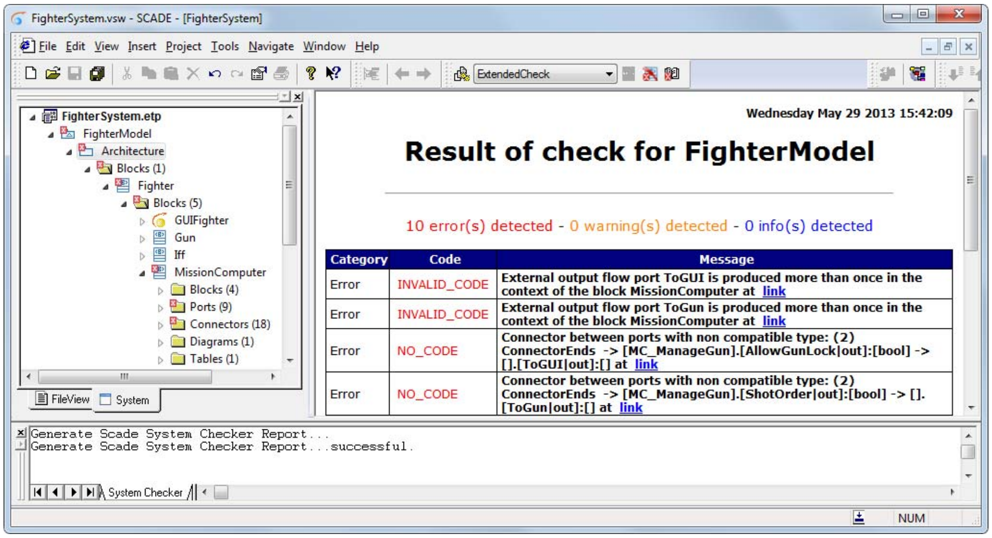
Checks are processed with a set of selectable rules (predefined rules or user-defined rules). The selection is defined in configurations allowing for distinct checks. These checks can target specific parts of the model. For example, some checks apply to the Functional Decomposition only and some other checks apply to the Architecture Definition only.



- ✓ Predefined rules
  - ✓ Connector path rules
    - ✓ Check connector path for a connector connected to a reference
    - ✓ Verify if the connector path of NestedConnectorEnds are correct
  - ✓ Diagram representation rules
    - ✓ Check the representation on graphical diagrams
  - ✓ connector_compatibility_uml.ocl
  - ✓ datatypes_system.tcl
    - ✓ CheckUnitAndDimensionConsistency
  - ✓ flow_ports_uml.ocl
    - ✓ external_output_flow_port_is_produced_only_once
    - ✓ flow_port_is_connected
    - ✓ internal_intput_flow_port_is_produced_only_once
  - ✓ grouped_ports_uml.ocl

Figure 4.13: List of rules available for static checking

SCADE System Checker reports as shown in Figure 4.14.



Wednesday May 29 2013 15:42:09

Result of check for FighterModel

10 error(s) detected - 0 warning(s) detected - 0 info(s) detected

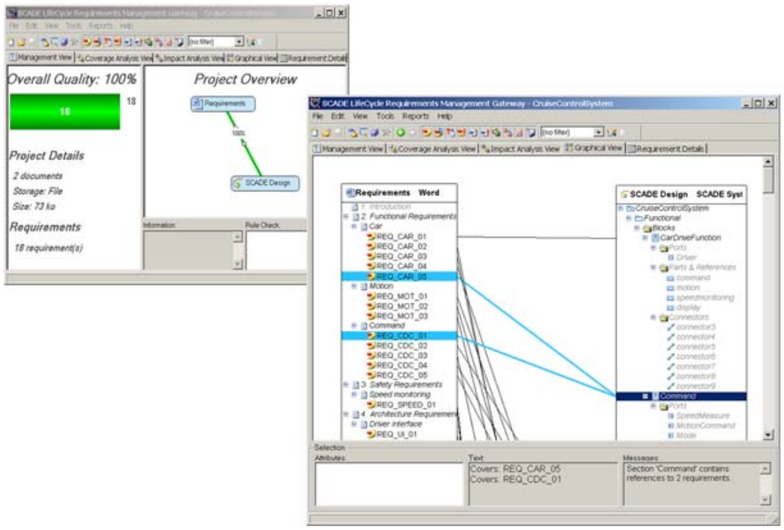
Category	Code	Message
Error	INVALID_CODE	External output flow port ToGUI is produced more than once in the context of the block MissionComputer at link
Error	INVALID_CODE	External output flow port ToGun is produced more than once in the context of the block MissionComputer at link
Error	NO_CODE	Connector between ports with non compatible type: (2) ConnectorEnds -> [MC_ManageGun].[AllowGunLock|out]:[bool] -> [].[ToGUI|out]:[] at link
Error	NO_CODE	Connector between ports with non compatible type: (2) ConnectorEnds -> [MC_ManageGun].[ShotOrder|out]:[bool] -> [].[ToGun|out]:[] at link

```
Generate Scade System Checker Report...
Generate Scade System Checker Report...successful.
```

Figure 4.14: SCADE System Checker

4.3.6 Traceability Management

SCADE LifeCycle Requirements Management Gateway is accessible via SCADE System in order, for example, to allocate requirements to functions and component architectural blocks (Figure 4.15).

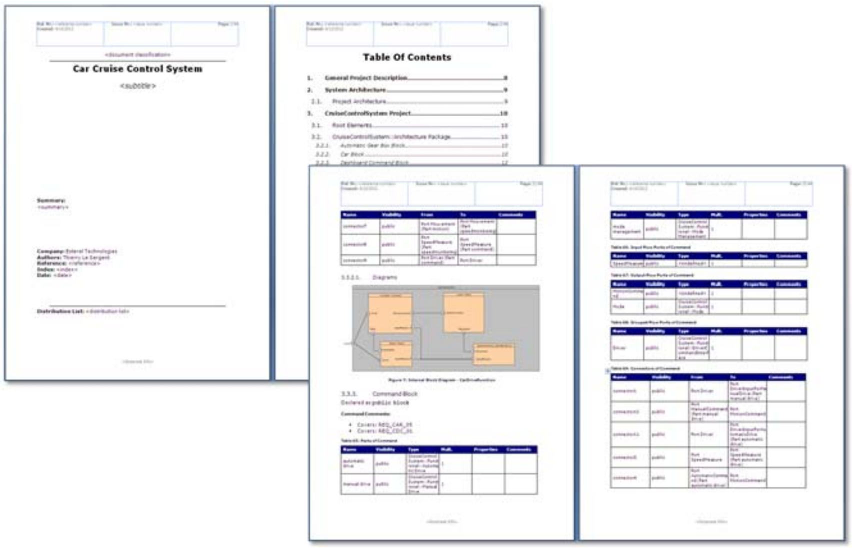


Selection
Attributes
Text
Covers: REQ_CAR_05
Covers: REQ_CDC_01
Messages
Section 'Command' contains references to 2 requirements

Figure 4.15: Traceability management in SCADE System

4.3.7 Model Documentation Generation

SCADE LifeCycle Reporter is accessible from SCADE System to generate complete documentation of the model (Figure 4.16).


Car Cruise Control System

<subtitle>

Summary:

Table Of Contents

1. General Project Description 8
2. System Architecture 9
2.1. Project Architecture 9
3. CruiseControlSystem Project 10
3.1. Root Elements 10
3.2. CruiseControlSystem::Architecture Package 10
3.2.1. Automatic Gear Box Block 10
3.2.2. Car Block 10
3.2.3. Dashboard Command Block 12

Figure 4.16: Complete documentation generation in SCADE System

4.3.8 System-Software Synchronization

Each block of a SCADE System model can be assigned to a specific implementation.

For example, blocks that represent a piece of software can be tagged as being implemented using SCADE Suite.

SCADE System allows for transforming a block interface into inputs/outputs of the corresponding software operator to be implemented in SCADE Suite.

When modifying either side, it is possible to resynchronize the corresponding tagged SCADE System blocks with the SCADE Suite operators. (Figure 4.17).

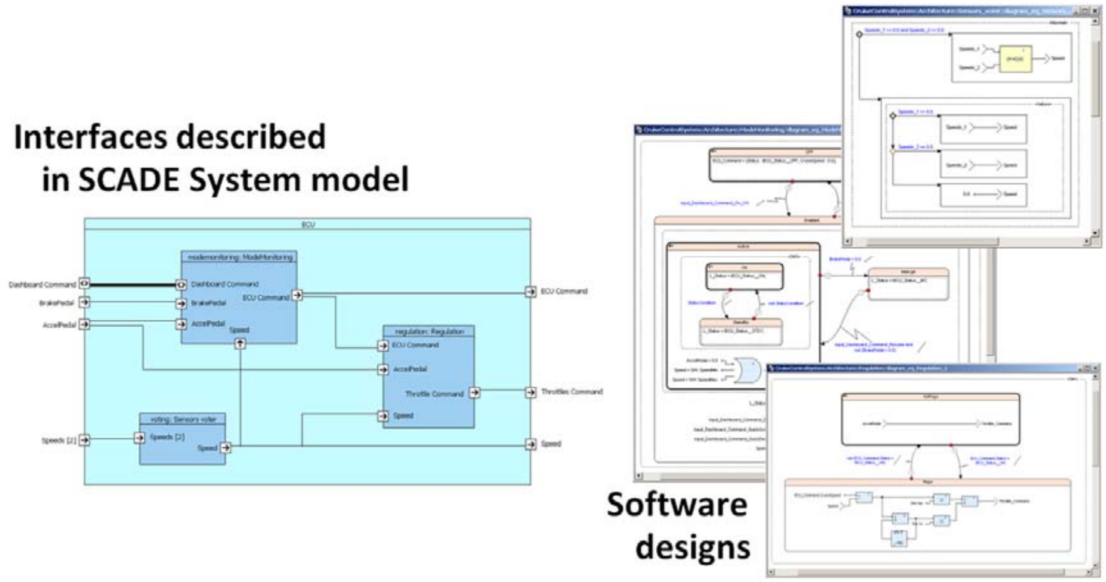


Figure 4.17: Synchronization with SCADE Suite

5. SCADE Model-Based Systems Engineering Process

5.1 SCADE Model-Based Systems Engineering V-Cycle

Figure 5.1 below represents the Model-Based Systems Engineering (MBSE) V-Cycle. This V-cycle is compliant with ARP-4754A and demonstrates the SDPD approach. In this SCADE MBSE V-cycle, one can distinguish twelve activities.

Three activities are supporting the process of establishing the System Requirements:

- Analysis of the Requirements
- Decomposition of the Functions
- Validation of the Requirements

Four activities are assigned to the establishment of the Architecture artifacts:

- Definition of the Architecture
- Validation of the Architecture
- Allocation of Functions to Items
- Allocation of Requirements to Items

The two following activities are dedicated to the implementation of Items:

- Models of the Items are elaborated and simulated (can be: software models, hardware electronics 0D models, hardware mechanics 3D models, etc.)
- Then, the Items are implemented according to their respective certification standard; for example:
  - Software items compliant with the DO-178B/C standard
  - Electronic hardware items compliant with the DO-254 standard

It is important to notice that, when developed with SCADE Suite or SCADE Display, models of the software items also embody the implementations: SCADE Suite and SCADE Display feature DO-178B/C qualified Code Generators that make the generated code a correct implementation of the model behavior.

The last three activities are about:

- Integrating
- Verifying
- Validating both the Items and the built System

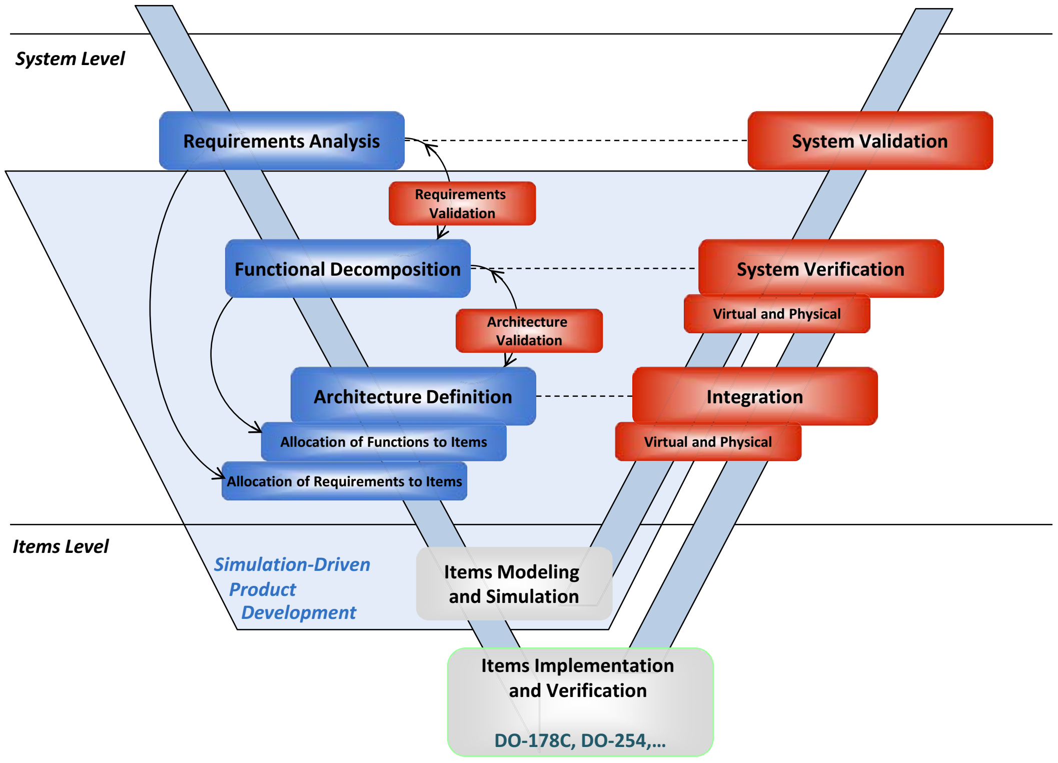


Figure 5.1: ARP-4754A compliant SCADE MBSE V-Cycle (System Level)

5.2 Requirements Analysis

Since the purpose of Systems Engineering process is to transform Requirements into Designs, Requirements are the primary and the most important focus of the Systems Engineers.

Requirements Analysis is the act of converting the stakeholders’ needs into a set of validated Requirements that will be managed with all the interdisciplinary teams.

The purpose of Requirements Analysis is to refine users’ needs and objectives, define expected performances, and identify constraints and standards that are limiting the solutions.

Achieving the goal of the Requirements Analysis, while all is coming from dissimilar ideas and diverse needs, is not an easy task.

Process and iterations are tightly required.

Requirements Analysis is conducted iteratively with Functional Decomposition, Architectural Design and Safety Analysis, and benefits from them.

However, the first goal of the process is to establish an initial set of System Requirements, called Elicitation of the Requirements, on which, and from which, the other activities are performed.

5.2.1 Requirements Elicitation

Requirements Elicitation is a complex process that basically intends to go from a collection of incomplete, unstructured, and possibly inconsistent needs to an organized knowledge base, as illustrated in Figure 5.2 below.

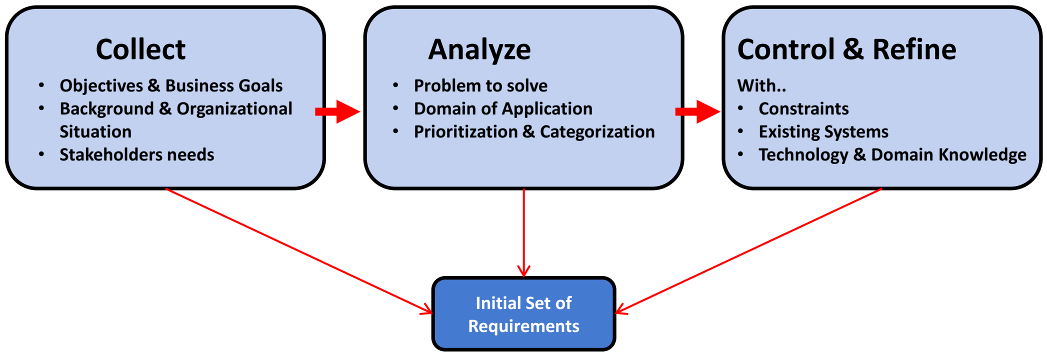

Collect
• Objectives & Business Goals
• Background & Organizational Situation
• Stakeholders needs
Analyze
• Problem to solve
• Domain of Application
• Prioritization & Categorization
Control & Refine
With..
• Constraints
• Existing Systems
• Technology & Domain Knowledge
Initial Set of Requirements

Figure 5.2: Requirements elicitation process

One primary and very important accomplishment consists in identifying the complete range of stakeholders:

- Who has an interest in the system throughout its entire life cycle?
- Have all the stakeholders expressed their needs?

The stakeholders of a project developing a given system are those that have a legitimate interest in this system. They are:

- Users (needing services delivered by the system)
- Operators (interacting with the system)
- Decision-makers (giving system objectives)
- Regulatory bodies (constraining the system)

The stakeholders give their "needs" to the project:

- Users: preliminary requirements (*a.k.a.*, parent requirements), etc.
- Operators: use cases, etc.
- Decision-makers: business case, budgets, schedule, organization, reuse policy, etc.
- Regulatory bodies: safety case, industry standards, applicable laws, etc.

Other "inputs" should be considered:

- Procedures, standards, guidelines that the project should conform to
- State-of the-art technology, technology breakthroughs,

Once all of the previous aspects are considered, it is possible to establish the preliminary requirements: what the system must accomplish and how well?

An initial classification of the System Requirements is made according to:

- Functional requirements
- Safety requirements
- Certification requirements
- Constraints requirements
- Derived requirements

Functional Requirements are representing the services that the system should provide. They encompass the performance requirements and the interface requirements.

Safety requirements are driving the requirements for safety prevention such as redundancies or physical partitioning.

The ARP-4754 process is meant to structure these inputs into the Requirements Analysis process.

The impact of Certification requirements is focused on functional and architectural dissimilarities of designs.

Constraint requirements (*a.k.a.,* Assumptions) are dealing with functional and architectural variants, maintenance, built-in-testing, installation, resource allocation, boundaries conditions, environmental assumptions, reuse constraints, manufacturing conditions.

For systems that are mostly implemented by software components, the FAA states:

"*Every system makes specific assumptions about the environment in which it will operate. Some of these assumptions are nothing more than the types, ranges, and units of the inputs it will accept and the outputs it will produce. Often, correct behavior of the system is dependent on more complex assumptions about its environment. These are actually requirements levied by the system on its environment. Identification of a system's environmental assumptions is essential for maintenance and to enable reuse. Failure to identify the environmental assumptions and the subsequent misuse of the system is a common cause of system failure.*" [DOT/FAA/AR-08/32]

Derived requirements are those that are added during and by the Systems Engineering process. These requirements do not change the requirements coming from stakeholders' needs. They usually refer to technology, design, methods, and tools requisites.

Another primary goal of this initial classification is to extract the leading requirements which are the subject of the next mini-step: Identification of the operations (generally speaking, these leading requirements are mostly functional requirements, but other breakthrough requirements can be considered for the identification of the Operations).

5.2.2 Operational Analysis

Operational Analysis is a **Requirement Elicitation** method, especially performed (but not only) when the system is developed from a concept (*i.e.*, "*from scratch*").

This activity is also known as Concept Analysis or Concept of Operations (CONOPS).

"*A concept of operations is a document describing the characteristics of a proposed system from the viewpoint of an individual who will use that system. It is used to communicate the quantitative and qualitative system characteristics to all stakeholders. CONOPS are widely used in the military, governmental services and other fields.*

A CONOPS generally evolves from a concept and is a description of how a set of capabilities may be employed to achieve desired objectives or end state." (Source: Wikipedia)

Operational Analysis primarily describes the necessary elements of the context and the system as illustrated in Figure 5.3 below [ANSI/AIAA].

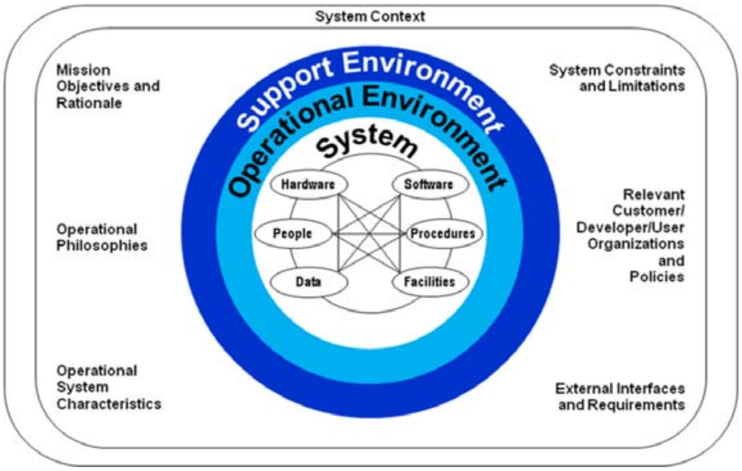

System Context
Mission Objectives and Rationale
Operational Philosophies
Operational System Characteristics
System Constraints and Limitations
Relevant Customer/ Developer/User Organizations and Policies
External Interfaces and Requirements
Support Environment
Operational Environment
System
Hardware
Software
People
Procedures
Data
Facilities

Figure 5.3: Operational Analysis

This activity starts with the identification of the System boundary.

For systems that are mostly implemented by software components the FAA recommends the following practice:

"*Recommended Practice 2.2.1: Define the system boundary early in the requirements engineering process by identifying a preliminary set of monitored and controlled variables. […] The monitored variables represent quantities in the environment that the system responds to, while the controlled variables represent quantities in the environment that the system will affect. For example, monitored values might be the actual altitude of an aircraft and its airspeed, while controlled variables might be the position of a control surface, such as an aileron or the displayed value of the altitude on the primary flight display.*" [DOT/FAA/AR-08/32]

To help identify System Context, including the users and operators of the system, SCADE System offers the capability to model a block diagram with Actors. This top-level model is structuring the placing of the system in its environment where it interacts with its users and other actors (Figure 5.4)

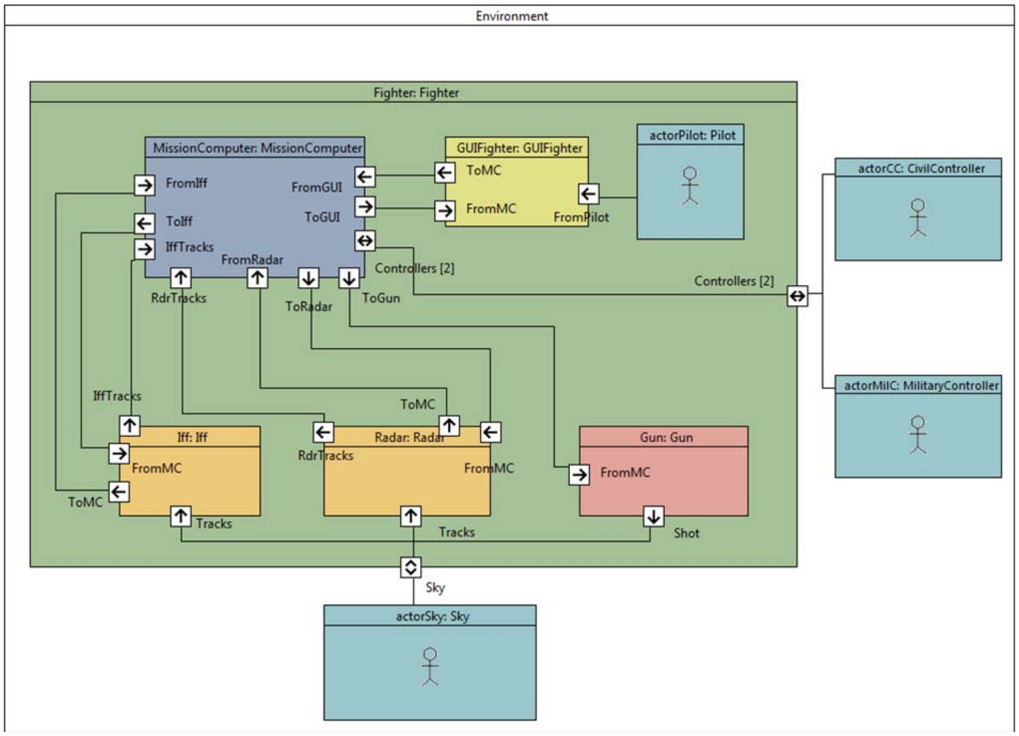


Figure 5.4: Top-level model in SCADE System

Relying on Context Analysis, Concept Analysis then describes the missions (*a.k.a.*, scenarios of use) of the system in its operational environment as an integrated view that ensures all the subsequent requirements are identified and are coherent (individual functions must meet their specification and the whole must meet the users' needs).

Graphical techniques and tools can be used to best describe the users' perspectives and operations: story boards, state diagrams, use cases or sequence and activity charts, and Prototyping.

5.2.3 Prototyping the Operations

As far as aircraft systems made of electronic hardware and software are concerned, prototyping means Virtual Prototyping, which is a combination of Modeling and Simulation techniques.

Important Note

Prototyping is both an Elicitation means and a Validation means (see Section 5.3). Here, it is about prototyping the operations, while, for validation purpose, the functions are prototyped.

Operation is also known as Mission or Scenario of Use.

Figure 5.5 below shows a Fighter Mission System using a Radar and an IFF (Identification Friend or Foe). On the bottom left, a panel of graphical widgets that allows creating scenarios. On the top right, the system information displayed to the pilot.

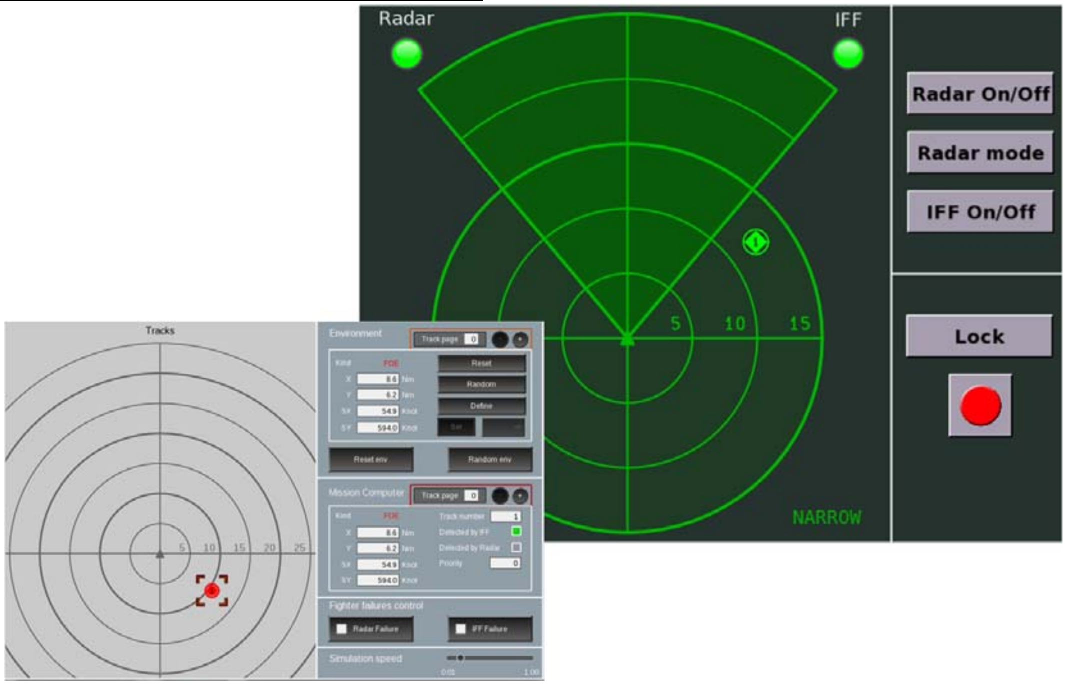


Figure 5.5: Prototyping operations

SCADE LifeCycle Rapid Prototyper includes a wide collection of widgets. It enables the construction of a virtual prototype, like the above Fighter Mission System.

5.2.4 A Focus on Derived Requirements

"*At each phase of the development activity, decisions are made as to how particular requirements or groups of requirements are to be met. The consequences of these design choices become requirements for the next phase of the development. Since these requirements result from the design process itself, they may not be uniquely related to a higher-level requirement and are referred to as derived requirements.*" [ARP-4754A]

Architecture artifacts are coming from decisions made during the System design process. The Architecture requirements are passed to the teams in charge of the next phase: the Implementation phase.

This practice, *i.e.*, creating Derived Requirements, is somehow banished at Item development level, when governed, for example, by DO-178B/C objectives: derived requirements can be identified during the process, but the usual practice is to create additional high-level requirements that allow canceling them.

As a consequence, to avoid confusion, one calls "*Requirements from System Design*", the Requirements that are made when decomposing the functions with SCADE System and that are better expressed by a model than by a text.

Using SCADE LifeCycle Requirements Management Gateway, it is possible to declare SCADE System model elements as requirements by means of dedicated comments.

5.3 Functional Decomposition

The objective of this process is two-fold:

- Define elementary functions that are able to be allocated to items
- Define a functional model of the system that is able to contribute to the validation of the requirements

It allows focusing on what has to be achieved, not how. When doing so, more solutions may be systematically explored (Figure below).

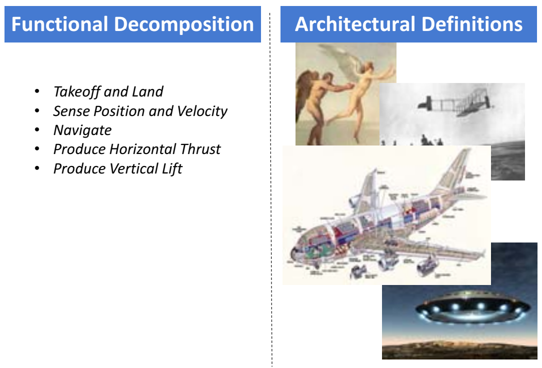

Functional Decomposition
Architectural Definitions
Takeoff and Land
Sense Position and Velocity
Navigate
Produce Horizontal Thrust
Produce Vertical Lift

Figure 5.6: Functional Decomposition versus Architectural Definitions

The purpose of the process is to model a breakdown of the functions from the actual set of requirements. Interfaces are identified and modeled as well. Each function is expressed as an action, starting with a verb, for example: "Control the flight".

The decomposition of a top-level function stops when the actual “leaf” function is supposed to fit the architectural definition (it means this supposition is confirmed only when Functional Decomposition, Architecture Design, and Allocation of Functions to Items are completed).

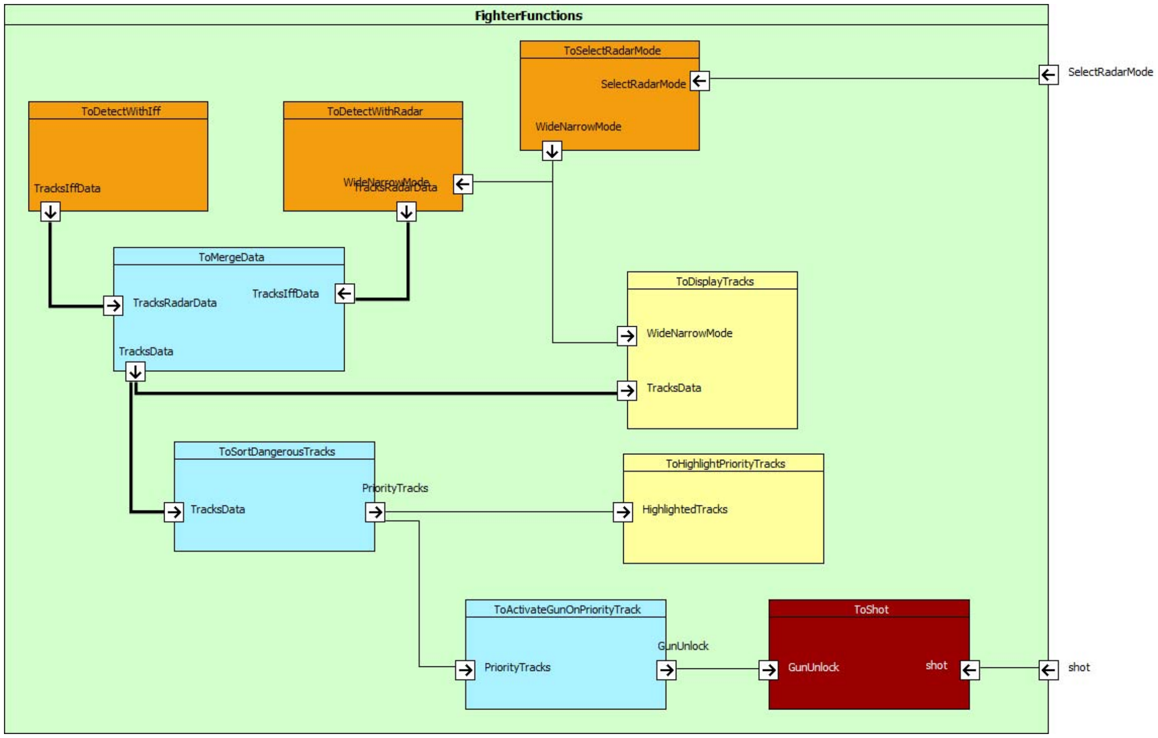
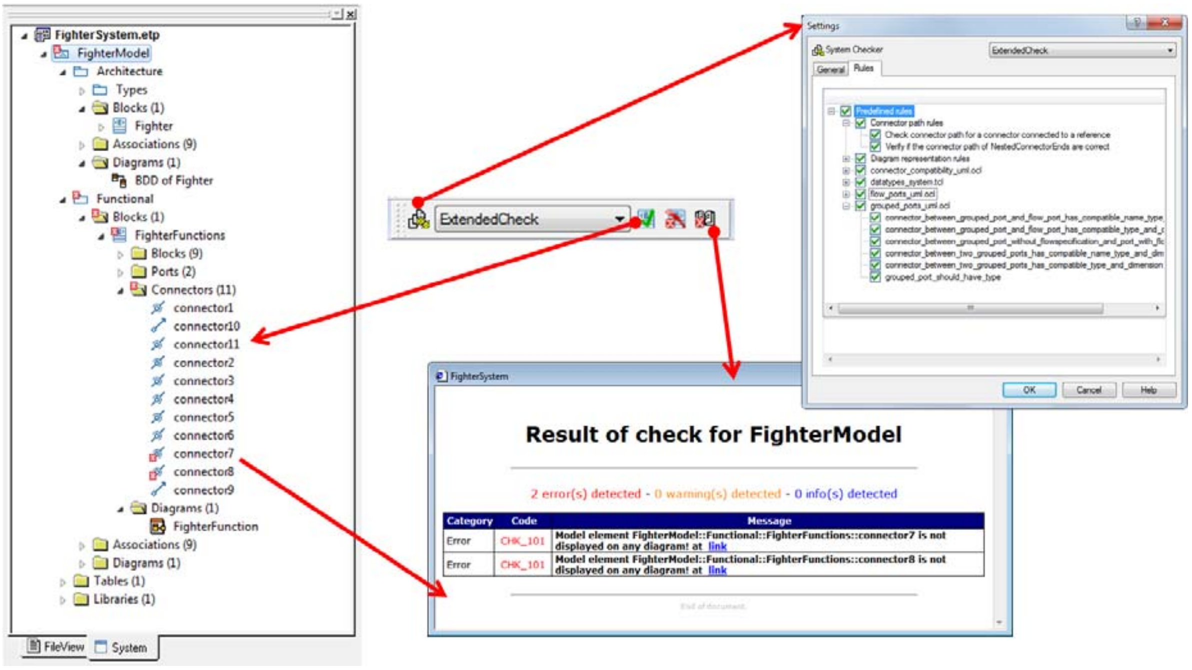


Figure 5.7: Functional decomposition of an aircraft system

Each Function must be verified regarding:

- Naming conventions
- Description and attributes
- Uniqueness
- Usefulness (within the functional decomposition)

SCADE System Checker can check the rules that can be automated.



```
FighterSystem.etp
  FighterModel
    Architecture
      Types
      Blocks (1)
        Fighter
      Associations (9)
      Diagrams (1)
        BDD of Fighter
    Functional
      Blocks (1)
        FighterFunctions
          Blocks (9)
          Ports (2)
          Connectors (11)
            connector1
            connector10
            connector11
            connector2
            connector3
            connector4
            connector5
            connector6
            connector7
            connector8
            connector9
          Diagrams (1)
            FighterFunction
      Associations (9)
      Diagrams (1)
    Tables (1)
    Libraries (1)
```

```
Predefined rules
  Connector path rules
    Check connector path for a connector connected to a reference
    Verify if the connector path of NestedConnectorEnds are correct
  Diagram representation rules
  connector_compatibility_uml.ocl
  datatypes_system.tcl
  flow_ports_uml.ocl
  grouped_ports_uml.ocl
    connector_between_grouped_port_and_flow_port_has_compatible_name_type
    connector_between_grouped_port_and_flow_port_has_compatible_type_and_
    connector_between_grouped_port_without_flowspecification_and_port_with_flo
    connector_between_two_grouped_ports_has_compatible_name_type_and_dim
    connector_between_two_grouped_ports_has_compatible_type_and_dimension
    grouped_port_should_have_type
```

Result of check for FighterModel

2 error(s) detected - 0 warning(s) detected - 0 info(s) detected

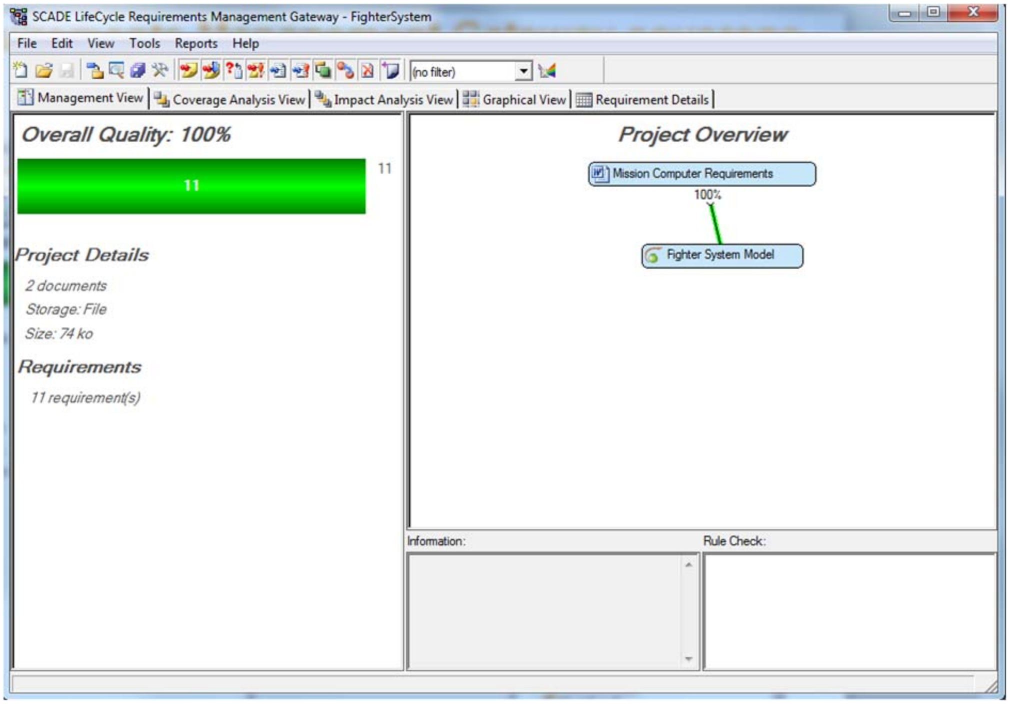
Category	Code	Message
Error	CHK_101	**Model element FighterModel::Functional::FighterFunctions::connector7 is not displayed on any diagram! at link**
Error	CHK_101	**Model element FighterModel::Functional::FighterFunctions::connector8 is not displayed on any diagram! at link**

End of document.

Figure 5.8: SCADE System Checker run on an aircraft system model

The overall Functional Decomposition must be checked as well:

- Consistency
- Readability
- Balance between level of abstraction and level of granularity
- Requirements Coverage (Exhaustiveness)



Overall Quality: 100%

11

Project Details

2 documents

Storage: File

Size: 74 ko

Requirements

11 requirement(s)

Figure 5.9: Requirements Management Gateway coverage measurement

5.4 Requirements Validation

The drivers of this activity are: completeness and correctness.

Usual objectives and means for requirements validation include:

- Validation plan that defines the methods, tools, and schedule implemented to validate the requirements
- Rigor, determined by the FDAL
- The various checklists defined for validation through peer review
- Validation matrix (or other adequate means) to track the status of the Validation process
- Validation summary that gives status of the validation plan and results

5.4.1 Completeness Checks

The completeness of a set of requirements is a demonstration that should be ensured by independent engineering peer review, as it is not an easy task to implement automations or tooling to support this activity.

Checklist is the usual basis for executing such activity, including, but not limited to, the following list of questions (complete list is in section 5.4.4 of the ARP-4754 standard):

- Are all Parent Requirements covered?
- Are all Functions of the Function Decomposition traced back to Requirements?
- Are all scenarios of use (Operations) and of maintenance represented?
- Are all types of requirements represented: Safety, Regulatory, Company standards?
- Are all interfaces to other systems, actors, and processes embodied?
- For a defined use, is the prohibited use defined as well?
- Are Assumptions clearly defined?
- Etc.

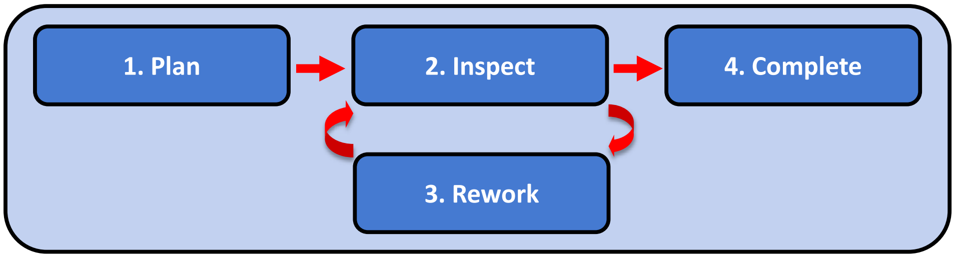


Figure 5.10: Engineering peer review process

5.4.2 Correctness Checks

For Correctness Checks, engineering peer review is an appropriate means as well.

In addition, modeling and simulation can be used to represent the actual understanding of the system functions that are traced to the source requirements.

The checklist is including, but not limited to, the following list of questions (complete list is in section 5.4.3 of the ARP-4754 standard):

- Is the requirement unambiguous?
- Is it clearly stated as a requirement?
- Is it not redundant?
- On the opposite, does it conflict with others?
- Is it feasible to serve it?
- Is it verifiable? (Testable?)
- If it is a derived requirement: is it justified?
- Is it necessary?

5.4.3 Prototyping Functions for Correctness Check

Prototyping is recognized by the ARP-4754A standard as an appropriate means to demonstrate that a requirement is correct.

SCADE Suite is used to prototype:

- Algorithms and control laws
- Control logic and behaviors

SCADE LifeCycle Rapid Prototyper is used to prototype Human-Machine interfaces.

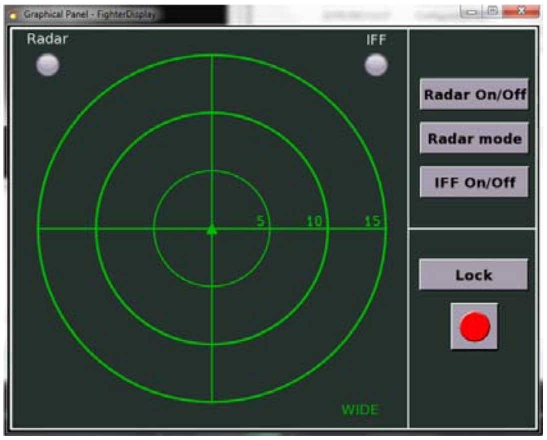


Figure 5.11: Prototyping of functions

ANSYS Simplorer capabilities are used to model and simulate the physical environment in order to perform closed loop simulation.

Co-simulation of these 0D-Simulation tools enables the prototyping and simulation of the functions of the system. A balance between realistic conditions of simulation and time to simulate is made.

At the end, the correctness of the requirements that are traced to the simulated functions is validated in the approximated context of the simulation.

5.4.4 Focus on Assumptions Validation

Assumptions are stated at the same time as requirements. However, Assumptions are statements that cannot be provable while the system is being developed.

Thus, Validation of Assumptions consists of:

- Ensuring that they are explicitly specified as Assumptions and are correctly circulated as is
- Verifying they are "reasonably categorized as Assumptions"
- Justifying all of them

The point is about the term: "*reasonably categorized as Assumptions*". Thus, the usual Assumptions that can be met and accepted are listed below:

- Operational Assumptions (also called: Environmental Assumptions): they are related to air traffic, operational procedures, passengers, etc.
- Design Assumptions related to the crew interface, to interface with other systems, and associated with reliability topics:
  - Example of crew interface assumption: crew response time to a message displayed by the system
  - Example of system interface assumption: probability of bus errors
  - Example of reliability topic that triggers assumption: frequency of maintenance tasks
- Serviceability Assumptions and Installation Assumptions that can be validated against the procedure standards assigned to the project.

5.5 Architecture Definition

"The organization produced through functional analysis is a logical architecture that may not take into account additional constraints, such as the need to satisfy system safety requirements, integrate with legacy systems, or to meet implementation constraints imposed by a particular platform. This practice describes an iterative process that starts from the previously developed functional architecture and leads to an architecture that addresses these concerns. This architecture is then used as the framework for organizing the detailed requirements." [DOT/FAA/AR-08/32]

Architecture definition conveys how to build the system and with what. Thus, at the end of the process, the requirements for the physical artifacts are defined; this is called Physical Architecture.

But, for complex systems and/or when trade-off between several architecture opportunities is both necessary and dense, it is sometimes better to define how to build the system prior to with what to build it. It is called Logical Architecture.

Doing like this, the organization of the items of the system is defined apart from the physical solutions to be established, including reuse and other constraints.

In the SCADE MBSE approach, it is possible to consider the Logical Architecture:

1 as the first of the architecture solutions (at the same level of the others, and from which the others are derived)

2 or as a level of the architecture definition from which all the physical solutions are established

In the second case, a level of traceability linking is added.

In all cases, it is recommended to keep all architecture alternatives in configuration in order:

- to record how the decision is made
- to be able to retrieve the Logical Architecture

In this chapter, we consider Logical Architecture at the same level Physical Architectures are considered.

5.5.1 Architecture Modeling

Objective of the process

- Identify items
- Identify interfaces between items
- Organize items breakdown

Types of item that are considered:

- Software items
- Hardware items (electronics that carry software or operate logical function)
- Electrical items (Actuator, electrical device, motor)
- Mechanical items (gear box, torque shaft)
- Chemical items (battery)
- Hydraulic items (pump, tank, duct, turbo)

Types of interface between items:

- Software to Software
- Software to Hardware

- Hardware to Hardware
- Hardware to Mechanics
- Mechanics to Mechanics

At this stage, a "*Make or Buy*" policy is leading the Systems Engineering decision making: existing company's solutions and/or COTS solutions may be considered.

The constraint requirements govern the implementation solutions: the physical architecture is led by the "Design to Cost" policy and should respect the performance and safety constraint requirements.

Items can be characterized using additional information. SCADE System Annotation capability allows extending the objects of the model with, for example, the following attributes or additional information:

- Attributes related to the physical aspects of the item, such as: weight, volume, boundaries, resources (electrical, hydraulic, pneumatic power, etc.)
- Attributes associated with the operational aspects of the item: tasks assigned to the item, safety specification, complexity, Technology Readiness Level (TRL), etc.
- Performance parameters (useful for future trade-off taking into account stakeholders' criteria)
- Attributes attached to the development process of the item: dependencies between items (some items must be developed before others to enable a correct development plan), cost of development, cost of product, development schedule, quality articles, etc.

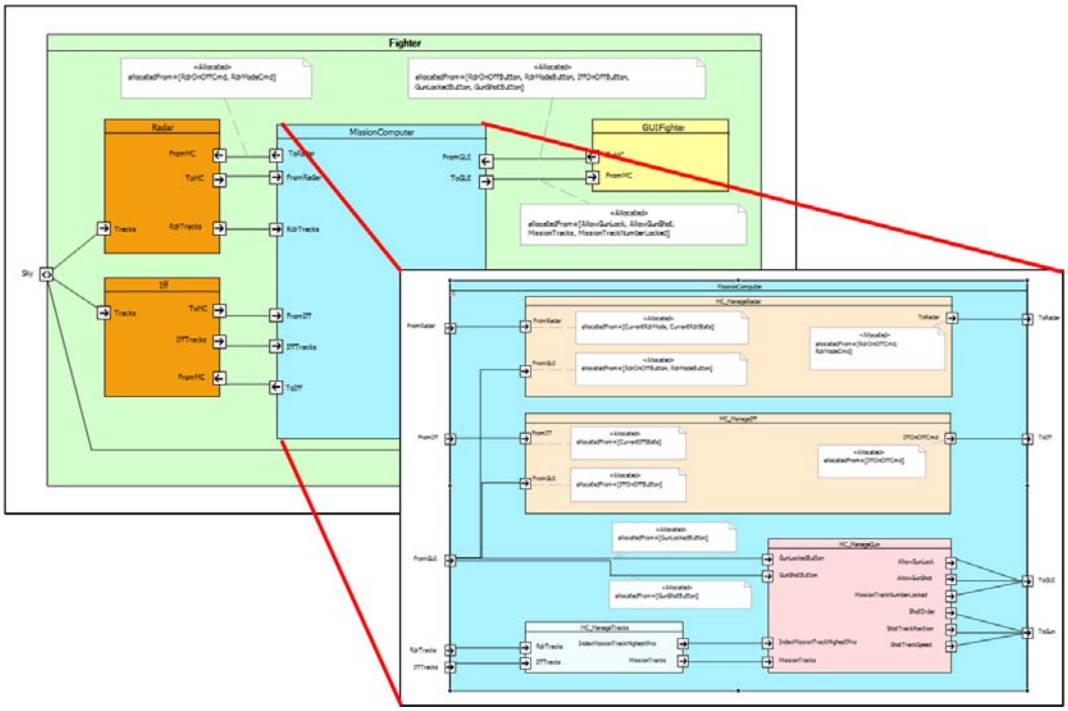


Figure 5.12: Architecture definition of the Fighter Mission system

5.5.2 Relationship with Safety Analysis

"*For safety-critical systems, the process of modifying the functional architecture to accommodate implementation constraints is often driven by the need to achieve the very high levels of reliability dictated by the system safety process. For commercial avionics systems, this process is described in ARP 4761. One of the first steps of this process is the system Functional Hazard Assessment (FHA) that identifies the high-level system hazards. The FHA is then used during the Preliminary System Safety Assessment (PSSA) to determine if the system could contribute to the realization of any of these hazards. If so, derived system design safety requirements are levied by the PSSA.*" [DOT/FAA/AR-08/32]

Among other activities, Failure Mode and Effect Analysis (FMEA) are an integral part of the Global Safety Analysis of the System.

FMEA Approach is an inductive step-by-step approach that identifies all possible failures in a design. Failure Mode defines the way in which something can fail. Effect Analysis is about reviewing the consequences of the failure.

This analysis is performed on the preferred Architecture of Items that was defined.

FMEA Process consists in calculating a Risk Priority number for each Failure Mode that has significant Effect as shown in Figure 5.13.

In the MBSE context, local FMEA is performed for each functional block, then propagation to the upper level of the architecture is examined.

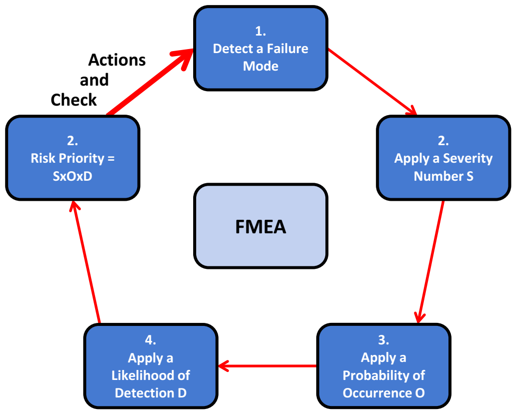

1. Detect a Failure Mode
2. Apply a Severity Number S
3. Apply a Probability of Occurrence O
4. Apply a Likelihood of Detection D
2. Risk Priority = SxOxD
Actions and Check
FMEA

Figure 5.13: Local FMEA at block level

SCADE System's graphical representation significantly facilitates FMEA within blocks as well as the propagation to the whole system.

5.6 Architecture Validation

Check each item and their communication means:

- **Naming**: Verify that the name/description of the Item is clear and is meaningful
- **Uniqueness**: Verify that no other equivalent element (in terms of attributes & relationships) exists within the same architecture
- **Usefulness**: Verify that no item is detached from the Product architecture, all ports are connected to other ports or to actors.

Check the whole actual Architecture to ensure the quality of the defined System architecture. The following points are suggested to be evaluated:

- **Readability**: The Architecture must be understandable by all stakeholders. There must be no ambiguity in the architecture description
- **Level of Abstraction**: The architecture must show the right elements at the right place. This is good moment to collect innovative ideas for solutions
  - Does the item support the function in the right way?
  - Could this be done differently? In case of problem, state it as functional analysis warning for future consideration
- **Level of granularity**: at the same level of abstraction, the granularity of items should be equivalent, in terms of item value and associated communication means.
- **Coverage**: using SCADE System Allocation Tables (please see next section) check that the items of the Architecture Definition cover the functions of the Functional Decomposition.
- **Consistency** of the whole architecture:
  - Is there redundancy of different parts of the architecture?
  - Are there contradictions or inconsistencies within the architecture?

SYNTHESIS AND SELECTION OF THE ARCHITECTURE SOLUTION

A trade-off review is required to compare candidate system architectures. The objective is to select the preferred solution, *i.e.*, most adapted to the operational, functional, business, and organizational drivers coming from the whole set of stakeholders.

Modeling and 0D-Simulation can be used to help select the preferred Architecture.

For a given criterion (for example: power consumption, weight, temperature, etc.) a set of items can be abstracted/modeled according to the various architectures. Simulation results are compared.

5.7 Allocations

As shown in the figure below, there are multiple allocations created along the SCADE MBSE process.

However, in the SCADE MBSE V-Cycle (Figure 5.1), two allocation activities are pinpointed:

- Allocation of functions to items
- Allocation of requirements to items

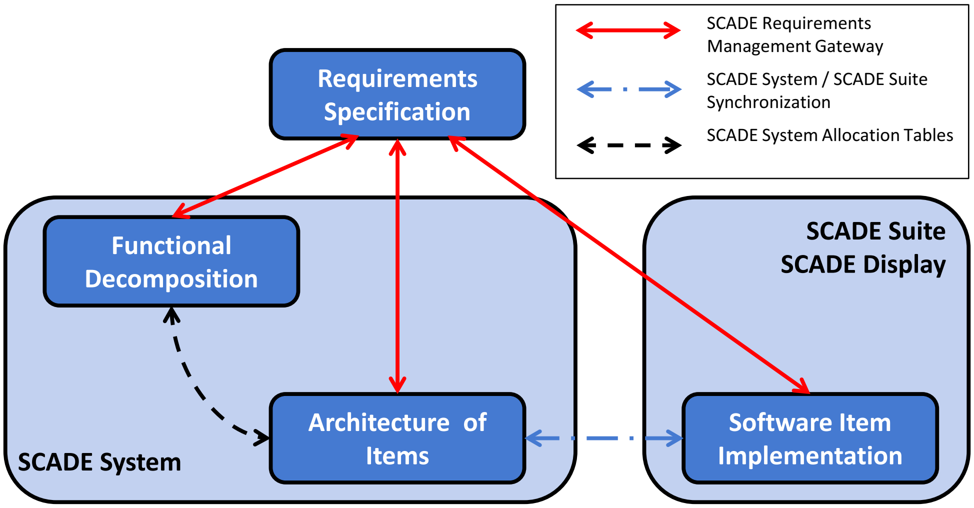

SCADE Requirements Management Gateway
SCADE System / SCADE Suite Synchronization
SCADE System Allocation Tables
Requirements Specification
Functional Decomposition
Architecture of Items
SCADE System
SCADE Suite
SCADE Display
Software Item Implementation

Figure 5.14: Several types of allocations

Allocation of functions to items is highlighted because it is the best method to verify that the functional decomposition and the architecture organization are satisfactorily completed and that they match together.

Allocation of requirements to items is highlighted as well because it is the major outcome of the initial phases of the whole SCADE MBSE process. The main inputs of the software process mentioned in the DO-178B standard are requirements allocated to software.

5.7.1 Allocation of Requirements to Functions

This allocation involves traceability links between requirements and functions.

SCADE LifeCycle Requirements Management Gateway supports the tracing of links between requirements and functions.

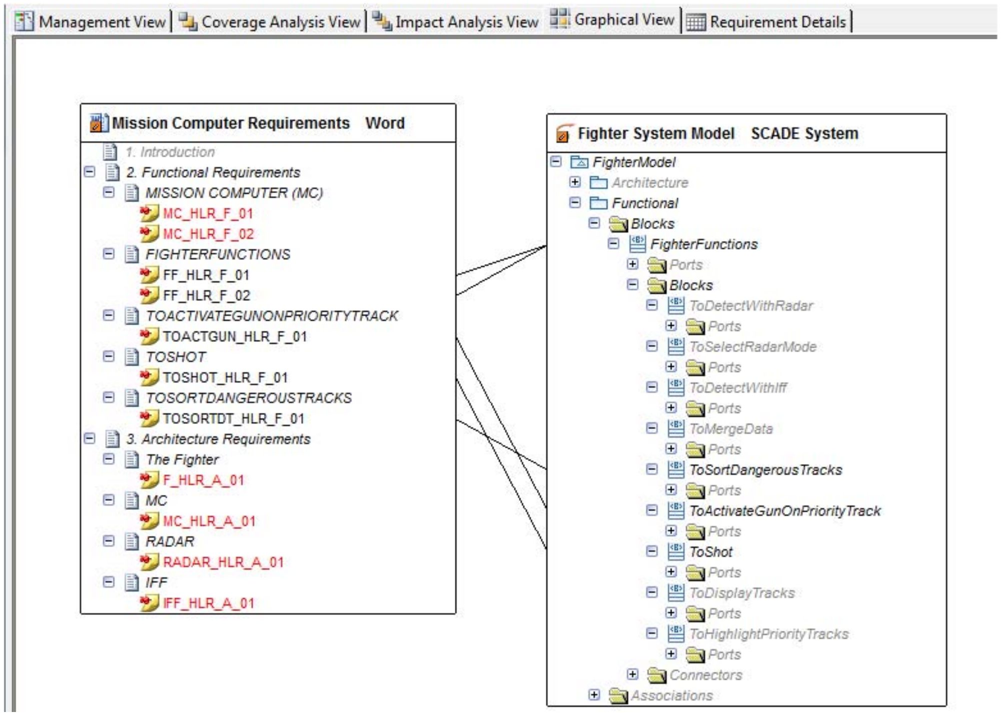


Figure 5.15: Requirements Management Gateway links between requirements and functions

SCADE LifeCycle Requirements Management Gateway analyzes the traceability information to elaborate and display results on:

- How the requirements are covered
- What are the uncovered requirements
- What is the impact of a requirement change

This allocation activity stops when all requirements are covered by functions.

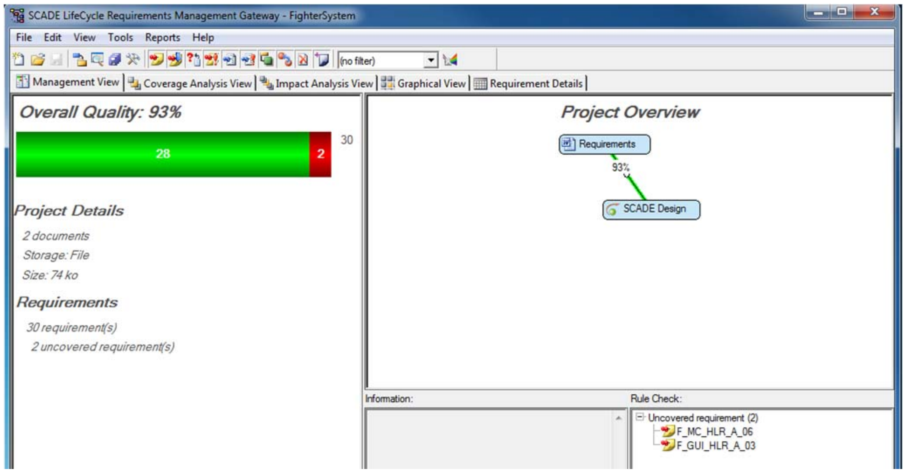


Figure 5.16: Requirements Management Gateway coverage of requirements by functions

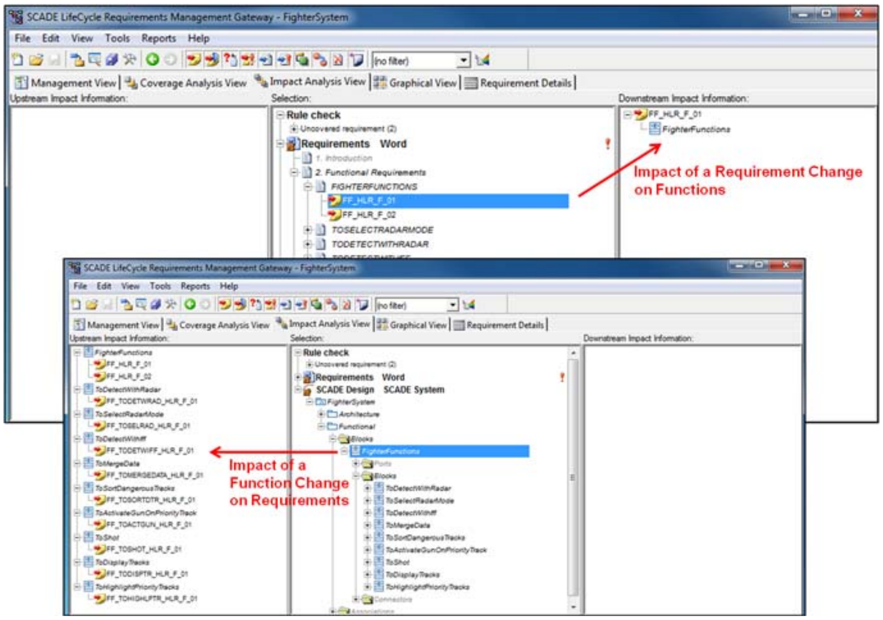


Figure 5.17: Requirements Management Gateway impact analysis between requirements and functions

5.7.2 Allocation of Functions to Items

The allocation activity using SCADE System Allocation Tables consists in “projecting” one object of the functional decomposition package to an object of the architecture definition package.

These objects are typically Blocks, Ports, and Data.

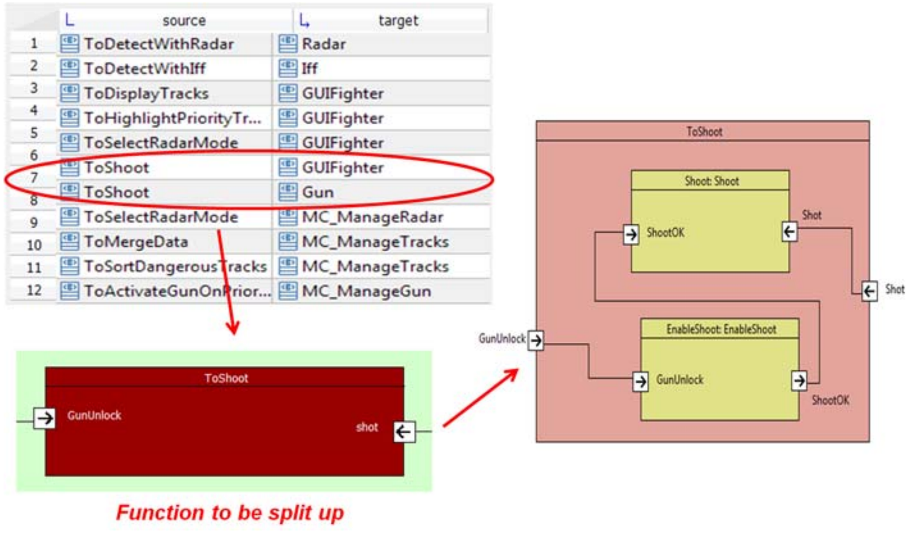
Objects are allocated: blocks that represent functions to blocks that represent item.

Interfaces of these objects and data that describe these objects can be allocated as well.

For example, a Speed variable that is communicated by a function to another function can be allocated to a set of logical data of a Logical Architecture definition and then to an ARINC 429 frame defined in a Physical Architecture.

Best practices:

- Allocation activity is driven by the question: which item is supporting this given function?
- Except if replication is required for safety reason, each function is allocated to only one item; if a function is about to be allocated to more than one item, it should mean that this function is realized several times for redundancy purposes. It is not a good practice to have a function that is jointly realized several items; in that case the function must be further split up.
- On the other hand, an item can support several functions.



	source	target
1	ToDetectWithRadar	Radar
2	ToDetectWithIff	Iff
3	ToDisplayTracks	GUIFighter
4	ToHighlightPriorityTr...	GUIFighter
5	ToSelectRadarMode	GUIFighter
6	ToShoot	GUIFighter
7	ToShoot	Gun
8	ToSelectRadarMode	MC_ManageRadar
9	ToMergeData	MC_ManageTracks
10	ToSortDangerousTracks	MC_ManageTracks
11	ToActivateGunOnPrior...	MC_ManageGun

Figure 5.18: Allocation table of functions to items: an efficient way to manage functional decomposition

5.7.3 Allocation of Requirements to Items

Once requirements are allocated to functions and functions to items, and 100 percent coverage is achieved, the last duty before giving requirements and system design decisions to the implementations is to allocate requirements to items and finally to deliver the allocated requirements to every team in charge of item implementation.

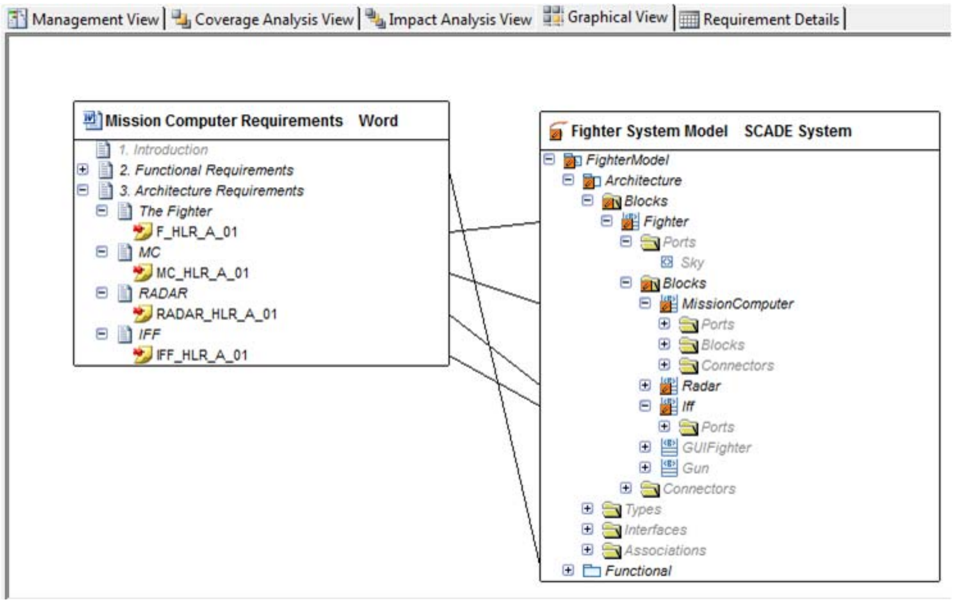


Figure 5.19: Requirements Management Gateway links between requirements and items

5.8 Dissemination of Interfaces and Data

In addition to the requirements, the other design decisions made during early systems engineering activities must be dispatched in an organized way among:

- Items (Blocks) definitions and naming
- Inputs and Outputs
- Data that describe the Items

5.8.1 System/Software Synchronization

The team in charge of developing a given software item expects from the Systems Engineering process:

- The requirements allocated to the item
- The top-level architecture and interface, including naming of the item

SCADE System provides a synchronization feature between SCADE System model and SCADE Suite software model that avoids duplication of efforts and error-prone redesign:

- Synchronization of interfaces defined at system level and refined at software level
- Management of the incremental design thanks to the re-import feature: the System Model Evolution directly impacts the relative software model without clearing local changes done on this software model

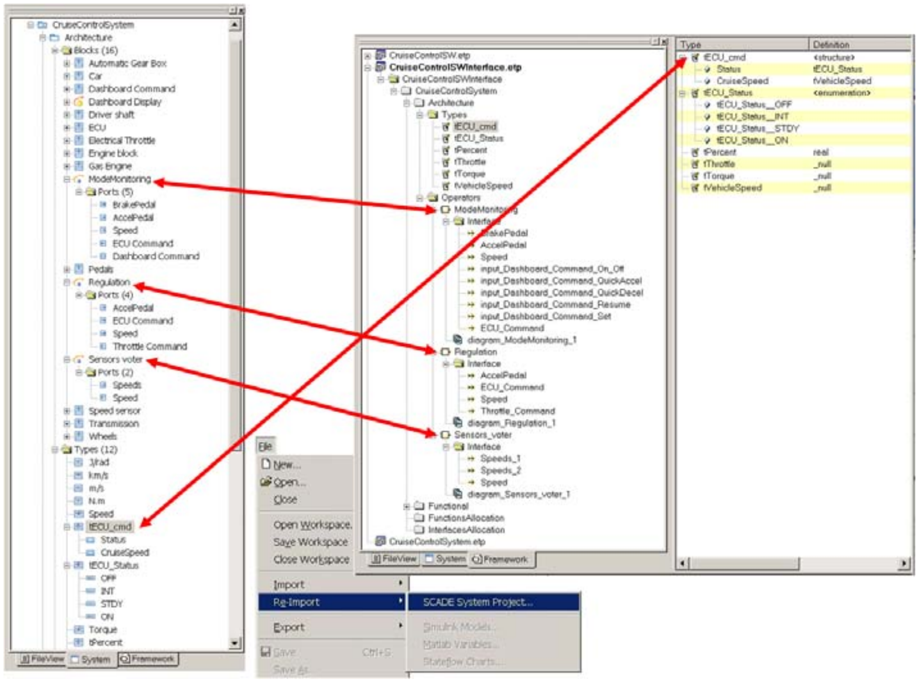


Figure 5.20: System – Software Synchronization: System Model in SCADE System

5.9 Data Dictionary Management

It is compulsory that System Engineers provide to the teams in charge of the implementations additional rationale or comments on why a requirement was established and/or why a particular value or range was specified.

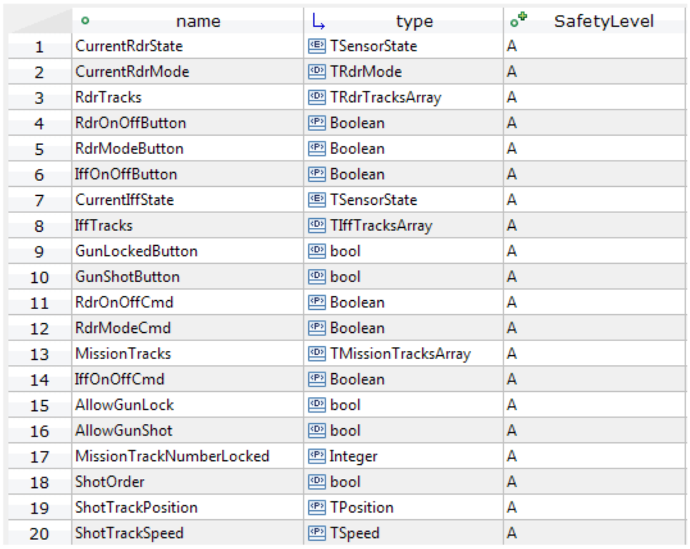
For Software Items, the FAA recommends what follows:

"*Recommended Practice 2.9.1: For each input the software must read, provide a description of anything the software developer must know to access and correctly interpret the input. This may include an input description, the data format, the range of values it may assume, its location, and any protocols to follow when accessing it.*

Recommended Practice 2.9.7: For each output the software must set, provide a description of anything the software developer must know to access and correctly set its value. This may include an output description, the data format, the range of values it may assume, its location, and any protocols." [DOT/FAA/AR-08/32]

Many data can be defined at SCADE System model level. These data carry additional rationale, statements, ranges, etc.

In SCADE System, such data can be created from several sources: properties, tables, etc.



	name	type	SafetyLevel
1	CurrentRdrState	TSensorState	A
2	CurrentRdrMode	TRdrMode	A
3	RdrTracks	TRdrTracksArray	A
4	RdrOnOffButton	Boolean	A
5	RdrModeButton	Boolean	A
6	IffOnOffButton	Boolean	A
7	CurrentIffState	TSensorState	A
8	IffTracks	TIffTracksArray	A
9	GunLockedButton	bool	A
10	GunShotButton	bool	A
11	RdrOnOffCmd	Boolean	A
12	RdrModeCmd	Boolean	A
13	MissionTracks	TMissionTracksArray	A
14	IffOnOffCmd	Boolean	A
15	AllowGunLock	bool	A
16	AllowGunShot	bool	A
17	MissionTrackNumberLocked	Integer	A
18	ShotOrder	bool	A
19	ShotTrackPosition	TPosition	A
20	ShotTrackSpeed	TSpeed	A

Figure 5.21: Data Dictionary in SCADE System

5.10 ARP-475A A-1 Table: Sections Related to Requirements

Table A-1, Process Objectives, Outputs and System Control Category, lists the objectives and expected outputs of the Systems Engineering processes.

The processes related to requirements are:

- Aircraft and System Development Process and Requirement Capture
- Safety Assessment Process
- Requirements Validation Process

The table below provides the contributions of the SCADE MBSE approach.

Objectives		Section	Output	Comments	SCADE Model-Based Systems Engineering Solution
Nr.	**Description**				
2.0 Aircraft and System Development Process and Requirements Capture					
2.1	Aircraft-level functions, functional requirements, functional interfaces and assumptions are defined	4.1.4 4.2 5.3	List of Aircraft-level functions Aircraft-level requirements	Note: requirements capture process objectives presented in section 5.3 are included in this development process	Aircraft-level functions are high-level activities and are not necessarily associated with a single, physical system implementation. Not managed by the SCADE MBSE approach.
2.2	Aircraft functions are allocated to systems	4.1.5 4.3	System Requirements		
2.3	System requirements, including assumptions and system interfaces are defined	5.3	System Requirements		SCADE System is supporting the achievement of these objectives: Functional decomposition using SCADE System combined with SCADE LifeCycle Requirements Management Gateway and SCADE LifeCycle Reporter allow the correct and complete establishment of system requirements.
2.4	System derived requirements (including derived safety-related requirements) are defined and rationale explained	4.4 5.3.1.4 5.3.2	System Requirements		
2.5	System architecture is defined	4.1.6 4.4 5.8.4.4	System Design Description		SCADE System enables the establishment of the architecture of items. The top level System Design Description is automatically generated from the SCADE System model using SCADE LifeCycle Reporter.

Objectives		Section	Output	Comments	SCADE Model-Based Systems Engineering Solution
Nr.	Description				
2.6	System requirements are allocated to the items	4.1.7 4.5 4.6 5.3	Item Requirements		SCADE System allocation tables together with SCADE LifeCycle Requirements Management Gateway enable the allocation of requirements to every item.
2.7	Appropriate item, system and aircraft integrations are performed	4.6.3 4.6.4	Verification Summary		ANSYS Simplorer allows for virtual integration of software items developed with SCADE Suite and SCADE Display and Reduced Order Models of other items within a model of the aircraft environment (MIL). The same tools used in conjunction with Hardware prototyping tools also allow to do a mix of physical and virtual integration (SIL, HIL).
3.0 Safety Assessment Process					
3.1	The aircraft/system functional hazard assessment is performed	5.1.1 5.2.3 5.2.4	Aircraft FHA System FHA		
3.2	The preliminary aircraft safety assessment is performed	5.1.2 5.2.3 5.2.4	PASA		
3.3	The preliminary system safety assessment is performed	5.1.2 5.1.6 5.2.3 5.2.4	PSSA		
3.4	The common cause analyses are performed	5.1.4	Particular Risk Assessment		
			Common Mode Analysis		
			Zonal Safety Analysis		
3.5	The aircraft safety assessment is performed	5.1.3 5.1.6	ASA		

Objectives		Section	Output	Comments	SCADE Model-Based Systems Engineering Solution
Nr.	**Description**				
3.6	The system safety assessment is performed	5.1.3 5.1.6	SSA		The selected architecture of items and their interactions modeled in SCADE System is given to the Safety team to perform safety analysis on it.
3.7	Independence requirements in functions, systems and items are captured	5.3.2 5.2.3 5.1.3	System, HW, SW Requirements PASA PSSA		
4.0 Requirements Validation Process					
4.1	Aircraft, system, item requirements are complete and correct	5.4 5.4.2c 5.4.3 5.4.4	Validation Results	Includes coordination of interfaces between systems and between items	Functional Decomposition model and Architecture model of the System elaborated using SCADE System are providing an organized formal basis to assess completeness and correctness of the set of requirements. SCADE LifeCycle Rapid Prototyper together with SCADE Suite, SCADE Display, and ANSYS Simplorer allow the prototyping of functions that enables the validation of the correctness of the requirements. SCADE LifeCycle Requirements Management Gateway and SCADE System allocation tables are supporting the validation of the completeness of the requirements.
4.2	Assumptions are justified and validated	5.4.2d	Validation Results		
4.3	Derived requirements are justified and validated	5.3.1.4 5.3.2 5.4.2	Validation Results		
4.4	Requirements are traceable	5.4.3 5.4.4	Validation Results		Requirements traceability is ensured with the use of SCADE LifeCycle Requirements Management Gateway.

Objectives		Section	Output	Comments	SCADE Model-Based Systems Engineering Solution
Nr.	Description				
4.6	Validation compliance substantiation is provided	5.4.2e 5.4.2f 5.4.8 5.4.7.4	Validation Summary (including Validation Matrix)		

5.11 Item Modeling

Modeling, including Simulation, is recognized as the shortest path to item.

It enables rapid evaluation of design choices. It avoids hidden pitfalls. It helps rehearsing development strategies so that the implementation phase is significantly optimized.

3D models for mechanical items can be developed using a large set of ANSYS Tools.

0D models for electronic hardware items can be elaborated with ANSYS Simplorer.

0D models of software items can be developed with SCADE Suite and SCADE Display.

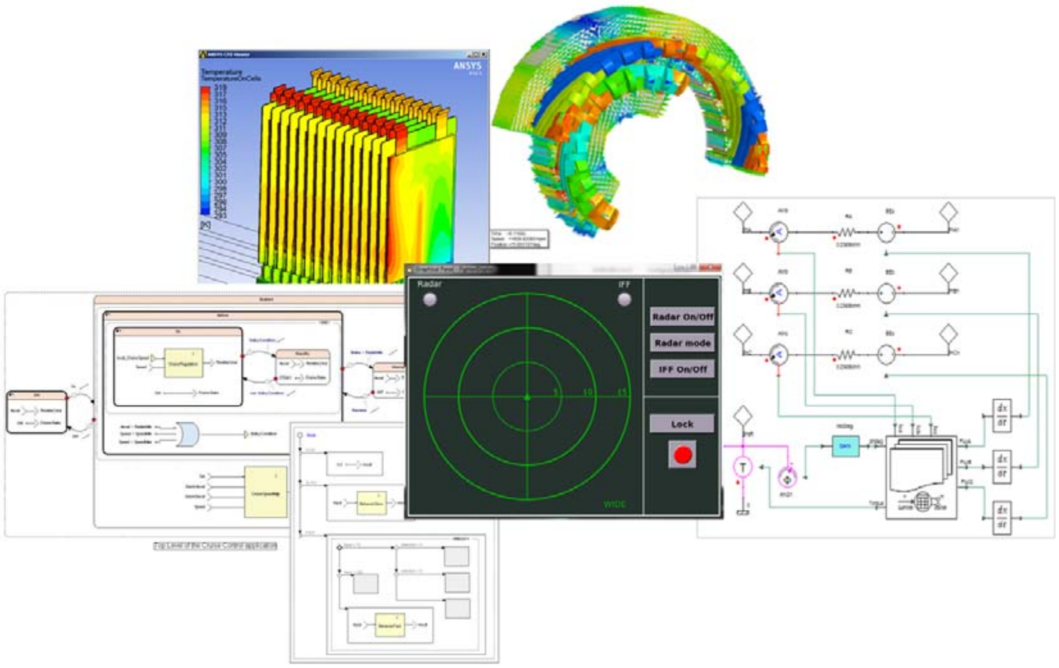


Figure 5.22: 3D models and 0D models

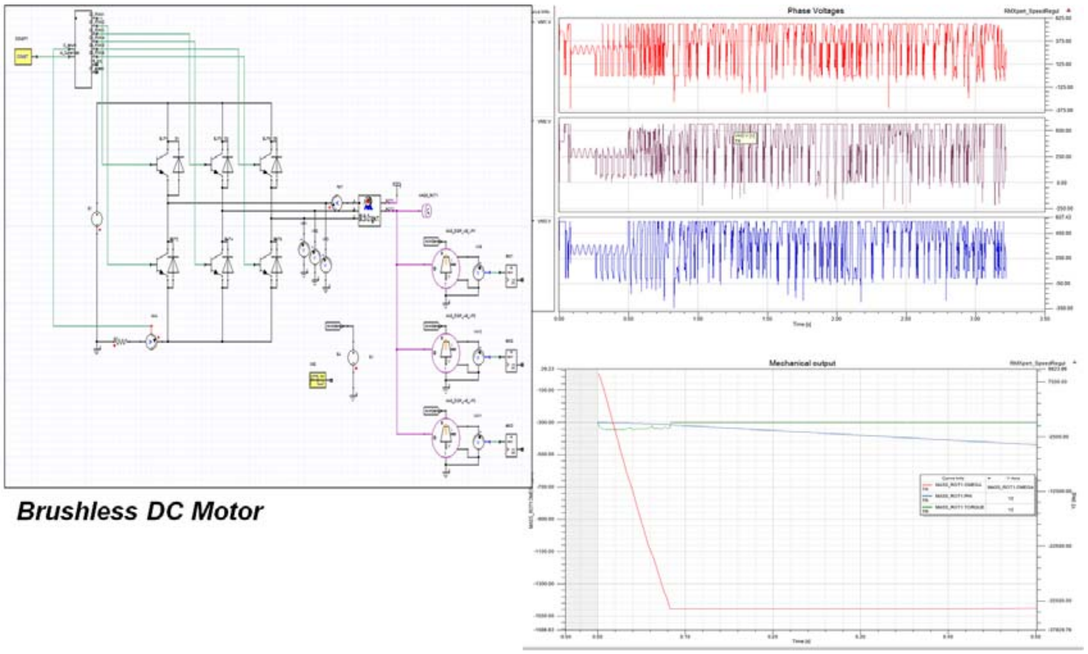


Figure 5.23: Simulation in Simplorer

While, many techniques are employed to increase their fidelity to real behaviors, most of the models stay virtual, models made with SCADE Suite or SCADE Display are also the real software applications to be integrated as physical items of the built system.

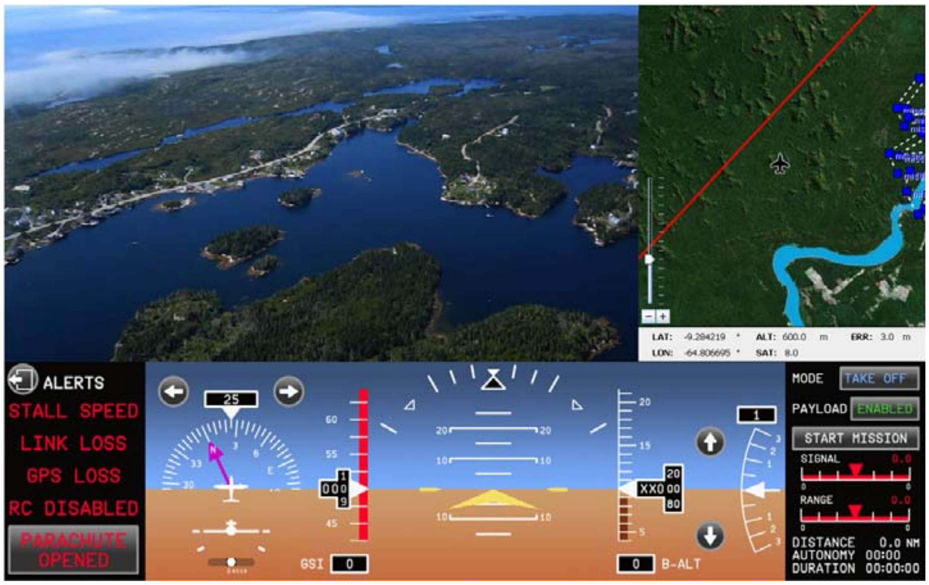


Figure 5.24: Simulation of SCADE Display Application (UAV Ground Station Control)

5.12 Item Implementation

Implementation of software and hardware electronics are respectively driven by DO-178B (or DO-178C), and DO-254 guidelines.

ARP-4754A, DO-178, and DO-254 are assurance-focused processes that establish that the development, from system design to item design, is conducted in a way that guarantees the aircraft safety.

They are all dependent on each other (see Figure 3.1) and all objective-based (as summarized in the following tables):

- Table A-1 for ARP-4754A
- Tables A1 to A10 for DO-178B/C
- Table A-1 for DO-254

Main Development Processes identified by each standard are linked:

Table 5.1: Development Processes by standard

ARP-4754A Development	DO-178B/C Development	DO-254 Development
Requirements & Functions	Requirements	Requirements
System Architecture	Design	Conceptual
Allocation of Requirements	Coding	Detailed
Implementation	Integration	Implementation
		Transition to Production

The majority of Integral and Supporting Processes identified by each standard are linked as well:

Table 5.2: Integral and Supporting Processes by standard

ARP-4754A Integral Processes	DO-178B/C Supporting Processes	DO-254 Supporting Processes
Configuration Management	Configuration Management	Configuration Management
Process Assurance	Quality Assurance	Process Assurance
Certification Authority Coordination	Certification Liaison	Certification Liaison
Verification	Verification	Verification
Requirements Validation		Validation
Safety Assessment		
DAL Assignment		
Requirement Capture		

ARP-4754A, DO-178B/C, DO-254 have a very important part in the overall systems development process.

Please consult our handbooks that thoroughly handle this matter. [DISPLAY-HB] and [SUITE-HB]

"Extract from "*Efficient Development of Safe Avionics Software with DO-178B Objectives Using SCADE Suite®*" handbook:

"*The function and architecture of an embedded computer system (i.e., Flight Control, Braking, Cockpit Display, etc.) are defined by system engineers; the associated control laws are developed by control engineers using some* *informal notation or a semi-formal notation mainly based on schema-blocks and/or state machines; and the embedded production software is finally specified textually and coded by hand in C or Ada by software engineers. In this context, qualified code generation from formal models is a technology that may carry strong Return on Investment (ROI), while preserving the safety of the application. Basically, the idea is to describe the application through a software model, including the control laws as described above, and to automatically generate the code from this model using a qualified code generator, in the sense of DO-178B, resulting in the following advantages to the development life cycle:*

- *When a proper modeling approach is defined:*
- *It fulfills the needs of the control engineers, typically using such notations as data flow diagrams and state machines.*
- *It fulfills the needs of the software engineers by supporting the accurate definition of the software requirements and by providing efficient automatic code generation of software having the qualities that are expected for such applications (i.e., efficiency, determinism, static memory allocation, etc.).*
- *It allows for establishing efficient new processes to ensure that safety criteria are met.*
- *It saves coding time, as this is automatic.*
- *It saves a significant part of verification time, as the use of such tools guarantees that the generated source code conforms to the software model.*
- *It allows for identifying problems earlier in the development cycle, since most of the verification activities can be carried out at model level.*
- *It reduces the change cycle time, since modifications can be done at model level and code can automatically be regenerated."*

5.13 Integration and Verification and Validation

Successful system integration and integration testing result from:

- Accurate allocation of requirements to items
- Careful interfaces definition
- True incremental life cycle integration
- Correct deployment of configuration management and other Systems Engineering management activities
- Comprehensive integration testing strategy that verifies the assembled system operates as it should

In addition, especially for software items integration, a thorough path ranging from full Virtual Integration to full Physical Integration is recognized as the most effective way to master this extremely challenging Systems Engineering phase.

ANSYS Simplorer offers this capability, through the FMI/FMU inter-modeling standard, to co-simulate with SCADE Suite or SCADE Display. Then, a software item designed with SCADE Suite or SCADE Display can be integrated in a virtual environment designed with ANSYS Simplorer to fix the single and combined issues of the latest integration prior to integration in "the real world".

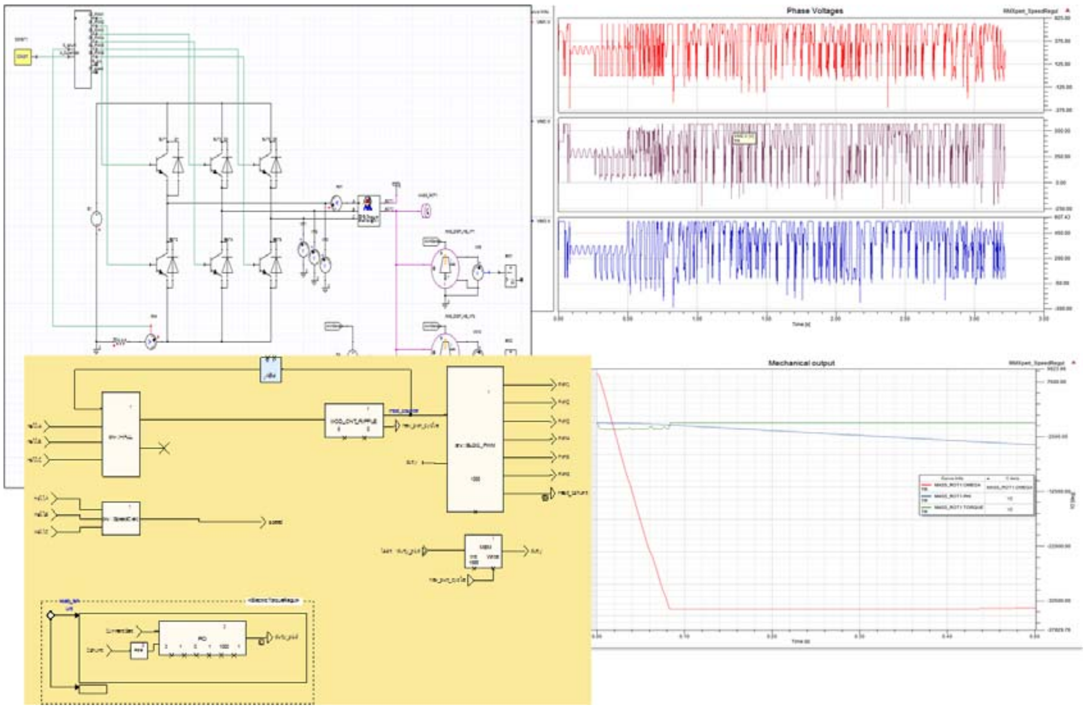


Figure 5.25: Virtual integration of SCADE Suite application in Simplorer simulation

One of the major challenges in Virtual Integration is the fidelity of the model that represents the environment of the item under integration testing.

The capability of ANSYS Simplorer to integrated Reduced Order Models (ROM) of complex 3-D models is a way to reach this fidelity objective.

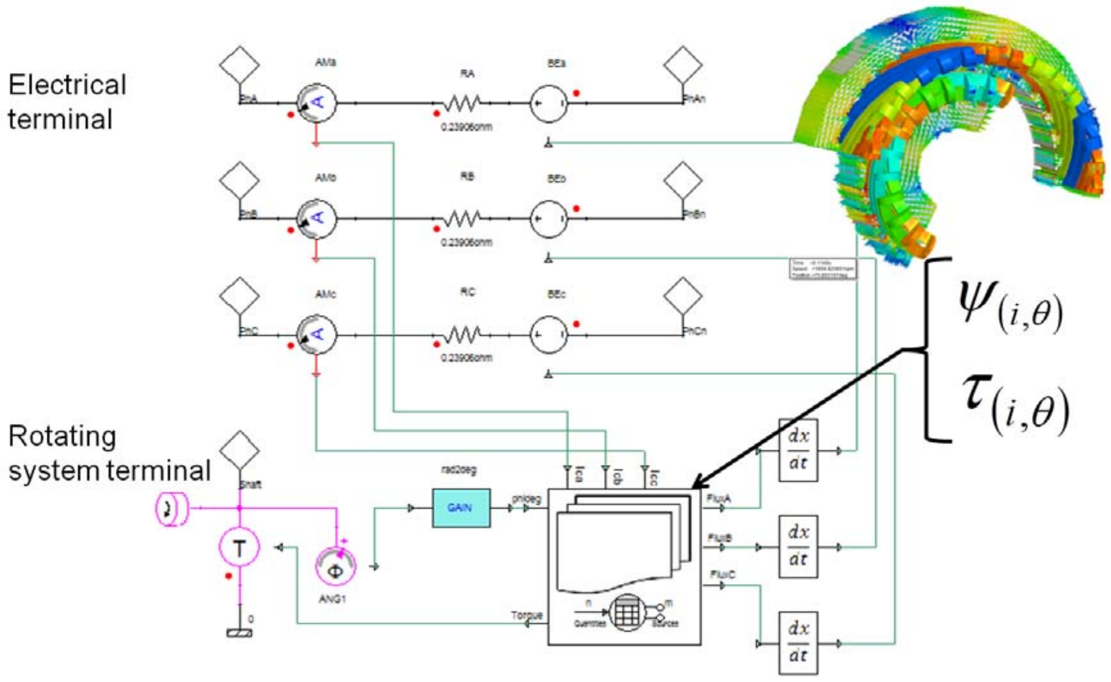


Figure 5.26: Power of ROM

When turning to the Verification and Validation activities (Verification that the built item/system is rightly done, *i.e.*, correctly responds to the requirements, Validation that the built item/system does right, *i.e.*, does what the stakeholders expect), the following Virtual and Physical Environments are respectively called:

- *Model-in-the-Loop* (MIL) testing involves simulation of the tested model coupled with a model of its environment.
- *Software-in-the-Loop* (SIL) testing evaluates the functions of the generated or hand written code in co-simulation on the host machine.
- *Hardware-in-the-Loop* (HIL) testing combines real hardware components with software-based simulation.

At this point, the conditions for accurate and thorough testing are met, because HIL environment in which the item/system is placed for testing has a larger scope of parameterization than the targeted real plant in which the item/system is placed at the end.

For example:

- Nominal testing and testing at boundaries of ranges
- Robustness testing: testing beyond specified ranges

- Establishment of the test scenario that leads to the Worst-Case Execution Time (WCET) – usually to be measured in the real plant or to be analyzed/ altered with the real plant conditions
- Safety testing: testing at failure conditions

SCADE Suite provides a gateway with National Instrument LabVIEW toolset.

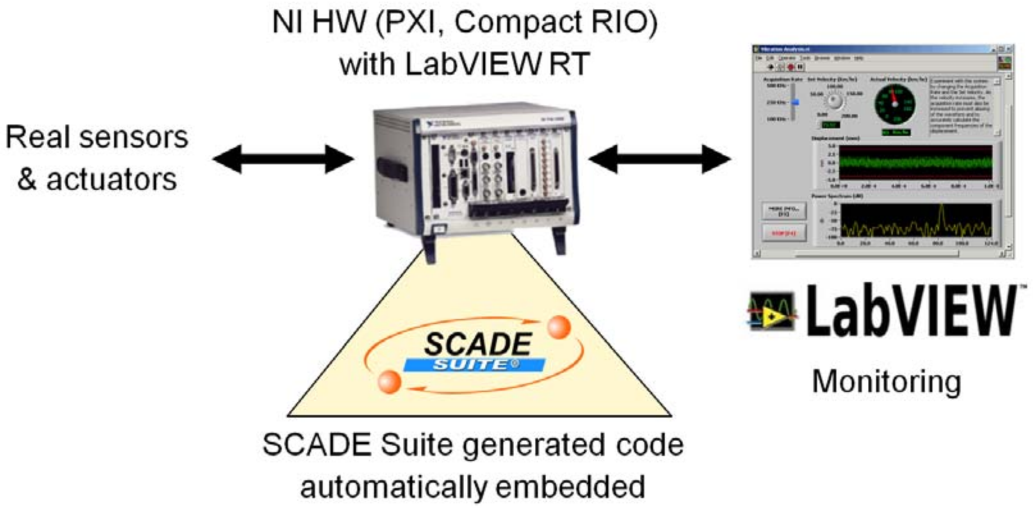

NI HW (PXI, Compact RIO) with LabVIEW RT
Real sensors & actuators
SCADE SUITE
SCADE Suite generated code automatically embedded
LabVIEW
Monitoring

Figure 5.27: Example of HIL simulation with NI LabVIEW and SCADE System

5.14 ARP-475A Tables Related to Implementation, Integration, and V&V

Objectives		Section	Output	Comments	SCADE Model-Based Systems Engineering Solution
Nr.	**Description**				
5.0 Implementation Verification Process					Objective: ensure the system implementation satisfies the validated requirements
5.1	Test or demonstration procedures are correct	5.5.4.3	Verification Procedures		Test Engineers that have to design comprehensive integration test cases and procedures can efficiently rely on the model of the system designed with SCADE System.
5.2	Verification demonstrates intended functions and confidence of no unintended function impacts to safety	5.5.1 5.5.5.3 5.5.5.2	Verification Procedures		
			Verification Results		
5.3	Product implementation complies with aircraft, and system requirements	5.5.1 5.5.2	Verification Procedures		Please consult our handbooks [DISPLAY-HB] and [SUITE-HB]
			Verification Results	Specific item verification activities are performed under DO-178B/ED-12B and DO-254/ED-80	
5.4	Safety requirements are verified	5.5.1 5.5.5.3	Verification Procedures and Results (ASA, SSA)	See Appendix A, section 3.0, Safety Assessment Objectives for specific safety objectives and control categories	
5.5	Verification compliance substantiation is included	5.5.6.3 5.5.6.4	Verification Matrix		
			Verification Summary		

Objectives		Section	Output	Comments	SCADE Model-Based Systems Engineering Solution
Nr.	Description				
5.6	Assessment of deficiencies and their related impact on safety is identified	5.5.6.4	Verification Summary		
			Problem Reports		
6.0 Configuration Management Process					All data from SCADE tools are easily managed in configuration thanks to SCCI-based connection to the major Configuration Management tools.
6.1	Configuration items are identified	5.6.2.2	CM Records		
6.2	Configuration baseline and derivatives are established	5.6.2.3	Configuration Baseline records		
6.3	Problem reporting, change control, change review, and configuration status accounting are established	5.6.2.4	Problem reports CM Records		
6.4	Archive and retrieval are established	5.6.2.5	CM Records		

6. Conclusions

This handbook identified the main challenges of engineering a Civilian Avionics System that must meet ARP-4754A objectives:

- Establish the complete set of correct requirements of the System
- Enable early safety analysis
- Build the preferred architecture of items that will be developed by specific teams
- Organize the requirements allocated to every items
- Set up interfaces of items and create naming for all the teams
- Make decisions, create assumptions, produce useful data for all the teams
- Constantly manage an truthful configuration of System artifacts
- Manage a streamlined integration and V&V

ESTABLISH THE COMPLETE SET OF CORRECT REQUIREMENTS OF THE SYSTEM

No other part of designing a System is as difficult as establishing the System Requirements. No other task has such important impact on the resulting system if done wrong. No other part of the Systems Engineering work is as costly to correct later.

We showed that graphically decomposing functions with SCADE System and modeling operations using SCADE LifeCycle Rapid Prototyper both assist Systems engineers in validating the requirements correctness.

We highlighted several advantages of the SCADE Model-Based Systems Engineering approach:

- Modeling with SCADE System helps visualize the entire system in its environment, thus facilitates the common understanding with users and other stakeholders
- Modeling with SCADE System naturally disambiguate requirements and functions; in addition, SCADE System Checker allows checking duplications, naming consistency, and all other rules that the Systems Engineering team thinks appropriate
- Iterations (adding details) and Increments (completing the model) are facilitated
- Systems engineers stay focus on the primary problem rather than on the document structure and consistency
- Dependencies and allocations are explicitly captured
- Documentation is automatically generated from the model

We exposed, as well, that using SCADE LifeCycle Requirements Management Gateway is key to validate the requirements completeness.

ENABLE EARLY SAFETY ANALYSIS

There is trend towards Model-Based Safety Analysis (MBSA) that intends to automate portions of the safety Analysis process.

MBSA is based on the existence of one or several formal models of the System.

Thus, MBSA relies on MBSE, and we demonstrated in this Handbook that the SCADE System architectural models are the primary inputs of the MBSA process.

This joint approach carries several benefits:

- Tighter relationship between systems and safety analysis based on shared models
- Earlier exploration of potential safety hazards, when it is less costly to introduce additional requirements for safety reason
- Easier and faster verification of the impact of safety requirements on the system architecture

BUILD THE PREFERRED ARCHITECTURE OF ITEMS

Models are often used to describe the architecture of a System.

SCADE MBSE approach brings much more than just drawings. SCADE MBSE approach provides the opportunity to establish several architectures and supports the synthesis and finally the design choice.

We showed how SCADE System preserves the integrity of all artifacts designed at Systems Engineering level and distributes and maintains the elements required to keep all item implementations in sync.

ALLOCATE THE REQUIREMENTS TO ITEMS

This is the primary outcome of the Systems Engineering process. This is about safe, on-time, accurate communication.

All the inner activities (from Requirements Analysis to Architecture Definition) of the SCADE MBSE are mostly focused on this objective: define the set of requirements that are allocated to a given Item, thus to a given team.

We illustrated how SCADE System, and its ecosystem, provides the path and the means to elaborate this outcome.

SET UP INTERFACES AND NAMING AND PRODUCE USEFUL DATA

This is the also an extremely important outcome of the SCADE MBSE process.

In order to keep all the implementations of items in sync and to efficiently anticipate a painless integration, it is mandatory to produce, distribute and keep updatable the interfaces and naming included in the architecture.

In the same way, data contained within architectural descriptions aid the teams understanding and applying the system structure.

We showed how SCADE System and SCADE Suite / SCADE Display synchronization feature efficiently automates this Systems Engineering task.

Managing the Interface Control Documentation (ICD) is a real pain. We indicated how SCADE System Data Dictionary features provide Systems engineers with means to create and store data related to contents of interfaces, and, also importantly, with means to export and import files of data.

MANAGE THE CONFIGURATION OF SYSTEM ARTIFACTS

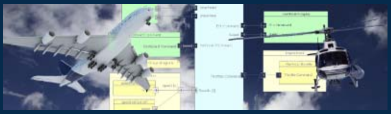
We reminded you that Systems Engineering is an iterative and incremental problem solving process.

Secure management of system artifacts using SCADE System inner model representation of diagrams, flows and tables of data, as well as SCADE LifeCycle RM Gateway and SCADE Configuration Management Interface, have been shown as efficient contributors to a safe progressive baselining of System artifacts, and finally to a proficient management of the Systems Engineering Project LifeCycle.

STREAMLINE INTEGRATION AND V&V

Suitable integration of Items starts with a reliable architecture and well-known interfaces between Items.

In addition, we exposed how the wide collection of 0D at 3D modeling tools provided by ANSYS and partners are contributing to a cautious step-by-step Integration and V&V process, from virtual to physical.



Appendixes and Index

A ARP-4754A Integral Process Tables

DEVELOPMENT PLANNING ELEMENTS

Planning Element	Element Description	4754A sections
Development	Establish the process and methods to be used to provide the framework for the aircraft/system architecture development, integration and implementation.	3 and 4.0
Safety Program	Establish scope and content of the safety activities related to the development of the aircraft or system.	5.1.5
Requirements Management	Identify and describe how the requirements are captured and managed. Sometimes these elements are included in conjunction with the validation elements.	5.3
Validation	Describe how the requirements and assumptions will be shown to be complete and correct.	5.4
Implementation Verification	Define the processes and criteria to be applied when showing how the implementation satisfies its requirements.	5.5
Configuration Management	Describe the key development related configuration items and how they will be managed.	5.6
Process Assurance	Describe the means to assure the practices and procedures to be applied during system development are followed.	5.7
Certification	Describe the process and methods that will be used to achieve certification	5.8

The table below classifies objectives by DAL as follows:

- R*: Recommended for certification with process independence [1]
- R: Recommended for certification
- A: As negotiated for certification
- N: Not required for certification

1. Independence for safety artifacts is achieved when the safety activity is performed by a person(s) other than the developer of the system/item.

Table A.1: Process Objectives, Outputs, and System Control Category

Objectives		Section	Output	Applicability and Independence by Development Assurance Level				
Nr.	Description			A	B	C	D	E
1.0 Planning Process								
1.1	System Development and integral processes activities are defined	5.8.1 5.8.4.1	Certification Plan	R	R	R	R	R
		3.1 5.1.5 App B	Safety Program Plan	R	R	R	R	N
		3.1 5.8.4.3	Development Plan	R	R	R	R	N
		5.4.2a 5.4.7.1	Validation Plan	R	R	R	A	N
		5.5.3 5.5.5.1	Verification Plan	R	R	R	A	N
		5.6.2.1	Configuration Management Plan	R	R	R	R	A
		5.7.2	Process Assurance Plan	R	R	R	R	N
1.2	Transition criteria and interrelationship among processes are defined	3.2	Plans in Objective N°1	R	R	R	A	N
2.0 Aircraft and System Development Process and Requirements Capture								
2.1	Aircraft-level functions, functional requirements, functional interfaces and assumptions are defined	4.1.4 4.2 5.3	List of Aircraft-level functions Aircraft-level requirements	R	R	R	R	N
2.2	Aircraft functions are allocated to systems	4.1.5 4.3	System Requirements	R	R	R	R	N
2.3	System requirements, including assumptions and system interfaces are defined	5.3	System Requirements	R	R	R	R	N

Table A.1: Process Objectives, Outputs, and System Control Category (Continued)

Objectives		Section	Output	Applicability and Independence by Development Assurance Level				
Nr.	Description			A	B	C	D	E
2.4	System derived requirements (including derived safety-related requirements) are defined and rationale explained	4.4 5.3.1.4 5.3.2	System Requirements	R	R	R	A	N
2.5	System architecture is defined	4.1.6 4.4 5.8.4.4	System Design Description	R	R	R	A	N
2.6	System requirements are allocated to the items	4.1.7 4.5 4.6 5.3	Item Requirements	R	R	R	R	N
2.7	Appropriate item, system and aircraft integrations are performed	4.6.3 4.6.4	Verification Summary	R	R	R	A	N
3.0 Safety Assessment Process								
3.1	The aircraft/system functional hazard assessment is performed	5.1.1 5.2.3 5.2.4	Aircraft FHA System FHA	R*	R*	R	R	R
3.2	The preliminary aircraft safety assessment is performed	5.1.2 5.2.3 5.2.4	PASA	R*	R*	R	A	N
3.3	The preliminary system safety assessment is performed	5.1.2 5.1.6 5.2.3 5.2.4	PSSA	R*	R*	R	A	N
3.4	The common cause analyses are performed	5.1.4	Particular Risk Assessment	R	R	A	N	N
			Common Mode Analysis	R*	R*	A	N	N
			Zonal Safety Analysis	R	R	A	N	N
3.5	The aircraft safety assessment is performed	5.1.3 5.1.6	ASA	R*	R*	R	A	N

Table A.1: Process Objectives, Outputs, and System Control Category (Continued)

Objectives		Section	Output	Applicability and Independence by Development Assurance Level				
Nr.	**Description**			**A**	**B**	**C**	**D**	**E**
3.6	The system safety assessment is performed	5.1.3 5.1.6	SSA	R*	R*	R	A	N
3.7	Independence requirements in functions, systems and items are captured	5.3.2 5.2.3 5.1.3	System, HW, SW Requirements PASA PSSA	R*	R*	R	R	N
4.0 Requirements Validation Process								
4.1	Aircraft, system, item requirements are complete and correct	5.4 5.4.2c 5.4.3 5.4.4	Validation Results	R*	R*	R	A	N
4.2	Assumptions are justified and validated	5.4.2d	Validation Results	R*	R	R	A	N
4.3	Derived requirements are justified and validated	5.3.1.4 5.3.2 5.4.2	Validation Results	R*	R*	R	A	N
4.4	Requirements are traceable	5.4.3 5.4.4	Validation Results	R	R	R	A	N
4.6	Validation compliance substantiation is provided	5.4.2e 5.4.2f 5.4.8 5.4.7.4	Validation Summary (including Validation Matrix)	R	R	R	A	N

Table A.1: Process Objectives, Outputs, and System Control Category (Continued)

Objectives		Section	Output	Applicability and Independence by Development Assurance Level				
Nr.	**Description**			**A**	**B**	**C**	**D**	**E**
5.0 Implementation Verification Process								
5.1	Test or demonstration procedures are correct	5.5.4.3	Verification Procedures	R*	R	R	A	N
5.2	Verification demonstrates intended functions and confidence of no unintended function impacts to safety	5.5.1 5.5.5.3 5.5.5.2	Verification Procedures	R*	R	R	A	N
			Verification Results	R*	R	R	A	N
5.3	Product implementation complies with aircraft, and system requirements	5.5.1 5.5.2	Verification Procedures	R	R	R	A	N
			Verification Results	R*	R	R	A	N
5.4	Safety requirements are verified	5.5.1 5.5.5.3	Verification Procedures and Results (ASA, SSA)	R*	R*	R	A	N
5.5	Verification compliance substantiation is included	5.5.6.3 5.5.6.4	Verification Matrix	R*	R	R	A	N
			Verification Summary	R*	R	R	A	N
5.6	Assessment of deficiencies and their related impact on safety is identified	5.5.6.4	Verification Summary	R*	R	R	A	N
			Problem Reports	R*	R	R	A	N
6.0 Configuration Management Process								
6.1	Configuration items are identified	5.6.2.2	CM Records	R	R	R	A	N
6.2	Configuration baseline and derivatives are established	5.6.2.3	Configuration Baseline records	R	R	R	A	N
6.3	Problem reporting, change control, change review, and configuration status accounting are established	5.6.2.4	Problem reports CM Records	R	R	R	R	N
6.4	Archive and retrieval are established	5.6.2.5	CM Records	R	R	R	R	N
7.0 Process Assurance Process								

Table A.1: Process Objectives, Outputs, and System Control Category (Continued)

Objectives		Section	Output	Applicability and Independence by Development Assurance Level				
Nr.	**Description**			**A**	**B**	**C**	**D**	**E**
7.1	Assurance is obtained that necessary plans are developed and maintained for all aspects of system certification	5.7.3	Evidence of Process Assurance	R	R	R	A	N
7.2	Development activities and processes are conducted in accordance with those plans	5.7.4	Evidence of Process Assurance	R	R	R	A	N
8.0 Certification and Regulatory Authority Coordination Process								
8.1	Compliance substantiation is provided	5.8.3	Certification Summary	R*	R*	R*	R	N
		5.8.4.2	Configuration Index	R*	R*	R*	R	N

B References

[ANSI/AIAA]	“Guide for the Preparation of Operational Concept Documents”, ANSI/AIAA G-043- 1992.
[ARINC 429]	Aerospace standard that describes the architecture, interfaces, and protocol to carry data on a data-bus. Published 2001.
[ARP-4754A]	“Guidelines for Development of Civil Aircraft and Systems”, Society of Automotive Engineers, 2010-12.
[ARP-4761]	“Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment” Society of Automotive Engineers, 1996.
[ARP-5056]	“Flight Crew Interface Considerations in the Flight Deck Design Process for Part 25 Aircraft”, Society of Automotive Engineers, 2006.
[DISPLAY-HB]	Efficient Development of Safe Avionics Software with DO-178B Objectives Using Esterel SCADE ®, Second Edition (revision 1), Esterel Technologies, February 2012.
[DO-178B/ED-12B]	“Software Considerations in Airborne Systems and Equipment Certification”, RTCA/EUROCAE, December 1992.
[DO-248B/ED-94B]	Final report for clarification of DO-178B “Software Considerations in Airborne Systems and Equipment Certification”, RTCA Inc, October 2001.
[DO-254/ED-80]	“Design Assurance Guidance for Airborne Electronic Hardware”, RTCA Inc, April 2000.
[DOT/FAA/AR-08]	“Guidelines for the Development of a Critical Composite Maintenance and Repair Issues Awareness Course”, U.S. Department of Transportation, Federal Aviation Administration, DOT/FAA/AR-08/54, February 2009.
[DOT/FAA/AR-08/32]	“Requirements Engineering Management Handbook”, FAA, June 2009.
[IEEE P1220]	“Standard for the Application and Management of the Systems Engineering Process”, IEEE, [Final Draft], September 1994.
[IEEE 610.12-1990]	“Standard Glossary of Software Engineering Terminology”, IEEE.
[IEEE 1471-2000]	“Recommended Practice for Architectural Description for Software-Intensive Systems”, IEEE.
[INCOSE]	“Systems Engineering Handbook. A Guide for System Lifecycle Processes and Activities”, INCOSE-TP-2003-002-03.2, January 2010.
[ISO/IEC15288]	“Systems and Software Engineering - System Life Cycle Process”, ISO/IEC 15288:2008.
[RTCA/DO-254]	“Design Assurance Guidance for Airborne Electronic Hardware”, RTCA SC-180, EUROCAE WG-46, April 19, 2000
[RTCA/DO-178B]	“Software Considerations in Airborne Systems and Equipment”, RTCA, January 2012.
[SUITE-HB]	Efficient Development of Safe Avionics Software with DO-178B Objectives Using SCADE Suite®, Fifth Edition (revision 2), Esterel Technologies, January 2012.
[SYSTEM-UM]	“SCADE System User Manual”, Version 2.0 release, Esterel Technologies, March 2013.

C Acronyms and Glossary

ACRONYMS

A/C	Aircraft
API	Application Programming Interface
ARP	Aerospace Recommended Practices
CCA	Common Cause Analysis
CONOPS	CONcept Of OPerationS
COTS	Commercial Off-The-Shelf
DAL	Development Assurance Level
EASA	European Aviation Safety Agency
ECU	Electronic Control Unit
EUROCAE	European Organization for Civil Aviation Equipment
FAA	Federal Aviation Administration
FDAL	Functional Development Assurance Level
FHA	Functional Hazard Analysis
FMI/FMU	Functional Mock-up Interface / Functional Mock-up Unit
HIL	Hardware-in-the-Loop
HLR	High-level Requirement
ICD	Interface Control Documentation
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
ILS	Instrumented Landing System
INCOSE	International Council on Systems Engineering
IP	Intellectual Property
ISO	International Organization for Standardization
HTML	HyperText Markup Language
IDAL	Item Development Assurance Level
LLR	Low-level Requirement
MBSE	Model-Based Systems Engineering
MC/DC	Modified Condition/Decision Coverage
MIL	Model-in-the-Loop
N/A	Not Applicable
N.B.	Nota Bene
PSSA	Preliminary System Safety Assessment
RM	Requirements Management
ROI	Return On Investment
RTCA	Radio Technical Commission for Aeronautics
SAE	Society of Automotive Engineers
SCADE	Safety Critical Application Development Environment
SCCI	Source Code Control Interface
SDPD	Simulation-Driven Product Development
SE	Systems Engineering
SGL	Standard Graphic Library
SIL	Software-in-the-Loop
SQA	Software Quality Assurance
SRATS	System requirements allocated to software
SSA	System Safety Assessment
SSM	SCADE State Machine
SW	Software
TRL	Technology Readiness Level
V&V	Verification and Validation
WCET	Worst Case Execution Time

GLOSSARY

0D-Simulation

0D-Simulation of a model does not handle spatial dimensions. Only time dimension is considered. 0D-Model is an approximation, in general via mathematical equations, of a 3D-domain problem.

Certification

Legal recognition by the certification authority that a product, service, organization, or a person complies with the requirements. Such certification comprises the activity of technically checking the product, service, organization, or person, and the formal recognition of compliance with the applicable requirements by issue of a certificate, license, approval, or other documents as required by national laws and procedures. In particular, certification of a product involves: (a) the process of assessing the design of a product to ensure that it complies with a set of standards applicable to that type of product so as to demonstrate an acceptable level of safety; (b) the process of assessing an individual product to ensure that it conforms with the certified type design; (c) the issuance of a certificate required by national laws to declare that compliance or conformity was found with standards in accordance with items (a) or (b) above.

Certification credit

Acceptance by the certification authority that a process, product, or demonstration satisfies a certification requirement.

Failure

The inability of a system or system component to perform a required function within specified limits. A failure may be produced when a fault is encountered.

Fault

A manifestation of an error in software. A fault, if it occurs, may cause a failure.

Fault toleranceThe built-in capability of a system to provide continued correct execution in the presence of a limited number of hardware or software faults.

Formal methodsDescriptive notations and analytical methods used to construct, develop, and reason about mathematical models of system behavior.

Hardware-in-the-Loop (HIL)

HIL provides a simulation platform that incorporates ways to flexibly handle the behavior of the plant (*i.e.,* the environment) of the component under test.

Hardware/software integration

The process of combining the software into the target computer.

High-level requirements

Software requirements developed from analysis of system requirements.

Host computer

The computer on which the software is developed.

Independence

Separation of responsibilities, which ensures the accomplishment of objective evaluation. (1) For software verification process activities, independence is achieved when the verification activity is performed by a person(s) other than the developer of the item being verified, and a tool(s) may be used to achieve an equivalence to the human verification activity. (2) For the software quality assurance process, independence also includes the authority to ensure corrective action.

Integral process

A process that assists the software development, processes, and other integral processes and, therefore, remains active throughout the software life cycle. The integral processes are the software verification process, the software quality assurance process, the software configuration management process, and the certification liaison process.

Item

A hardware or software element having bounded and well-defined interfaces [ARP-4754A]

Low-level requirements

Software requirements derived from high-level requirements, derived requirements, and design constraints from which source code can be directly implemented without further information.

Mechatronics

“Mechatronics” defines objects that are made of hardware mechanics, hardware electronics, and software.

Model-in-the-Loop (MIL)

MIL is a version of SIL, where the simulation of the component and its environment is handled at model level.

Modified Condition/Decision Coverage

Every point of entry and exit in the program was invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken all possible outcomes at least once, and each condition in a decision was shown to independently affect that decision's outcome. A condition is shown to independently affect a decision's outcome by varying just that condition, while holding fixed all other possible conditions.

Reduced Order Model

A Reduced Order Model is an approximation model of a complex model, by canceling one or more of its dimensions (from 3D to 2D: a thin parallelepiped can be approximated by a rectangle; from 3D to 1D: a thin tube can be reduced as a thread; from 3D to 0D: an electrical circuit can be replaced by an equivalent equation).

Robustness

The extent to which software can continue to operate correctly despite invalid inputs.

Software-in-the-Loop (SIL)

SIL provides a pure software simulation platform, where the plant behavior is emulated by software.

Standard

A rule or basis of comparison used to provide both guidance in and assessment of the performance of a given activity or the content of a specified data item.

Test case

A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program oath or to verify compliance with a specific requirement.

Tool qualification

The process necessary to obtain certification credit for a software tool within the context of a specific airborne system.

Traceability

The evidence of an association between items, such as between process outputs, between an output and its originating process, or between a requirement and its implementation.

Validation

The process of determining that the requirements for a product are correct and complete. Are we building the right aircraft system/function/item? [ARP-4754A]

Verification

The evaluation of an implementation of requirements to determine that they have been met. Did we build the aircraft system/function/item right? [ARP-4754A]

Widget

In computer programming, a widget is an element of a graphical user interface (GUI) that displays an information arrangement changeable by the user, such as a window or a text box. The defining characteristic of a widget is to provide a single interaction point for the direct manipulation of a given kind of data. In other words, widgets are basic visual building blocks which, combined in an application, hold all the data processed by the application and the available interactions on this data. (Source: Wikipedia)

D Software Simulation with SCADE Suite, SCADE Display, and ANSYS Simplorer

The virtual prototyping capabilities of SCADE Suite, SCADE Display and ANSYS Simplorer working together are efficient to rapidly develop virtual prototypes which are essential contributors of the Requirements Validation process, making sure that requirements are understood and correct.

As shown in Figure D.1 and Figure D.2, SCADE Suite and SCADE Display are DO-178B/C qualified Model-Based Development tools that reduce cost, risk, and time-to-certification.

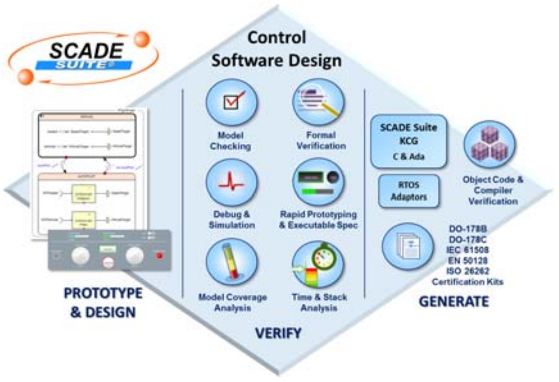


Figure D.1: Software modeling with SCADE Suite

SCADE System is providing a function of synchronization with SCADE Suite and SCADE Display, in order to pass and maintain the detailed architectural artifacts and interfaces elaborated at system level.

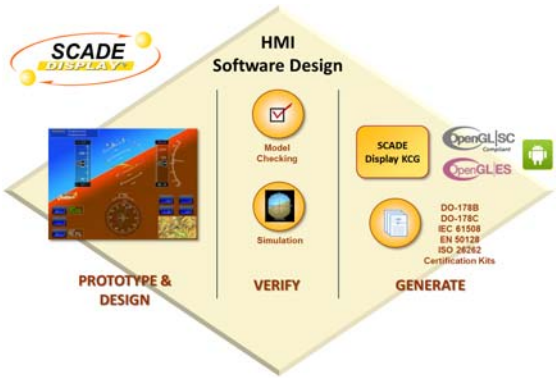


Figure D.2: HMI modeling with SCADE Display

The management of requirements and their traceability links is enabled by SCADE LifeCycle Requirements Management Gateway, as presented in Figure D.3.

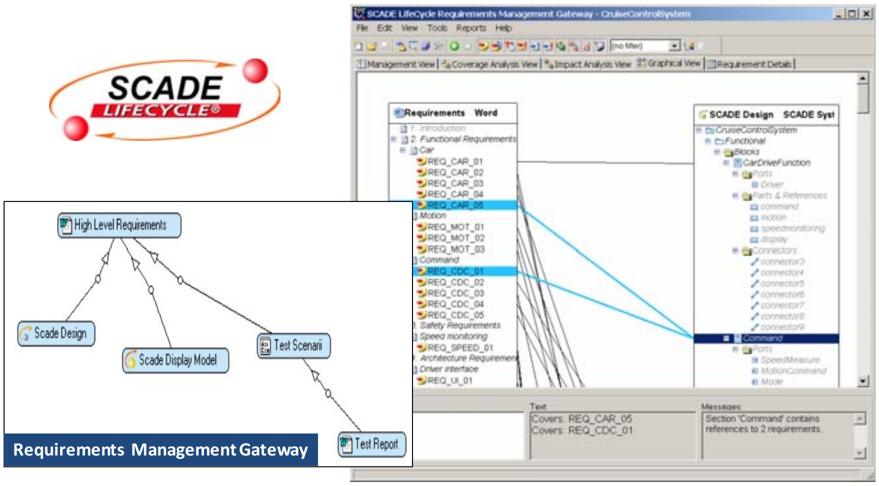


Figure D.3: Managing Requirements with SCADE LifeCycle

The overall ANSYS vision of System Integration and V&V driven by simulation is presented in Figure D.4. From complete Virtual Simulation to Model-in-the-Loop (MIL), Software-in-the-Loop (SIL) and Hardware-in-the-Loop (HIL), the progressive nature of Integration and Verification is accompanied.

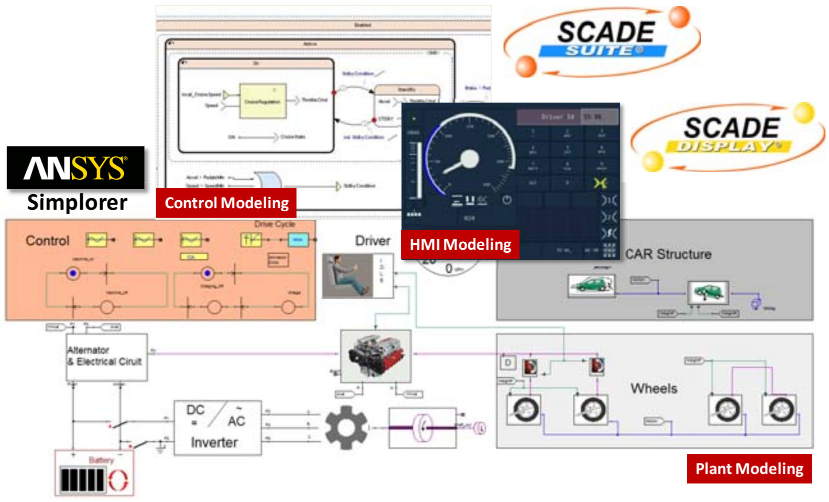


Figure D.4: Conceptual analysis and virtual prototyping with ANSYS tools

Index

Contact Information

Submit questions to Technical Support at
scade-support@esterel-technologies.com

Contact one of our Sales representatives at
scade-sales@esterel-technologies.com

Direct general questions about Esterel Technologies to
scade-info@esterel-technologies.com

Discover the latest news on our products and technology at
http://www.esterel-technologies.com