Studies in Systems, Decision and Control 134

Eugenio Brusa
Ambra Calà
Davide Ferretto

Systems Engineering and Its Application to Industrial Product Development

Springer

Studies in Systems, Decision and Control

Volume 134

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

The series "Studies in Systems, Decision and Control" (SSDC) covers both new developments and advances, as well as the state of the art, in the various areas of broadly perceived systems, decision making and control- quickly, up to date and with a high quality. The intent is to cover the theory, applications, and perspectives on the state of the art and future developments relevant to systems, decision making, control, complex processes and related areas, as embedded in the fields of engineering, computer science, physics, economics, social and life sciences, as well as the paradigms and methodologies behind them. The series contains monographs, textbooks, lecture notes and edited volumes in systems, decision making and control spanning the areas of Cyber-Physical Systems, Autonomous Systems, Sensor Networks, Control Systems, Energy Systems, Automotive Systems, Biological Systems, Vehicular Networking and Connected Vehicles, Aerospace Systems, Automation, Manufacturing, Smart Grids, Nonlinear Systems, Power Systems, Robotics, Social Systems, Economic Systems and other. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution and exposure which enable both a wide and rapid dissemination of research output.

More information about this series at http://www.springer.com/series/13304

Eugenio Brusa · Ambra Calà
Davide Ferretto

Systems Engineering and Its Application to Industrial Product Development

Springer

Eugenio Brusa
Department of Mechanical and Aerospace Engineering
Politecnico di Torino
Turin
Italy

Ambra Calà
Institute of Ergonomics, Manufacturing Systems and Automation
Otto-von-Guericke-Universität
Magdeburg
Germany

and

Siemens AG Corporate Technology
Erlangen
Germany

Davide Ferretto
Department of Mechanical and Aerospace Engineering
Politecnico di Torino
Turin
Italy

ISSN 2198-4182 ISSN 2198-4190 (electronic)
Studies in Systems, Decision and Control
ISBN 978-3-319-71836-1 ISBN 978-3-319-71837-8 (eBook)
https://doi.org/10.1007/978-3-319-71837-8

Library of Congress Control Number: 2017959903

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To all of our Masters,
who allowed us discovering
the amazing world of Science
and Engineering

and

in memory of prof. Sergio Chiesa,
our unforgotten guide along
the paths of the Systems Engineering

Preface

Mastering the complexity of innovative systems currently looks a challenging goal of design and product development as well as embedding a suitable degree of smartness in devices, machines, and equipment to make them able of adapting their operation to variable conditions or effects of a harsh environment. This goal is achieved through a continuous monitoring of the system in service, an effective control of its behavior, and a wide connectivity toward many other systems. Only an effective system design and manufacture, able to cover all the required actions, can assure this kind of assessment overall the life cycle since a very early concept of the product to a full disposal and service.

Complexity makes hard managing the product development, because of the number of functions, subsystems, components, and related interfaces usually involved, like in motor vehicles, robots, railway systems, aircrafts, and spacecrafts as well as in large industrial manufacturing systems or very innovative microsystems and bioinspired devices. A crucial issue in this activity is performing a bright and complete elicitation of requirements, which need to be fully and suitably allocated to the system components, through a clear traceability, especially in systems produced as a result of material processing and assembling of parts. Moreover, the product must fit the requirements associated with some customer needs, innovation targets, and technical standards and be compatible with the manufacturer's capabilities.

As it looks clear from the current state of the art, since several years, the Systems Engineering assures a suitable answer to the needs above mentioned. It provides a methodology to drive the product lifecycle assessment that is implemented through a well-defined process, being based on some specific and graphical languages and even formalized in several tools enabling the required analyses, taking advantage of the capabilities of some dedicated commercial software. Those contents lead to create a platform, consisting of a sort of tools chain, which might be used and shared among different industrial and professional partners to digitalize both the information and even the whole industrial product development, as far as the current strategy referred to as "Industry 4.0/The Factory of the Future" brightly suggests and supports. The so-called *Model-Based Systems Engineering* (MBSE) is

then successfully proposing an effective and modern alternative to the document-based approach, using data models as a main element of the design process. Some technical standards already drive the user in implementing the Systems Engineering, thus leading to develop a systematic approach the design aimed at satisfying the customer needs. Suitable capabilities in the manufactured system are assured by the so-called architectural frameworks, which support the system development and integration.

The Model-Based Systems Engineering allows proceeding with a modeling activity which investigates requirements, behavior, and architecture through a combined operational, functional, and logical analysis, being linked and interoperated with a mathematical and physical modeling, which is typically more known and widely used within the industrial engineering. A full integration of all the activities of the *Product Lifecycle Management* (PLM) is currently going on, to include the system architecture definition and its *Application Lifecycle Management* (ALM) as well as the *Product Data Management* (PDM), i.e., the design activity together with the tasks of production, testing, homologation, and service. A recognized standard certification to qualify the Systems Engineer is even available as the *International Council on Systems Engineering* (INCOSE) provides.

The scenario above described is strongly integrated with the increasing development of both the network and the cyber-physical systems, for a fully distributed connectivity, to be exploited in advanced smart systems and devices as well as in intelligent manufacturing, according to the most recent strategies of innovation as the "Industry 4.0" initiative and the "Lean manufacturing" idea. Simultaneously, the system smartness and connectivity together increase the demand of data transmission and elaboration, thus linking this topic to the technology of big data management, while they benefit of the progress in information technology, through a secure cloud based on the network.

The context just described motivates the fast diffusion of the Model-Based Systems Engineering as a tool for innovating all the production processes. The increasing demand of specialized software and of educational activities as well as the number of workshops and conferences focused on this topic confirm this trend. However, it might be remarked that several contributions to the literature about the Systems Engineering widely grew up during the last years, thus making the Reader sometimes confused, especially when approaching this topic at first.

The Systems Engineering topics are so many that it looks rather difficult mastering its skills, without a preliminary classification of contents. Technical domains involved are mainly those of engineering and computer science, although many other ones play the role of a daily user of this methodology. According to the most recent development of the Systems Engineering, whose typical application fields were the software and electronic systems even for space missions, the current focus consists of several industrial systems, being gradually innovated by introducing the tailored solutions of mechatronics. It is worthy noticing that a significant advancement was introduced between the very early implementation of the Systems Engineering and its recent evolution, since several new applications are focused on the production of systems, which need to be manufactured through a material processing.

Usually, they exhibit some attributes related both to their physical nature and to the functions performed, thus requiring to model both their functional and physical behaviors together. This need is changing the scenario of the typical applications of the Systems Engineering as software design.

This handbook expressively avoids to cover all the typical contents of the specialized literature of the Model-Based Systems Engineering, while is aimed at making easier a first approach to this topic and sharing a preliminary experience performed by the authors within some industrial domains, by proceeding in the modeling activity in a real industrial environment. The main goal is drawing a sort of simple and hopefully clear roadmap in modeling and developing the industrial and material systems and in implementing the Systems Engineering, particularly in the design activity. Therefore, the target audience of this handbook includes professional engineers, scientists, and students dealing with the ALM and the system architecture assessment, more than the PDM or the whole PLM.

The approach followed is that of introducing some examples of implementation of the Systems Engineering, by proceeding step by step from the screening of needs and the elicitation of requirements till a synthesis of the system design. Each action will be referred to the literature, related to the implementation of the *Systems Modeling Language* or *SysML* and to the use of some tools available on market, thus highlighting benefits, drawbacks, and current limitations of some dedicated software or even of some proposed methodologies. Several comments will be provided to describe the troubles shared among some users of the Systems Engineering as they were detected in daily practice by the authors. They wish that this handbook could briefly and gradually provide the Reader with a preliminary guideline to approach professionally the Model-Based Systems Engineering, by understanding its main contents and applying it to the industrial environment. As a desired result, this work might be considered as an integration of some textbooks of Machine Design, and it is aimed at completing the education within Engineering Design or at simply providing a friendly introduction to the Systems Engineering.

Turin, Italy — Eugenio Brusa
Magdeburg/Erlangen, Germany — Ambra Calà
Turin, Italy — Davide Ferretto
December, 2017

Acknowledgements

The authors wish to express their sincere gratitude to all the partners and participants of the ARTEMIS JU Crystal Project (2013–2016), for their contribution in building the knowledge of the applied Systems Engineering, which aimed this work.

In addition, they thank indeed the IBM Corporation, the PTC Corporation, and The Mathworks Inc. for their fruitful support and for the authorization to exploit their tools within the examples described in this handbook.

Eventually, the authors would state that all product names and brands cited in this handbook are property of their respective owners and are used just for identification purposes and to describe the current state of the art. The eventual use of those names, logos, and brands does not imply any explicit endorsement.

Contents

1 Introduction 1
1.1 The Industrial Context 1
1.2 Goals of This Handbook 2
1.3 Test Cases and Implementation of Tools 3
1.4 Structure of the Handbook 5

2 The Systems Engineering 7
2.1 A Definition in a Nutshell 7
2.1.1 Main Goals 8
2.1.2 Four Pillars 11
2.2 Some Historical Notes 13
2.3 A Survey on the Literature About the Systems Engineering . . . 15
2.4 Technical Standards on the Systems Engineering 19
2.5 Software Tools for the Systems Engineering 21
References 22

3 The Methodology of Systems Engineering 25
3.1 Introduction 25
3.1.1 Definitions of System 25
3.1.2 The System Development as an Industrial Product 26
3.2 The Models of the Product Life Cycle 29
3.2.1 The Waterfall Diagram 30
3.2.2 The V-Diagram 30
3.2.3 The Spiral Diagram 33
3.3 The Architecture Frameworks 34
3.3.1 MODAF 35
3.3.2 UAF 37
3.3.3 Framework and Process 39

3.4 The Industrial Implementation of the Methodology 39
3.4.1 Key Issues of the SE Process 42
3.5 Overview on Known Methodologies to Implement the MBSE. . . 46
3.5.1 The INCOSE Object-Oriented Systems Engineering Methodology (OOSEM) 46
3.5.2 The IBM Rational Telelogic Harmony-SE 47
3.5.3 The IBM Rational Unified Process for System Engineering (RUP-SE) 47
3.5.4 The Vitech Model-Based System Engineering (MBSE) 48
3.5.5 The JPL State Analysis (SA). 48
3.5.6 The Object-Process Methodology (OPM) 48
3.5.7 The Architecture Analysis and Design Integrated Approach (ARCADIA) 49
3.5.8 The Systems Modeling Process (SYSMOD). 49
3.5.9 The Alstom ASAP Methodology 49
3.5.10 Synthesis About the Methodologies 49
3.6 A Reference Process: The ISO/IEC 15288 50
3.7 The Engineering Methods 52
3.8 The Languages for Systems Engineering 56
3.9 Unified Modeling Language—UML. 57
3.10 System Modeling Language—SysML 58
3.10.1 Requirements Diagram 59
3.10.2 Behavior Diagram 60
3.10.3 Structure Diagrams 64
3.10.4 Parametric Diagram 66
References 67

4 Systems, Customer Needs and Requirements 69
4.1 A Couple of Examples to Understand 69
4.1.1 Didactic Test Case: A Coiler for Wire Rod Production 70
4.1.2 Industrial Test Case: De-icing or Anti-icing System for a Commercial Aircraft 72
4.2 Implementation of the MBSE 73
4.3 Identification of the Customer Needs 74
4.3.1 Needs Versus Requirements 74
4.3.2 Looking for the Customer Needs 74
4.3.3 A Systematic Approach to the Identification of Needs 75
4.3.4 Source of Needs 77

4.4 The Stakeholders . . . 78
4.4.1 Didactic Test Case: Needs and Stakeholders . . . 79
4.4.2 Industrial Test Case: Needs and Stakeholders . . . 80
4.5 The Role of Requirements in the Product Development . . . 81
4.5.1 Definition of Requirement . . . 82
4.5.2 Classification of Requirements . . . 83
4.5.3 Syntax and Attributes of Requirements . . . 85
4.6 Tools for Writing Requirements . . . 86
4.6.1 Requirements Manager . . . 86
4.6.2 Requirements Quality and Authoring Suites . . . 90
4.7 Requirements Refinement and Assessment . . . 91
4.7.1 Didactic Test Case: Classification and List of Requirements . . . 91
4.7.2 Industrial Test Case: Classification and List of Requirements . . . 104
References . . . 113

5 Operational Analysis . . . 115
5.1 Goals and Tasks . . . 115
5.2 The Operational Analysis Deployed Through the SysML . . . 116
5.3 Implementation and Operational Context . . . 119
5.3.1 Didactic Test Case . . . 119
5.3.2 Industrial Test Case . . . 126
5.4 Requirements Derivation in Operational Analysis . . . 139
5.5 Synthesis of the Operational Analysis for Both the Test Cases . . . 143

6 Functional Analysis . . . 147
6.1 Introduction . . . 147
6.2 Handoff Between Operational and Functional Analyses . . . 149
6.3 Implementation of Functional Analysis Through the SysML . . . 150
6.4 Requirements Derivation, Traceability and Allocation . . . 155
6.5 Results and Outputs for the Logical Analysis . . . 157
6.6 Implementation: Deriving the Functional Architecture . . . 158
6.6.1 Didactic Test Case . . . 159
6.6.2 Industrial Test Case . . . 166
6.6.3 Comparison Between Use Case and Black-Box Based Approaches . . . 184
6.7 Results and Final Remarks About the Functional Analysis . . . 190
References . . . 191

7 Logical Analysis . . . 193
7.1 Meaning of the Logical Analysis . . . 193
7.2 Handoff Between Functional and Logical Analysis . . . 194

7.3 Implementation of the Logical Analysis Through the SysML ... 195
7.4 Requirements Satisfaction and Architecture Allocation 199
7.5 Towards the Next Phase 200
7.6 Implementation and System Logical Architecture 201
7.6.1 Didactic Test Case 201
7.6.2 Industrial Test Case 204
7.7 Requirements Traceability in the Logical Analysis 219
7.8 Results and Final Considerations About the Logical Analysis ... 222

8 Physical Analysis 225
8.1 Introduction 225
8.2 Handoff Between Logical and Physical Analyses 226
8.3 Formalisms and Models of the Physical Analysis 227
8.4 Requirements Allocation and Verification 229
8.5 Expected Results and Final Remarks 230
8.6 Implementation of the Physical Analysis of Complex Systems 231
8.6.1 Didactic Test Case 231
8.6.2 Industrial Test Case 244
8.7 Results and Final Considerations About the Physical Analysis ... 269
References 270

9 Heterogeneous Simulation 271
9.1 Introduction 271
9.2 Strategies of Model's Integration Within the Heterogeneous Simulation 272
9.3 Example of Interoperability Standard: The Functional Mock-up Interface 277
9.4 Implementation of the Heterogeneous Simulation in the Ice Protection System Case Study 279
9.4.1 Models for the Ice Protection System Scenario 279
9.4.2 Simulation Results and Final Run 283
9.5 Traceability and Future Evolution of the Interoperability Within Large Toolchains 285
References 287

10 System Verification and Validation (V&V) 289
10.1 Introduction 289
10.2 The Best "V&V" Process 291
10.3 Verification, Validation and Accreditation (V&V, VV&A) 292
10.4 Software and Hardware 293
10.4.1 V&V in Software Engineering 294
10.4.2 V&V in Hardware Engineering 297
10.5 A Methodological Approach to the Industrial Product V&V ... 299

10.5.1 Practical Issues in Product Development and Relation with V&V . . . 299
10.5.2 Workflow of V&V . . . 301
10.5.3 Design Objectives . . . 303
10.5.4 Smartness and Smart-Nect-Ness . . . 304
10.5.5 DT&E and OT&E for the Industrial Product . . . 305
10.6 The Role of RAMS in V&V . . . 307
10.6.1 Reliability . . . 307
10.6.2 Maintainability . . . 308
10.6.3 Availability . . . 309
10.6.4 FMEA and FTA . . . 309
10.6.5 Dysfunctional Analysis . . . 310
10.6.6 Integration of RAMS and Numerical Simulation . . . 312
10.7 V&V Peculiarities of the Proposed Test Cases . . . 313
10.7.1 Didactic Test Case: V&V Issues . . . 313
10.7.2 Industrial Test Case: The RAMS Analysis . . . 320
References . . . 325

11 Systems Engineering and Product Lifecycle Management (PLM) . . . 327
11.1 The Big Picture . . . 327
11.2 The Configuration Control Management . . . 328
11.3 The Platform Building . . . 330
11.3.1 The PLM Collaboration Model . . . 331
11.3.2 The PLM Functional View . . . 331
11.3.3 The PLM Data Model Analysis and the Tool Chain . . . 332
11.4 The Tools Integration and Interoperation . . . 335
11.5 The Configuration Control Action . . . 337
11.6 Integration Between Analyses Within the Tool Chain . . . 339
11.6.1 Integration Between Design and RAMS . . . 339
11.6.2 Integrated Analysis . . . 340

12 Conclusion . . . 343

Index . . . 347

Acronyms and Symbols

AAF	Automotive Architecture Framework
AD	Activity Diagram
ADS	Air Data System
ADT	Administrative Delay Time
AF-EAF	Air Force Enterprise Architecture Framework
AFIoT	Architecture Framework for the Internet of Things
ALM	Application Lifecycle Management
ARCADIA	ARChitecture Analysis and Design Integrated Approach
ASAP	(ALSTOM) Advanced System Architect Program
ASME	American Society of Mechanical Engineering
BDD	Block Definition Diagram
BOM	Bill Of Materials
CA	Corrective Actions
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CBM	Condition-Based Maintenance
CCM	Configuration Control Management
CDR	Critical Design Review
CFD	Computational Fluid Dynamics
CN	Customer Need
COTS	Commercial Off The Shelf
CRYSTAL	CRitical sYSTem engineering AcceLeration
CS	Co-Simulation
CT	Cycle Time
DDV	Dual Distribution Valve
DNG	Doors Next Generation
DoD	(US) Department of Defense
DoDAF	(US) Department of Defense Architecture Framework
DR	Discrepancy Report

DT&E	Development, Test, and Evaluation
EASA	European Aviation Safety Agency
ECU	Electronic Control Unit
ESAAF	European Space Agency Architecture Framework
FBS	Functional Breakdown Structure
FHA	Functional Hazard Assessment
FI	Final Integration
FMEA	Failure Mode and Effects Analysis
FMECA	Failure Mode, Effects, and Criticality Analysis
FMI	Functional Mock-up Interface
FMS	Flight Management System
FMU	Functional Mock-up Unit
FTA	Fault Tree Analysis
IBD	Internal Block Diagram
ICBM	InterContinental Ballistic Missile
IML	Industrial Modeling Language
INCOSE	International Council On Systems Engineering
ID	IDentification code
I/O	Input/Output
IPS	Ice Protection System
ISA	International Standard Atmosphere
IT	Information Technology
KPI	Key Performance Indicator
LBS	Logical Breakdown Structure
LDT	Logistic Delay Time
LML	Lifecycle Modeling Language
LWC	Liquid Water Content
M*	Mean active maintenance time
MBSE	Model-based Systems Engineering
MDS	Model-Driven System
MDT	Maintenance Down Time
ME	Model Exchange
MODAF	(UK) Ministry Of Defense Architecture Framework
MOE	Measures Of Effectiveness
MOP	Measures Of Performance
MOS	Measure Of Suitability
MTR	Main Technical Review
MTBF	Mean Time Between Failures
MTBM	Mean Time Between Maintenance
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
MVD	Mean Volume Diameter
NAF	NATO systems Architecture Framework

OEI	One Engine Inoperative
OMG®	Object Management Group®
OOSEM	(INCOSE) Object-Oriented Systems Engineering Methodology
OPD	Object-Process Diagram
OPL	Object-Process Language
OPM	Object-Process Methodology
OSLC	Open Services for Lifecycle Collaboration
OT&E	Operational Test and Evaluation
PBS	Product Breakdown Structure
PCA	Product funCtionAlity
PD	Package Diagram
PDM	Product Data Management
PLM	Product Lifecycle Management
RAMS	Reliability, Availability, Maintainability, and Safety
RAO	Reliability Analysis Objective
RAT	Requirements Analysis Tool
RCD	Request, Confirm, Display
RD	Requirements Diagram
RMT	Requirements Management Tool
PDM	Product Data Management
PLM	Product Lifecycle Management
RAMS	Reliability, Availability, Maintainability, and Safety
RUP-SE	(IBM) Rational Unified Process for System Engineering
SAGE	Semi-Automatic Ground Environment
SA	(JPL) State Analysis
SAT	Static Air Temperature
SCA	System funCtionAlity
SCD	System Context Diagram
SD	Sequence Diagram
SDL	System Definition Language
SE	Systems Engineering
SEBoK	Systems Engineering Book of Knowledge
SMART	Specific, Measurable, Achievable, Relevant, and Traceable
SMD	State Machine Diagram
SoI	System of Interest
SPS	System Performance Specification
SR	System Requirement
SRS	System Requirements Specification
SVT	System Verification Test
SysML	Systems Modeling Language
SYSMOD	SYStems MODeling process
TOGAF	The Open Group Architecture Framework
TPM	Technical Performance Measurements

TR	TRaceability audit
TRR	Test Readiness Review
UAF	Unified Architectural Framework
UML	Unified Modeling Language
UCD	Use Case Diagram
V&V	Verification and Validation
VLBJ	Very Light Business Jet

Chapter 1
Introduction

Abstract The literature about the Systems Engineering are so wide that one might feel lost in reading. Realizing the context where this methodology is introduced and currently refined might help in mastering its main contents. This introduction briefly describes the approach proposed in this handbook to introduce the topic, through a linear rationale, developed through the chapters, and by means of a couple of examples, aimed at simplifying the understanding.

1.1 The Industrial Context

Designing a product which exhibits a large number of capabilities and skills definitely requires to conceive a real *system*, meant as an assembly of different subsystems, components and parts, highly integrated and fully interacting each other through several kinds of interfaces. Functions performed by the system are usually the result of an ordered sequence of activities made by each component or subsystem, or even by more than one, to achieve a defined goal. Nowadays the number of functions exploited in a system might be fairly large. This leads to involve many components thus needing a number of interfaces.

As a matter of facts, a growing up *complexity* related to the design, production and management in service of systems has to be faced and managed. Strictly speaking, the designer should conceive in details the expected behavior of the system and its architecture, to fulfill a clear and complete list of requirements, by brightly mapping the associations among *requirements*, *functions* and material *components* to assure the highest simplicity, safety, reliability and maintainability, with the lowest cost, the highest quality and the minimum number of parts interfaced. Key issues of this activity are the elicitation of requirements, the selection of the most suitable technology for the identified tasks, the *trade-off* among some candidate architectures and the early prediction of suitable verification and validation processes, usually based on testing and prototyping aimed at system homologation before delivery. Design must fit all of *customer needs* compatible with *technical requirements* and deal with the safety, manufacturing, maintenance

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_1

and service issues, through a consistent and predictive analysis of cost, risk and quality. Moreover, a straight coherence between the design and production of the system development is required to achieve the desired result.

Engineering the systems is definitely nothing new, but nowadays it looks extremely hard in safety critical systems, where the complexity above mentioned plays a relevant role. System design can exploit a long tradition, as documented in the technical literature, within the available technical standards or guidelines and even in the daily practice of several companies, which belong to a wide range of technical and industrial domains. Some of those companies are specialized within several methodologies of systems development and integration, like the aerospace and nuclear engineering, for instance. However, it is true that the early stage of systems definition, including the design requirements and conceptual activities, was based for a long time on few documents, being somehow separated from the models of the detailed design, where size, configuration and integration tasks were performed. Practically speaking, requirements, functional and numerical analyses are often performed by different people, generally through different tools and data are stored in some separated repositories, thus being only partially shared among those persons. This turns out into a severe limitation in innovation, efficiency and cost reduction.

1.2 Goals of This Handbook

The so-called Systems Engineering allows facing several issues related to the product development all along its lifecycle, when its complexity is high. However, a wide literature describes its contents, but a straight implementation might be difficult for a number of reasons.

Systems currently produced belong to several fields of technology. A preliminary distinction can be considered between *software* and *hardware*, e.g. industrial systems like electronic devices, mechanical and mechatronic systems. These are real objects coming from a material processing, being assembled and integrated. Rules and criteria for design and production are therefore related to the manufacturing process. Moreover, it might be immediately noticed that a system can be focused on the management of either *data* (generation, transmission, reception, elaboration and storage), thus leading to the treatment of *signal*, or *energy* (generation, transmission, conversion and storage), and dealing with different kinds of *power*. The first option is typical of the software engineering and computer science, where data management is a main goal of the system operation. Both of the above mentioned options are quite typical of electronic, mechatronic and mechanical hardware and their related technical domains. Power management, in particular, is a key issue for motion, dynamics, control and to produce work.

Realizing that the above mentioned differences hold allows the Reader easily understanding some current limitations in language, tools and software proposed for the Model Based Systems Engineering, which was originally conceived for the

software engineering more than for other industrial products, even quite popular like the aerospace, automotive, manufacturing and other specialized sectors. In following sections this issue will be clarified and pointed out in several steps.

To better entering this topic it may be immediately perceived that the literature develop several items of interest under the common title of Systems Engineering, strictly related to its implementation. First at all the approach and the related *methodology* has to be mastered, but even the *languages* (as the Unified Modeling Language or UML, the Systems Modeling Language or SysML or newer ones as the Industrial Modeling Language or IML, which has been more recently proposed). However, the language has to be distinguished from the *tools* and the *software* expressively developed to implement the approach, which are even related to the *method* and the *process* applied to develop the *digital models* of the product itself (or, sometimes, of the whole lifecycle development). In some cases it might be realized that the methodology looks mature for some foreseen applications, while some tools or software need to be further assessed. Therefore, a certain effort will be spent in following sections to make the Reader aware about the different contents of this subject on which several contributions of the literature are focused.

By the way the power of the Systems Engineering is that it demonstrated to be suitably applicable to several industrial domains with high level of confidence and few differences in some tailored tasks, basically following a common *generalized, intrinsically systematic and effective* approach. Moreover, since longtime, the design activity was performed starting from a very preliminary concept of product, often based on documents, while approaches like the Model Based Systems Engineering provide the tools to transform those documents into models, which are currently *digitalized* and *linked* to the numerical analyses, used in an advanced step of product development. Main benefits of this approach are the *reusability* in different projects, the *traceability* of each requirement, being allocated to functions and system components as well as *the automatic documentation* of the whole product development activity. Understanding those issues and how they are practically assured in operation by using the tools of the System Engineering is a key goal of this book.

A final critical issue is the *standardization* of the Systems Engineering process and tools. This is a priority of the approach to assure a common definition of the so-called *ontologies*, in several technical domains, and a full *interoperability* of software tools, eventually through some dedicated connectors. This topic will be herein faced and detailed as far as the current state-of-art allows.

1.3 Test Cases and Implementation of Tools

Two practical test cases will be proposed to show the Reader how all of those issues may be found and faced as soon as a real implementation of the Systems Engineering is performed. To allow proceeding step-by-step, two main levels will be considered. A sort of simplified and somehow didactic example consists of the

description of a rotor on active magnetic suspension either used to coil the wire rope in steelmaking, as it was really proposed to a manufacturer, or to support a flywheel energy storage system. In this case models will be intentionally quite simple to preliminary introduce some topics and procedures, although they reflect a real industrial application with all of relevant implications and practical problems. However, to give an impression of some critical issues in implementing the Systems Engineering in a real complex system, as an aircraft, the case of an ice protection system will be developed, as it is currently carried out by the manufacturers. In this case the depth of information shall be larger, to show some crucial decisions made to integrate methods and processes of different industrial partners and suppliers.

The didactic test case looks interesting to investigate a mechatronic application as the control of a spoiling system for the steel wide rod connected to a rolling mill. This example includes several issues related to the active control of structural systems, combined with the connection to a complex system like the steelmaking plant is. Moreover, some key issues of the smartness of new systems will be introduced and discussed as well as the relevant item of safety in industrial systems, which requires suitable actions of alarm and warning. Problems related to the hierarchy of control between master and slave systems or even between the operator and the system itself will be briefly explored.

The example concerning an ice protection system for a civil aircraft for passengers' transportation provides an overview on the integration of a mechanical subsystem within a main system, being operated in presence of quite variable environmental conditions, as are those related to the atmospheric flight. In this case the tight collaboration between functional and physical models will be investigated as well as their intrinsic interoperability.

Both those examples will be developed by resorting to some software tools available on the market and currently used in the Model Based Systems Engineering practices for the implementation within some industrial domains. Thanks to the availability of IBM and PTC who kindly authorized the use of their software tools for this publication, it could be possible including some results of the development of the test cases through the IBM Rational Doors® and Rhapsody® and the PTC Integrity Modeler® (formerly Artisan Studio® by Atego) to show the Reader the practical result of such activity. The above mentioned examples will be used to explore some interoperability issues, to connect those tools to other ones, like the MATLAB®, Simulink® or Modelica®, and to perform a preliminary dynamic analysis. The concept and the practical development of the so-called heterogeneous simulation will be herein analyzed. Moreover, both the examples show a high safety critical nature, to be widely and carefully considered within the product development as it will be herein discussed.

1.4 Structure of the Handbook

To reach the above mentioned goals, chapters and sections of this handbook will be mainly focused on a practical investigation of methods, processes, tools and criteria applied to:

- The elicitation, allocation and traceability of *Requirements*.
- The development of the so-called *Functional modeling*, which is aimed at including the operational, functional and logical analyses.
- The development of so-called *Physical modeling*, based on a mathematical modeling of systems and a numerical analysis, providing a simulation environment where the system behavior is investigated.
- The overall *interoperability* of those models, with a overview on heterogeneous simulation techniques.
- A preliminary description of the *verification* and *validation* of the systems.
- Some outlines about the strategic issues related to the *integration* between design and production, namely *Application Lifecycle Management* (ALM) and *Product Lifecycle Management* (PLM).
- A brief highlight on the *Configuration Change Management* as it is currently implemented through the available tools.

According to the Authors, this path might assure a gradual exploration of the proposed topic, being quite linear to be followed step by step, within the limitations of the very preliminary screening of this subject performed by this handbook.

Chapter 2
The Systems Engineering

Abstract Many definitions of the Systems Engineering are proposed, and analyzing some differences between them help to highlight contents and open issues, as they are described in this chapter. Nevertheless, a short historical outline could be helpful in appreciating some characteristics of this methodology, which are even more detailed by a wide literature, herein briefly described. Furthermore, an overview of technical standards dealing with the Systems Engineering is added, to define a roadmap for a deeper education in this field.

2.1 A Definition in a Nutshell

If one reads the definition proposed by the International Council on Systems Engineering (INCOSE), this interdisciplinary approach is described as "*a mean to enable the realization of successful systems*" and "*focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem*". Moreover "*it integrates all the disciplines and specially groups into a team effort forming a structured development process that proceeds from concept to production to operation*". It "*considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs*" (Walden, 2015).

The above and precise definition probably is the fastest way to describe at least the most important contents of the Systems Engineering (in the following simply SE), which doesn't substitute all the techniques of engineering the systems in a more general meaning, but provides new means to do this by following a more systematic approach. It can be immediately realized that some highlights are characteristic of the proposed approach and even cited by the above definition. Object of design must be a *system*, being naturally based on the interaction among different *components*, to be *assembled* together and *connected*. Success of product developed is directly related to the *customer satisfaction*, although suitable *metrics* to measure that satisfaction have to be defined. Product must provide only *functions*

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_2

strictly *required*, since its conceptual design, i.e. very early in the whole development path. A clear sequence of operations in development is suggested, thus starting from requirements, to define the design contents till the synthesis, being a first intermediate goal of the whole activity. *Validation* is then proposed as a necessary step to investigate how much the product fits the customer needs. This interpretation leads to the description of a *process*, from concept to operation, being referred to as *structured*, i.e. fairly well defined to be linear, repeatable and effective, and strictly oriented to *quality* and to the *user*. Moreover, the two main worlds of *technology* and *business* are both considered, thus suggesting the need for a clear *modeling* the technical contents as well as the business foreseen for that product.

Despite of the brightness of sentences, implementation of the SE is never so straight, especially in the hardware domain. It might happen that the real contents of those sentences are poorly understood, if the Reader doesn't know some goals which motivated the creation of the SE and some principal pillars of this approach. Screening those two issues may help in implementing it more easily.

2.1.1 Main Goals

Nowadays systems conceiving, designing, developing and integrating are all key strategic goals of the product development, especially in industrial manufacturing, when a direct material processing technology is exploited. Practically speaking, some needs could be detected.

Suitable tools for handling complexity. Product is today a *system of subsystems*, *components and parts*, and often embeds some kind of *smartness* and some *capability of communicating*. Consequently the number of functions exploited is increasing, together with the number of interfaces. As an example, one may think about electronic control units and sensors, which are often applied to mechanical and electromechanical elements, thus creating a full mechatronic product. It has to be manufactured through the contribution of several technical competences and a suitably design of each energy conversion exploited by the system has to be performed. This leads to a certain *complexity*, poorly manageable through the tools widely used in the past.

Traceability over the whole lifecycle. Nature of new products intrinsically requires a longer support after delivery, to perform monitoring, maintenance and control, and to assure a regular service. Cost might grow up fairly fast for the manufacturer, if the whole *Product Lifecycle Development* is never completely considered to clearly foresee all the actions due after market. Only a clear *traceability* in the development (from requirements to part numbers) of both the product elements and functions, seen as a direct result of the allocation of requirements, could allow covering the system lifecycle.

Models creation, digitalization, reusability and automatic documentation. A deep, complete and shared *documentation* is needed to make aware all the users and the operators about the details of the system operation, maintenance and even

failure. This task is required to consolidate the "know-how" of the manufacturer, managing the turn-over of people, but even to support the user in service. Moreover, documentation and *data* should be recovered, collected and shared through a unique data management system, with dedicated repositories, controlled access to data bases under high security standards. This requires a preliminary digitalization of the information, then a network of connections, as well as an efficient data elaboration. *Models*, instead of pure documents, are easily reused in new products or versions of the same system, shared and stored as well as embedded in different documents to be automatically produced. These documents and models can be shared among operators of the same company when developing, but even together with suppliers and customers to enhance the activities of acquisition together with service and maintenance in operation.

Reducing costs of the system development, human mistakes, and late re-engineering activity. The SE is so widely growing up within the industrial world because of its effectiveness in handling some multidisciplinary engineering projects, by decomposing the complexity of systems, but even thanks to the so-called "left shift" in the resources consumption over the lifecycle.

As Fig. 2.1 shows, if one describes the resources usually spent for each step of the project to develop the system, conceptual stage covers a small percentage of the total cost, while tests, production and final assessment to disposal are fairly expensive. By converse, the SE tends to improve the efficiency of the early steps of development by increasing significantly the resources applied, by already considering for each element of the system the required test, production and assessment needs, to prevent the risk of a late detection of defects which might require a complete re-engineering, thus causing a demand of a huge amount of resources, being several times those already spent at that time, during the development.

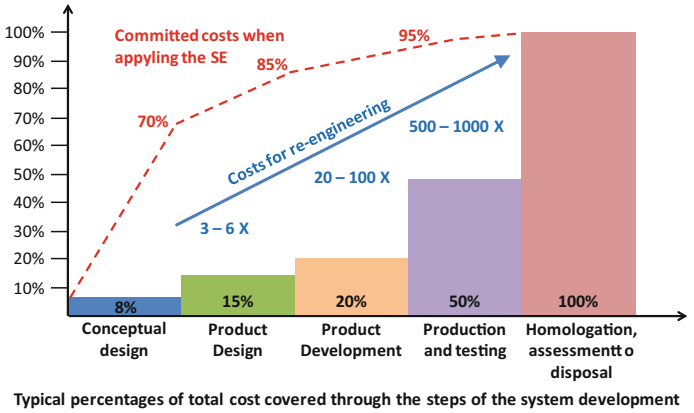


Fig. 2.1 Comparison between the usual distributions of the percentage of total costcovered for the system development and that proposed by the SE (Defense, 2001)

In this description a peak in costs occurs earlier in the project development and in time, but several costs usually foreseen for the late steps are anticipated. Therefore the peak is somehow shifted to the left side along the time axis, thus motivating the typical expression above mentioned. It can be remarked that a deeper conceptual activity might help in discovering problems before that any prototype is manufactured. Moreover, since the largest amount of money is spent in the early part of development, decisions are driven by a number of analyses, and, usually, lead to avoid a trial and error approach, typically more expensive and randomly effective.

Those needs actually are strictly related to some typical goals, as the design for safety and the quality in production, but they could be even associated to a bright *process of change configuration management* and to the two activities of *verificating the requirements* and *validating the system*, respectively.

In practice, the SE helps in *decomposing the complexity* of the problem in several issues and somehow in different *views*, thus exciting a careful activity of concept design and a complete elicitation of requirements. In addition, integrating all the models which might be used to describe and analyze the system in a same *virtual platform* improves their correlation and helps to create a holistic perception of the whole system. Models allow defining *a rationale* to be reused and assessed through different projects and driving the product designer through several tasks, thus making easier to account for all the required steps.

Some problems of *intrinsic safety* of product might arise in presence of a number of subsystems and components, if failure modes, causes and effects cannot be effectively predicted. Moreover, detecting clearly and fast any correlation between a failure occurring into a designed part and a requirement which motivated its inclusion within the overall system architecture is often rather difficult. Finally system prognostics, diagnostics and reliability can be effectively provided only through a clear *monitoring of events*. Therefore, the SE tools help in the safety analysis development, by showing how requirements, functions, components and parts are linked each other within the system architecture, thus making the *traceability* of requirements clear. Simultaneously, some issues of the so-called *functional modeling* allows defining actions, actors and relations, thus allowing to describe failure modes through a simple negation of those items. Designing an effective monitoring system to be applied to the product is never trivial and the availability of operational, functional and architectural analyses provided by the SE might enhance this skill as well as a more detailed numerical simulation.

It is important to remark that the SE somehow fills the gap between the overall "Project management", dealing with the organization and the development of the project, being interpreted as the set of activities leading to the realization of products, and the "Design" which is strictly, and almost only, related to the definition of the product elements through some dedicated and focused actions. It defines the *needs* to be satisfied, but even performs the technical *trade-off* among some candidate layouts, dealing with the whole system as well as with the smallest components, providing their *integration*, first through the design stage, then thanks to the manufacturing process and even after.

2.1.2 Four Pillars

As soon as the Reader will briefly consider the wide literature already available about the SE, probably an immediate feeling of shipwreck could arise because a number of books, guidelines, handbooks and also some Standards could be helpful, but very often they are focused on extremely different subjects. As usual in mechatronics, it might be remarked that SE involves many disciplines and each author is prone to explore the contents more familiar, focusing on a portion of the whole topic. Roughly speaking, it might be realized that four are the main areas of interest usually explored in the literature: *methodology*, *language*, *tools* and *data management*.

The methodology. Describing the methodology proposed by the SE looks often rather difficult without a bright roadmap as it was proposed by Estefan (2008). The methodology applied by the SE is based on *digital models*, easily shared within a community of users and easily stored by a data management system. This main property leads to define the SE methodology as a "model-based" approach or *Model-Based Systems Engineering* (MBSE). Two main models are developed, the first one drives the system engineer through the product development and concerns the *product life cycle*, while the second one describes both qualitatively (only by logic) and quantitatively (by numbers) the *product* itself, being the complex *system*.

Several reference models were proposed in the literature to define the *product life cycle development*. They are basically an ordered and rational list of interlinked steps to be performed in engineering the system. It might look as a quite obvious task of the whole design activity, but defining both the goals and the sequence of actions to be performed, as well as how they are mutually linked or even connected to the manufacturing process is crucial. Facing the problem of designing the system by either considering first the whole product and its requirements, then by decomposing it in components through a sort of *top-down approach*, or proceeding from the single component to the whole system through a *bottom-up approach*, was a matter of discussion. How the ALM and PLM actions could be suitably inter-related step by step, thus creating the traceability above mentioned was even deeply investigated.

As a relevant result, the MBSE currently provides a sort of checklist for the product developer and helps in handling some practical difficulties associated to the different levels of the system architecture and to its complexity. Particularly, on this assessment of the methodology two main issues were defined as the *process* itself, being focused upon the action to be performed in a due sequence through a logical path, and the *method*. Practically speaking the process defines *what* the system engineer must do and the method *how* to proceed.

The implementation of the process through the method includes the real product development whose object is the *system*. Therefore the modeling activity directly applies to it. Basically, the system functionality is first analyzed, through the *functional modeling*, to allow the allocation of requirements, then the performance of the system is predicted by the *physical modeling*. If the physical modeling is

based on some mathematical description of the system behavior and its language coincides with mathematics, the functional modeling for long time did not have an effective mean of expression, like equations are for numerical models. Nowadays some languages support the modeling of functions, behavior and structural layout of the system, which can be represented before than it could be detailed, for instance into a technical drawing. Moreover, to create a functional model, some standard tools are required, which might be understood by all the users involved in the product development.

The tools. Actually two kinds of tools allow implementing the MBSE approach, consisting in some *theoretical tools*, like several typical *diagrams* and some *engineering methods*, and in the *software* providing a digital and virtual environment where the MBSE methodology can be implemented through some standard language.

The language. Dealing with a complex system inherently means often involving many technical competencies. To establish a fruitful cooperation, a common terminology, or better a tool which might be easily understood and applied by all the developers of a same task, is required. To reach this goal, a fairly intuitive language based on a standard set of symbols and items could be found. In case of the SE, some examples are available. A prevalent tradition comes from the *Unified Modeling Language (UML)*, which was then adapted and enriched for this purpose as the *System Modeling Language (SysML)*, but even other ones are currently developed to overcome some limitations of those languages, particularly when applied to the industrial product manufacturing.

The data management. The *information* and the *data storage* are both critical issues of the SE. They require a careful design of a common environment, including hardware and software, to which all the authorized users can access, allowing the tools being interoperated, i.e. data can be automatically transferred from one tool to another one, without a direct action of the user. This common environment, to be referred as *platform*, should allow producing, storing, sharing and elaborating all the required data. The platform must be compatible with the needs of different users and operators and with the software products used by all of them, to perform the modeling activity and to develop some related services. The platform needs a connection through the network, being based on a data bus, a cloud or other web services. The so-called *interoperability of software* tools and the *cyber security* are crucial issues of the platform building activity.

Those four pillars, as they were here above briefly defined, actually allow interpreting the contents of several contributions proposed in the literature and even some research activities currently performed within the MBSE. A clear rupture with the past approaches, based on documents to support the system development, consists in the model based approach, even fully digitalized nowadays. This is a distinctive characteristic of the MBSE and motivates its current wide application.

By converse the standardization of *methods*, *languages* and *tools* is still under assessment, although it looks a relevant goal for a complete diffusion and application of this approach. The Reader shall surely appreciate in this handbook how much *tools* and *methods* of the MBSE depend on the software currently available

on the market, sold by several vendors. That's the actual challenging issue for a complete assessment of the SE. It is known that the different features exploited within the software products lead to a specific implementation of the methods and of the theoretical tools of the SE, according to some vendors' interpretation. A good standardization was already found in the system procurement, rather than in the product development, being the core in transportation and defense domains. In those cases, the product development is actually based on a complete definition of the *system capabilities*, described through a set of standards *views*, which are built up by resorting to the so-called *architecture frameworks*. These are another important component of the MBSE.

Next chapters will analyze more in details the above mentioned concepts to present the whole MBSE approach. Moreover, those key issues look useful for an easier interpretation of the existing literature. Some main references will be herein considered to start, although many others are available and published, whilst this short handbook is composed.

2.2 Some Historical Notes

The Systems Engineering, as is currently known by professionals and scientists, represents a convergence of several direct experiences performed by a number of system developers in several technical domains. Therefore the roots of the SE should be found definitely before than several theories and public references appeared since the latest 90s. For sure, the technical challenge of designing innovative and complex systems was typical of the fascinating story of the conquest of space. This was related to a fast growth of the space systems as well as of electronics, computer science and communications. Therefore, it does not surprise that a popular reference to learn the methodology of SE is the dedicated *NASA handbook* (NASA, 2007), but the real birth of the SE dates since the World War II. At that time clearly arose the need of managing some rather complex systems like aircrafts, tanks, but even the army logistics. The SE was evenly relevant for the operations research and decision analysis. During the 50s some preliminary references appeared and were applied to space programs and intercontinental ballistic weapons, thus promoting some methodologies to develop the systems (*Systems Engineering*) and to assure the full accomplishment of the goals of technical projects (*Project Management*), together with an effective prediction of risk (*Risk Analysis and Management*). Nevertheless, over the years, a bright necessity of relating those three main competences was satisfied by creating a suitable correlation through the SE.

After a first appearance of the *concept of system* in engineering within electric power distribution and telephones, the SE began to be conceived as a multidisciplinary approach in the early 50s of twentieth century. Basically, three main topics were related to the SE as the *general theory of systems*, the *cybernetics* and the *operations research* (Arrichiello, 2014).

In 1950 Ludwig von Bertalanffy identified in the "General System Theory" (Bertalanffy, 1972) the need of investigating biological and technical systems not only in terms of assembly of parts, but even for their mutual interactions as well as those with the environment outside their neighboroughs, introducing the new concept of *open system*. This immediately linked the *theory of systems* to the growing up fields of the *information technology* and of the *automatic control*.

In the same time, Norbert Wiener introduced the concept of *feedback*, seen as a possibility of adjusting the performance of systems by knowing their past experience, applying this idea to a new science that he called *cybernetics*, i.e. the control and communication in animals and machines (Wiener, 1948). An additional interesting issue of his theory was the introduction of the *system behavior* as a fundamental goal of the investigation, in all the fields of science and technology.

Operations research added a substantial contribution to this new discipline, during the World War II, as a mean to support *decisions*, then as a branch of mathematics and statistics, although it involved physicians, engineers and other scientists. That's why Weaver (1948) observed that when several members of diverse groups work together and create a unit, products are definitely greater than those obtained by a sum of parts and focused the attention upon the *system integration* and to the *holistic view* of product development.

The above mentioned origins found a favorable context in the Cold World War to grow up in projects like the *Intercontinental Ballistic Missile* (ICBM) and the *Semi-Automatic Ground Environment* (SAGE). The scale of those systems, the number of competences involved and the very challenging performances required made this context a perfect test rig for the SE tools and methods. Few years later the conquest of space and some programs like the *Apollo* missions finally brought to a formal definition of the SE process.

This growth of the tools of SE motivates some typical characteristics, which need today to be somehow updated and adapted to some other contexts like transports, health care, smart manufacturing and mechatronics. To catch immediately the substantial and powerful contents of the SE, it can be said that it smartly puts in evidence the *three basic ingredients* for an effective system design, described by the Royal Academy of Engineering in 1999 as follows: "every *design process* should begin from a *clear defined need*, should be performed through a *suitable vision* aimed to give an *effective response* to the need and the delivered product should meet the *expectations of the customer*, expressed by that need". As it will be herein explained, this simple process holds when the designer is aware about the *customer needs*, plays a deep attention to *requirements* and to a precise *validation of product* at the end of *process*. The SE helps the designer to consider the whole system, its neighboroughs, the *stakeholders* and all the *interactions* among them as well as among the system components *and interfaces*. In this purpose some additional benefits are provided.

The SE approach teaches to follow a structured rationale, to identify a hierarchy of items, to realize how the system behaves, to perceive the need of defining some suitable metrics, to *measure the performance* and to *model and simulate* that

behavior, even through the merge of different approaches and tools might make the *simulation heterogeneous*. The literature, the tools and the software basically give this kind of information and help the user to follow the approach.

2.3 A Survey on the Literature About the Systems Engineering

A key starting point to understand the SE is a brief survey upon some popular textbooks, tools and standards, which nowadays constitute an appreciated reference for the different topics above described.

A first issue concerns the *approach* and the *processes* applied. As a matter of facts the most recent guideline about the SE including a fairly deep information is:

- David Walden, Garry Roedler, Kevin Forsberg, Douglas Hamelin, Thomas Shortell—*Systems Engineering Handbook of INCOSE*, 4th Ed., John Wiley and Sons, 2015

where a complete description of the *approach*, of its straight implementation through some standard *views* and *tools*, respectively, is provided, after a deep assessment performed by authors and contributors, over the years. It is significant even for a professional certification, through the INCOSE, being a worldwide recognized community for the systems engineers (see for instance the website www.incose.org and the activities promoted by the sections).

This reference obviously covers a wide range of topics related to the SE, although a relevant support for the *application to the aerospace engineering* is given by:

- The *NASA Systems Engineering Handbook*, NASA/SP-2007-6105 Rev.1, National Aeronautics and Space Administration, NASA, Headquarters Washington, D.C. 20546, December 2007

which focuses on the technical domain of aerospace and is witness of the relevant contribution given to the SE by this domain, since the edition:

- Robert Shishko—NASA-SP-6105 *The NASA Systems Engineering Handbook*, NASA, June 1995

where the fundamentals of the SE were defined and detailed. Some concepts as the *flow of activities* within the product development, the *elicitation of requirements* based on the system *goals, mission and operation scenarios* are clearly stated. It could be compared to:

- ESA-ESTEC (Requirements and Standards Division), *Space Engineering Technical Requirements Specification*, European Space Agency (ESA) Requirements and Standards Division Technical Report ECSS-E-ST-10-06C, Noordwijk, The Netherlands

A deeper information about the *process* and the *views* of the MBSE could be found in the popular reference:

- *Systems Engineering fundamentals*, the Defense Acquisition University press, Fort Belvoir, VA, USA, 22060-5565, January 2001

being focused on the system acquisition process of the Department of Defense (DoD) of the USA, thus representing the implementation of the SE within the military domain, under the constraints of dedicated technical and military standards. Very often the literature refer to the *DoD approach*, defined by the above cited document, free and available on the web. It is crucial resorting to this kind of support when the system development is strictly subordinated to a precise *commitment document for the acquirement*, as it happens in case of the Army, Navy and Air Force.

Those three references surely give a complete overview about the whole holistic approach, but several authors proposed some deepened descriptions of the related *methods* for an easier decomposition of the system and to face its inherent complexity. Among the others:

- David Oliver, Timothy Kelliher, James Keegan—*Engineering complex systems (with models and objects)*, McGraw Hill, New York, 1997
- Alexander Kossiakoff, William Sweet, Samuel Seymour, Steven Biemer—*Systems engineering: principles and practices*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2003
- Charles Wasson—*System analysis, design and development: concepts, principles and practices*, Wiley, 2nd ed., July 2015

could be appreciated for introducing a clear classification of *goals, missions* and *scenarios* to deal with the *requirements* definition, the *operational and functional and analyses*, then to the *design synthesis*, through a bright path of activities and a driven decomposition of the system elements. This approach applies even to other technical domains like the health care, some industrial and social applications, including automotive, railways systems, smart cities and manufacturing.

It is worthy noticing that implementation in some cases is never so straight because of a lack of experience or an imperfect management of this approach. This difficulty is mainly found in the concept design and in managing the *integration* of system in production, therefore it motivates an extended analysis of the SE *management* as it was performed in:

- Andrew Sage, William Rouse—*Handbook of Systems Engineering and Management*, John Wiley and Sons Inc., 1996 and following editions
- Benjamin Blanchard—*System Engineering management*, Wiley, New York, 2004

A more integrated overview on the SE is provided by some other sources, either by groups deeply involved in the SE education, application or service supplying or even vendors of products for the SE. Among the most known there are:

- Richard Adcock (ed)—*Guide to the Systems Engineering Body of Knowledge* (SEBoK), The Trustees of the Stevens Institute of Technology, Hoboken, NJ, USA, vs.1.3, 2014 (available at www.sebokwiki.org)
- Some references shared for free through the web by the *Vitech Corporation* (www.vitechcorp.com)

A growing interest about the SE was shown by some important Society, like the *American Society of Mechanical Engineering* (the *ASME International*), since the appearance in the catalogue of publications of:

- Thomas Van Hardeveld, David Kiang—*Practical Application of Dependability Engineering: An Effective Approach to Managing Dependability in Technological and Evolving Systems*, The ASME Int., New York, 2015

where a detailed analysis of *dependability* and *traceability* of requirements within the system design is proposed, by introducing several concepts about the product lifecycle development, applied to the mechanical engineering.

Since a longer time the *Institute of Electrical and Electronics Engineers* (*IEEE*) supports the information and the education within the SE through several contributions, mainly based on some journals like:

- the *IEEE Systems Journal* of the IEEE System Council

However a specific focus on the SE is kept by the *International Council on Systems Engineering* (*INCOSE*), through several means, even in connection with other institutions or some firms, basically through the web, several publications and conferences.

This activity already led to the introduction of a newer interpretation of the SE as it was proposed more recently in terms of *Lean Systems Engineering* in:

- Bohdan Oppenheim—*Lean for Systems Engineering with Lean enablers for Systems Engineering*, Wiley, 2011
- Josef Oehmen (Ed.)—*The Guide to Lean Enablers for Managing Engineering Programs*, Joint MIT—PMI—INCOSE Community of Practice on Lean in Program Management, Cambridge, MA, USA, 2012 (http://hdl.handle.net/1721.1/70495)

Those references basically show a link between the strategic approach of the *Lean manufacturing* (King, 2009), enabling a careful reduction of waste and cost in production, and the systematic approach to the product development of the SE, with the aim of combining the effectiveness of the SE approach to the sustainability of the lean process.

All the above cited sources do not complete the wide list of references currently available, but it happens that in daily practice of SE some of those are mentioned for their well known contribution.

A straight implementation cannot be uncoupled from the tools and the test cases. Aside the above literature another set of references specifically deals with the *languages* of the Model Based Systems Engineering.

A unified approach among several technical domains is currently under development by resorting to the *System Modeling Language* (*SysML*), evolution and adaption to the SE needs of the former *Unified Modeling Language* (*UML*). Many authors provide a complete overview about the language and its rationale use within SE, although some are more frequently cited as:

- Sanford Friedenthal, Alan Moore, Rick Steiner—*A practical guide to SysML, the System Modeling Language*, The MK/OMG Press, 1999 (and following)
- The *OMG SysML*, (version 1.1 November 2008 and following)
- Tim Weilkiens—*Systems Engineering with SysML/UML—Modeling, Analysis, Design*, The MK/OMG Press, 2008
- Lenny Delligatti—*SysML Distilled: A Brief Guide to the Systems Modeling Language*, Addison Wesley, 2014
- Pascal Roques—*Modélisation de systèmes complexes avec SysML*, Eyrolles, 2013 (*in French*)

It might be considered that new evolutions are already foreseen, as the Reader could realize in:

- Lifecycle Modeling Language (LML) Specification, (http://www.lifecyclemodeling.org/spec/)

although a first convergence of methods and standard processes is still currently looked for, by a direct implementation of the SysML or at least some enriched version.

More than handbooks about the SysML, it seems crucial remarking the existence of some other sources oriented to its practical use through the methods of the MBSE. Surely very well known is:

- Hans–Peter Hoffmann—*Systems Engineering best practices with the Rational solution for systems and software engineering, Deskbook* (Model Based Systems Engineering with Rational Rhapsody and Rational Harmony for Systems Engineering), the IBM Software Group, © IBM Corporation, 2011

which enunciates the proprietary *Harmony© approach* proposed and implemented within the software tool *IBM Rational Rhapsody*® by IBM. That approach is still matter of testing for several technical domains, where contents are applied to introduce the SE itself. Because of this need, some complete examples of implementation are currently widely appreciated as the:

- Bruce Powel Douglass—*AGILE Systems Engineering*, MK Morgan Kaufmann, Waltham, MA, USA, 2016

where a bright development of a real test case is shown with all the details of modeling activity

Many other sources are already available in this field and listing all precisely should be unpractical, or even impossible. Those were proposed actually to outline some typical contributions, the related subjects covered by the literature and even

because some specific content has been matter of a refinement of the SE methodology, or it helped in assessing some critical issue, as soon as the SE was applied to the series and material products of industry.

Mentioning those references was aimed at showing to the Reader some different topics of education about the SE and even to point out a number of views, which enriched the approach, by making sometimes difficult a fast and complete convergence towards a generalized and comprehensive *standardization*, as is still currently looked for.

To complete this overview it might be mentioned that a specific need of references is still perceived within the definition of *metrics* for the evaluation of artifacts and products during the *verification* and *validation* activities, as it shall be soon explained in this handbook. A preliminary answer to this necessity was given by a dedicated series of textbooks, published by some universities, to support the classes, as, for instance at MIT (Massachusetts Institute of Technology):

- Steven Eppinger, Tyson Browning—*Design structure matrix methods and application*, Engineering System, MIT Press, 2013

which highlights, as a current trend of the literature, to move from some main references dealing with the concept of system and the general approach of the SE towards some detailed topic of its implementation, to assess and refine the *tools* previously proposed.

From the above mentioned citations it looks clear that:

- several technical domains are currently involved within the development of methods and tools related to the SE and express different approaches and needs;
- technicalities are related to some issues like the guidelines for the implementation, the languages and their assessment, the software tools and the creation of suitable data management systems;
- many other activities like the *safety engineering*, the *risk management*, the *maintenance of systems* are connected to the SE and their tools and methods must somehow meet those more typical of the MBSE.

2.4 Technical Standards on the Systems Engineering

More details about the SE processes are today already provided by several technical standards, being aimed at driving the user in implementing the SE and its tools. According to the brief sketch above offered about the history of the SE, it could be considered that a preliminary description of systems was provided by some *military standards* as the *DoD-Mil-Std 499* (1969), the *Mil-Std 499A* (1974) and the *Army Field Manual 770-78* (1979). They led to a preliminary draft of the *Mil-Std 499B* (1994), which actually was never released as itself, but it was enabler of the development of the *ANSI/EIA 632* "*Processes for Engineering a System*", which collects the recommendations of the *American National Standards Institute* (*ANSI*) and the *Electronic Industries Association* (EIA).

The management of the SE was then goal of the *IEEE 1220-1998* "*Standard for Application and Management of the Systems Engineering Process*", which was presented in 1998 and refined in 1999, respectively.

During the early 2000s the real references for the SE were released as the *ISO/IEC 15288:2002* "*Systems Engineering—System Life Cycle Processes*" (2002), as a result of the activity of the *International Electrotechnical Commission* (*IEC*) together with the *International Organization for Standardization* (*ISO*), whilst the guidelines of the INCOSE Handbook (recent version 2015) and of the NASA handbook (recent version 2007) were updated.

Other standards were even published as the *ISO/IEC 19760* "*Guide for ISO/IEC 15288—System Life Cycle Processes*", which drives to a straight application of the ISO/IEC 15288. The *Institute of Electrical and Electronics Engineers* (*IEEE*) basically accepted and applied the relevant contents of the ISO/IEC 15288 in the *IEEE Std 15288 2004*. Moreover, the NASA analyzed several issues of those standards within the *NASA NPR 7123.1A* "*Systems Engineering Processes and Requirements*".

As the Mil-Std 499, 499A and 499B basically describe the life cycle approach, while some details on the process, the goals, the outcomes and the activities were described by the ANSI/EIA 632 as well as by the ISO/IEC 15288 and the IEEE 1220, some differences could be detected among the three main standards. The ANSI/EIA 632 defines the *processes* required to engineering or re-engineering the system, the ISO/ICE/IEE 15288 provides a *framework* to define the system life-cycle, while the IEEE 1220 focuses on the system *management*.

A key contribution to this topic was even provided by the:

- ISO/IEC/IEEE 16085 (2006), "*Systems and Software Engineering Risk Management*"
- ISO/IEC/IEEE 15939 (2007), "*Systems and Software Engineering Measurement Process*"
- ISO/IEC/IEEE 16326 (2009), "*Systems and Software Engineering Project Management*"
- ISO/IEC/IEEE 24765:2009 (2009), "*Systems and Software Engineering Vocabulary*"
- ISO/IEC/IEEE 150261 (2009), "*Systems and Software Engineering System and Software Assurance, Part 1: Concepts and definitions*"
- ISO/IEC/IEEE 150262 (2010), "*Systems and Software Engineering System and Software Assurance, Part 2: Assurance case*"
- ISO/IEC/IEEE 42010 (2011), "*Systems and software engineering Architecture description*"
- ISO/IEC/IEEE 15289 (2011), "*Systems and Software Engineering Content of LifeCycle Information Products (documentation)*"
- ISO/IEC/IEEE 29148 (2011), "*Systems and software engineering Requirements Engineering*"
- ISO/IEC/IEEE 150263 (2011), "*Systems and Software Engineering System and Software Assurance, Part 3: Integrity Levels*"

Those standards clearly demonstrate how much the SE could benefit of the experience within the *software engineering*, although it applies to several other systems, like to aerospace and mechatronics. This link to the software engineering can be better appreciated if some technical standards of that field are analyzed.

In addition to the above mentioned ISO/IEC/15288, recently updated (2015) about the Systems Engineering, the:

- ISO/IEC/12207:2008 (2008) deals with the "*Systems and software engineering—software life cycle processes*", and applies to the software development, seen as a product, and to the processes proposed for a more general implementation to the industrial systems
- ISO/IEC/26262 (2011–2012) defines the *functional safety for automotive equipment throughout the life cycle of automotive electronic and electrical safety related systems* and looks like an adaption of the following one:
- IEC 61508 (2012) even applied to the automotive electric and electronic devices, being entitled "*Functional safety of electrical/electronic and programmable electronic safety-related systems (E/E/PES)*".

Some additional references strictly related to the SE were released directly by the ISO as the:

- ISO 17666:2003 (2003), "*Space Systems Risk Management*"
- ISO 31000:2009 (2009), "*Risk Management Principles and Guidelines*"
- ISO/IEC 31010:2009 (2009), "*Risk Management and Assessment Techniques*"

It is worthy noticing that in all of the above mentioned standards, quite independently on the specific goal of each one, some clear attributes of the SE approach are stated. The approach is *solution-oriented*, operates on the base of well defined *needs* and proceeds through a *holistic view* of the system, including operations, environments, stakeholders, and the whole *life cycle*, as it was brightly confirmed by the *SeBok Handbook* (2014). Among all, the ISO/IEC/IEEE 15288 represents the result of a gradual evolution of those standards. The process of SE there identified follows a rationale which starts from the exploration of needs, investigates functions and behaviors, proposes and analyses some suitable architectures to reach a final synthesis of design.

2.5 Software Tools for the Systems Engineering

The implementation of the SE requires some *software tools* to collect the requirements and to proceed with the modeling activity proposed by this approach. Among the existing software some are quite well known and to make aware the Reader will be herein briefly cited.

The IBM Rational DOORS® is currently widely used for the requirements elicitation. It is usually integrated with the IBM Rational Rhapsody® which allows

implementing the typical tools of the SysML language. However, other vendors already developed similar packages like Agilian® (Visual Paradigm), Artisan Studio® (Atego) nowadays embedded into the PTC Integrity Modeler®, Enterprise Architect® (Sparx Systems), Cameo Systems Modeler® (No Magic) and UModel® (Altova). They are available on market in different versions and usually sold by the vendors above cited within brackets. In addition, are distributed for free the Modelio® (Modeliosoft) and Papyrus® (Atos Origin).

Selection of software tools is quite a critical issue of the SE platform definition. The above mentioned tools offer different contents, in terms of options, modules and features, but a main issue is the format for exchanging the data and their compatibility for a full *interoperability* with several other tools currently used within engineering. This topic will be herein deeply described, since it looks the main bottleneck for a very fast and easy implementation of the SE in all the technical domains and for every application, although the problems occurring in some case might be overcome by resorting to some standard connectors like the *Functional* Mock-up Interface (FMI) or by basing the creation of the *tool chain* on some interoperability standard like the *Open Services for Lifecycle Collaboration* (OSLC).

References

Adcock, R. (2014). *Guide to the Systems Engineering Body of Knowledge (SEBoK)* (Bd. 1.3). Hoboken, NJ, USA: The Trustees of the Stevens Institute of Technology.

ANSI/EIA 632 "*Processes for Engineering a System*".

Army Field Manual 770-78 (1979).

Arrichiello, V. (2014). Introduction to the systems engineering. In *INCOSE Italian Chapter Conference on Systems Engineering (CIISE 2014)*. Rome, Italy.

Bertalanffy, Lv. (1972). General systems theory. *The Academy of Management Journal, 15*(4), 407–426.

Blanchard, B. (2004). *System engineering management*. New York: Wiley.

Defense, A. U. (2001). *systems engineering fundamentals*. Fort Belvoir, VA, USA: Defense Acquisition University.

Delligatti, L. (2014). *SysML distilled: A brief guide to the systems modeling language*. Addison Wesley.

Division, E.-E. R. (2017). *Space engineering technical requirements specification, technical report ECSS-E-ST-10-06C*. Noordwijk, The Netherlands: European Space Agency (ESA) Requirements & Standards Division.

DoD-Mil-Std 499 (1969).

Douglass, B. P. (2016). *AGILE systems engineering*. Waltham, MA, USA: MK Morgan Kauf-mann.

Eppinger, S., & Browning, T. (2013). *Design structure matrix methods and application, engineering system*. MIT Press.

Estefan, F. A. (2008). *Survey of the model-based systems engineering (MBSE) candidate methodologies*. The INCOSE MBSE Initiative.

Friedenthal, S., Moore, A., & Steiner, R. (1999). *A practical guide to SysML, the system modeling language*. The MK/OMG Press.

Hoffmann, H.-P. (2011). *Systems engineering best practices with the rational solution for systems and software engineering, deskbook (model based systems engineering with rational rhapsody and rational harmony for systems engineering).* The IBM Software Group.

IEEE 1220-1998 "*Standard for Application and Management of the Systems Engineering Process*".

ISO/IEC 15288:2002 "*Systems engineering—System life cycle processes*".

ISO/IEC 19760. (2002). "*Guide for ISO/IEC 15288—System life cycle processes*".

King, P. L. (2009). *Lean for the process industries.* Taylor and Francis.

Kossiakoff, A., Sweet, W., Seymour, S., & Biemer, S. (2003). *Systems engineering: Principles and practices*. Hoboken, NJ: Wiley.

Lifecycle Modeling Language (LML) Specification. (2017). http://www.lifecyclemodeling.org/spec/.

Mil-Std 499A (1974).

Mil-Std 499B (1994).

NASA. (2007). *Systems engineering handbook* (Bd. rev.1 and following versions). NASA/SP-2007-6105.

Oehmen, J. (2012). *The guide to lean enablers for managing engineering practice on lean in program management.* Cambridge, MA, USA: MIT—PMI—INCOSE Community of Programs.

Oliver, D., Kelliher, T., & Keegan, J. (1997). *Engineering complex systems (with models and objects)*. New York: McGraw Hill.

Oppenheim, B. (2011). *Lean for systems engineering with lean enablers for systems engineering.* Wiley.

Roques, P. (2013). *Modélisation de systèmes complexes avec SysML.* Ey-rolles.

Sage, A., & Rouse, W. (1996). *Handbook of systems engineering and management.* Wiley.

SeBok Handbook. (2014). www.sebokwiki.org.

Shishko, R. (1995). *The NASA systems engineering hand-book.* NASA-SP-6105.

Van Hardeveld, T., & Kiang, D. (2015). *Practical application of depend-ability engineering: An effective approach to managing dependability in technological and evolving systems*. New York: The ASME Int.

Vitech Corporation. (2017). von www.vitechcorp.com.

Walden, G. K. (2015). *Systems engineering handbook of INCOSE.* Wiley.

Wasson, C. (2015). *System analysis, design and development: Concepts, principles and practices* (2nd ed.). Wiley.

Weaver, W. (1948). *Science and complexity, American scientist, 36*, 536. New York City: Rockefeller Foundation.

Weilkiens, T. (2008). *Systems engineering with SysML/UML—Modeling, analysis, design.* The MK/OMG Press.

Wiener, N. (1948). *Cybernetics: Or control and communication in the animal and the machine*. Boston, MA: Technology Press.

Chapter 3
The Methodology of Systems Engineering

Abstract It is well known that the amount of contents related to the Systems Engineering often makes confused the users about their suitable exploitation in a linear product development process. Therefore, in this first technical chapter, some basic concepts are defined and some typical models of product development are shown. They are compared and distinguished from some architecture frameworks currently applied to define the system capabilities and some crucial views. A sketch of the main steps of the Systems Engineering implementation is finally drawn, together with a list of some proprietary approaches proposed by societies, companies and software vendors for its straight deployment. Tools and languages are even described, together with some engineering methods and the SysML language is used as a relevant example to describe the main diagrams applied in functional modeling.

3.1 Introduction

As it was already stated in previous chapters, the Reader should master the *methodology* of the SE which includes *processes*, *methods* and *tools*, both theoretical and software. Those allow implementing a *model* of product development, defining some typical *views* of the system and some related *functional* and *physical* models to accomplish the whole design activity. To catch completely the core of the SE understanding the definition of *system* is essential. Main differences between the so-called *architecture framework* and the SE *process* will be analyzed as well as some *methods* and *tools* related to the use of standard *languages*.

3.1.1 Definitions of System

According to a popular description, the *system* is an "*assembly of elements linked each other and strongly interacting and interdependent*". This definition clearly

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_3

points out the intrinsic nature of *assembly* and of the *interaction among components*. For a full understanding of the proposed process applied to system development, the NASA handbook on SE explicitly goes deeper in details, by stating that:

> A system is a *set of interrelated components* which *interact* with one another in an *organized fashion* toward a *common purpose*. The components of a system may be quite diverse, consisting of *persons*, *organizations*, *procedures*, *software*, *equipment* and/or *facilities* […]

thus stressing the *common goal* of all parts composing the system and the *different nature* of those components (even human). This interpretation is somehow explained by the definition of system found in the Standard Mil-Std 499B, being seen as:

> An integrated composite of people, products and processes that provides a *capability to satisfy a stated need* or objective

where the connection between *system capability* and *need* is clearly stated.

To complete that scenario, the INCOSE handbook brightly summarizes those definitions by calling *system*:

> An *interacting combination* of elements viewed in relation to *function*.

Those definitions are sufficient to collect the main keywords related to the system to be developed through the SE. Moreover, *engineering the system* finally means that a product is developed:

- To satisfy some *needs*, as a solution-oriented *assembly of components* which might be even *heterogeneous* (people, products, processes).
- Through a *harmonized integration* able to express some specific *capabilities* through dedicated *functions*.
- In such a way that each function is *expression on the interrelated activity* of the system components, organized in *a defined architecture*, to be assessed through a selection (*trade-off*) among some proposed and compatible *available technologies*.

3.1.2 The System Development as an Industrial Product

Practically speaking, the real contents of the system design and production include three main activities, according to definitions above mentioned.

Goals, mission and requirements identification. To satisfy some defined *needs*, the system has to provide some suitable functions, which are associated to a preliminary identification of *goals*, *missions* and *scenarios* which better describe its lifecycle, together with the *actors* involved in such operation and the *constraints* limiting its capabilities. That knowledge allows defining the system *requirements*, through a careful elicitation, which should be complete and exhaustive, as much as possible, to reach the benefits of the "*left-shift*" previously described.

System behavior. To suitably shape the system to fit requirements, its behavior has to be predicted. In a first step of investigation the *neighborhoods of system* should be clearly identified as well as the different *usages (or use cases)*, to define the *actors* who interact with the system itself. System behavior can be then described by understanding the context of its operation, in terms of *interactions* between system and external actors, of *sequences* of *activities* performed, through different *states* temporarily reached and held by the system.

System architecture. After a preliminary exploration of functions required to satisfy the needs, each function should be associated to a sub-system, component and part to define a *configuration* which exploits a certain *technology*. A quantitative analysis of the system behavior could refine the requirements and assess the system *layout* to select and eventually size the components to be included in the system *integration*.

Those activities are essentially based on some concepts, which will be herein defined and deployed.

The concept of traceability. The main idea promoted by the SE is to assure a complete *traceability* from the requirement to the manufactured part as it is sketched in Fig. 3.1. This approach allows relating any failure occurring to a component or part to the corresponding requirement and identifying any potential critical issue associated to the system safety. Moreover, if each requirement is properly *allocated* to some function, then to subsystems, components and numbered parts, the system complexity is decomposed and a clear connection between *need* and *solution* is established.

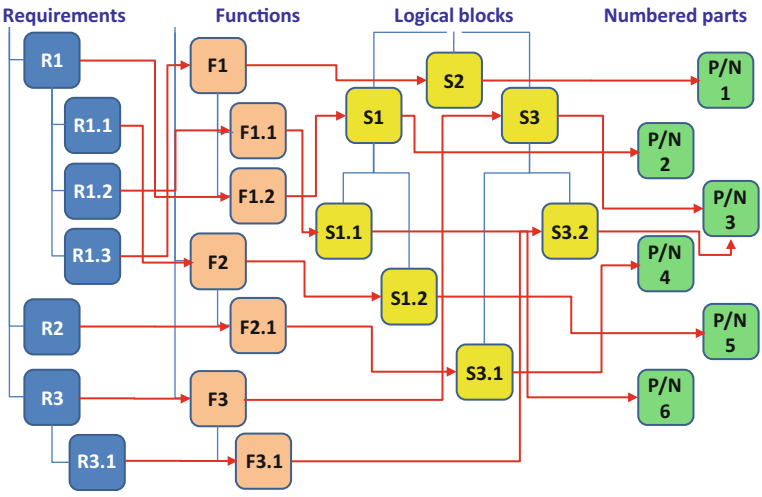


Fig. 3.1 Traceability and allocation of requirements through the product development in systems engineering

This approach fits the needs of designer to have a linear correspondence between each requirement and a material component of the system, assuring the *traceability*. Nevertheless, the rationale, bringing the project from an identified need through the requirement, till the realization within a system component, is based on two intermediate steps. First at all, each requirement should be corresponding to a *function* or a set of functions, which the system has to provide. Every function is seen as an action, being performed by a *logical block* first, then by a *material component*, usually available on market or directly manufactured and identified by its code number. Obviously this interpretation mainly describes a *material product*, made through a technological process, although in some issues even the software could be developed following the same approach.

It might be noticed that the above interpretation already defines a specific *process* to be implemented, but not all of models of SE-compatible process proposed in the literature perfectly correspond to those items. Particularly, the distinction between a *logical*, or even in some reference, *functional block* and a *numbered part* in some case is no longer used. In principle, it might be helpful if one considers that difference as substantial, since the block is never yet associated to a specific commercial or manufactured device like the numbered part is, but simply to a specific component. Function "rotating", for instance, might be associated to a generic "motor" as a logical block, then allocated to a "brushless motor with given set of properties", when the real component is defined.

As Fig. 3.1 describes, among the requirements "R" collected through a preliminary elicitation, father ("R1") and son ("R1.1") requirements are each by each related to a function ("F1") or subfunction ("F1.2"), being realized by the subsystem ("S1") or component ("S1.2"), which represent a simple actor, without a precise correspondence to a pre-defined commercial device, which is only associated later, when a dedicated architecture is defined, as a numbered part ("P/N 2").

If the system is an aircraft and the requirement is "being capable of propelling itself during the flight", the function is the propulsion, while the logical block is a generic engine, which might be built by resorting to different technologies (turbine, turbofan, …), whilst after a preliminary *trade-off* the designer identifies the real nature of the turbo machinery, for instance a turbofan, with some power and some defined dimensions.

This *process* seems extremely simple and linear, but some are the difficulties which affect its fast and easy implementation. The *elicitation of requirements* is never so fast and simple, but it has to be refined through several iterations. Associating the *logical blocks to real components* needs a detailed analysis of available technologies and a bright *trade-off* among some candidate solutions. *Modeling* those entities actually requires different tools, since requirements are set through verbal information, numbered parts are objects, in the material product development. Functions and logical blocks need to deploy a smart *view* on the system behavior and layout.

Those exigencies motivate the four pillars described in previous chapter. They meet in two specific items:

- The need of a systematic approach, described by a *process*, aimed at introducing a *method*, which exploits a certain *language* and some *tools*.
- The deployment of that approach through several *views*, which focus on different attributes of the whole system. This is extremely important if the system is seen as a good to be acquired.

The literature deeply investigated the role of processes and views in the SE methodology, somehow making the Reader confused about their application. To clarify this issue, it is worth noticing that different technical domains are prone to focus either on:

- The product development from the customer need to the constructed system, being seen as an *object of the whole manufacturing process*, thus identifying the *steps of design and manufacturing* as the core of the development;
- The commitment of producing the system, being seen as *a good to be acquired to be integrated into a larger system* (a fleet, an army, …), thus requiring that a set of skills (*capabilities*) could be exploited in operation and they could be simultaneously used as main targets to define the system design through the whole manufacturing process, from design to disposal.

The first approach assumes that the *product life cycle development* is the first task to be accomplished, by selecting and implementing a suitable model. The second one adopts a set of *standard views* as a suitable reference to proceed, focusing on the so-called architecture framework.

Despite a common feeling of the SE operators, views and models of product life cycle are not in contrast, perhaps they could be seen as two points of view of the same ideal matrix of the system design and production.

The *process* defined by the SE *methodology* meets the collection of *views* which can be realized to describe and instantiate the system properties and attributes, by following an *architecture framework* used by the system engineer to develop the product. Those concepts are somehow confused in the literature, or at least, unsuitably superposed, thus leading to some understanding. It is worth noticing that *process* and *views* are somehow interdependent. Tools used to perform the process allow obtaining some outputs, which can be used to depict the required views, as well as views are essential in the elicitation of system requirements, defining the architecture candidate to be applied to build up the system, or even in identifying actors, interfaces and connections.

3.2 The Models of the Product Life Cycle

As it was just above stated, to describe the approach proposed for developing the system, the specialized literature usually introduce a model of the product life cycle, corresponding to some diagrams, which defines the sequence of actions to be performed to reach a design synthesis first and the product disposal after. It is worth

noticing that some current limitations affect the application of these models of product lifecycle to all the technical domains, nevertheless their capability to communicate an intuitive impression of the process suggested by the SE is considerably relevant.

3.2.1 The Waterfall Diagram

The most intuitive model of life cycle consists in the so-called *waterfall diagram*, being simply a sequence of blocks or items, listed by reproducing a water drop from the top to the bottom, to represent the time at which actions are performed. Figure 3.2, for instance, shows an example of waterfall, whose main limitation is that very often it is incomplete and actions related to the design and manufacturing or testing activities, respectively, are poorly associated. This approach turns out into a misunderstood interpretation of all the relevant implications and links existing between the ALM and PLM stages.

3.2.2 The V-Diagram

To overcome the limitation of previous model, the so-called *V-diagram* is often proposed to show some typical key steps of the SE process, as it looks in Fig. 3.3.

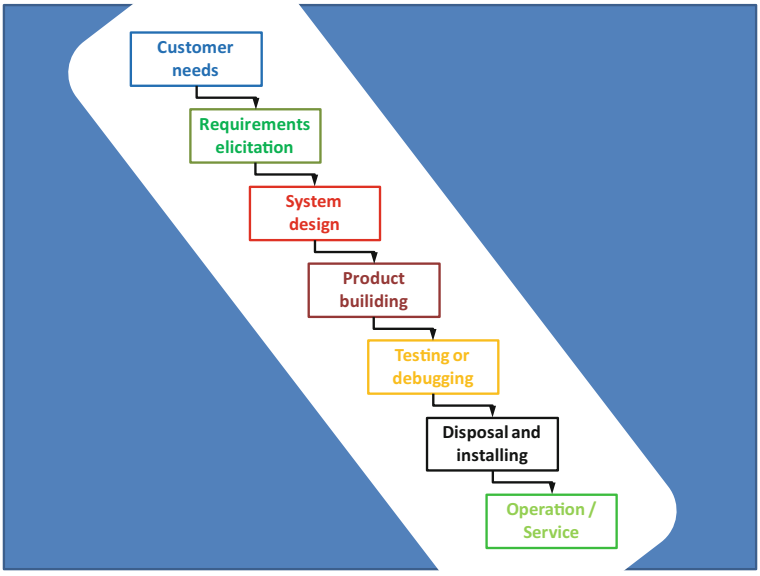


Fig. 3.2 Model of the product lifecycle described by a waterfall diagram

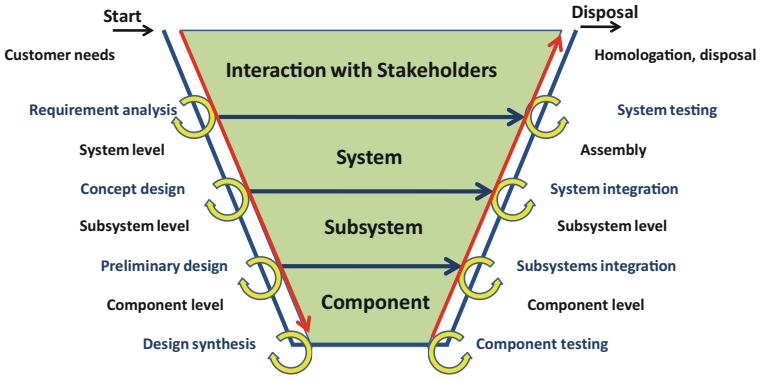


Fig. 3.3 Description of the "V diagram" applied to the product development in systems engineering

This diagram is quite popular and provides a number of useful outlines about the process to be applied, although some elements are often interpreted as poorly precise to allow an immediate perception of contents, as it will be herein pointed out. The path to be followed starts from the upper left corner, goes down to the bottom through the left side, and then continues by going up through the right upper corner, across the actions described on the right side.

It is clear that a *holistic approach* is there evidenced, since the Reader can easily find all the steps of the system development from the very early concept (start) to the service and after (disposal). The hierarchy of levels suggests to start from the whole system to design the details of components and parts, thus preferring a *top-down approach*. It might be immediately remarked that is quite unusual for a material product, since it is often applied a *bottom up approach*. The "V" structure of this diagram emphasizes the need of considering each action of the design activity, on the left side, strictly related to a corresponding issue of production, depicted on the right side. The *Application Lifecycle Management*—ALM, corresponding to the left wing and the *Product Data Management*—PDM, being the right wing, should be never considered as two separated activities, and almost uncoupled. By converse each design issue needs to be *verified* and *validated*. This is extremely important because, for each function, a *test* must be foreseen to verify whether the requirement is fulfilled and to validate whether the corresponding need is satisfied. In this activity, there are some key operations like *testing*, *integration* of elements (components, subsystems, and the whole system) and *assembly*, which need a preliminary check about the feasibility of product.

The chronology of steps somehow provides some guidelines to be followed. Within the design activity, on the left side, the Application Lifecycle Management is basically considered. It starts with a key item like the detection of the *customer needs*, thus stating that customer plays a fundamental role in the product development and the related *needs* are the source of requirements and specifications to

design a *solution*, being the product itself. The requirements should carefully consider the *stakeholders*, e.g. all of actors (persons, processes, systems and devices) exhibiting some kind of interest for the system, thus motivating a direct interaction, being different use case by use case.

The right side of the *V-diagram* helps in perceiving the need of relating each step of the ALM to some key activities of *production*, as *testing*, applied to parts, components, subsystems and to the whole system, together with the *integration*, the final *assembly* and even the actions associated to, *transportation*, *installation*, *service* and *disposal*. An evident need depicted by the diagram consists in assessing some *standard metrics* to be applied for an evaluation of *performance* and suitable tools to perform both the *verification* and the *validation*.

The above described model is very well known in the literature, although nowadays many Systems Engineers are prone to find a couple of limitations in that representation of the product lifecycle development. A first rough *limitation* is that requirements elicitation looks like complete and finished after the first steps of the process. Actually it definitely doesn't. In many cases, needs, standards and practice of technical domains suggest a number of requirements, but only a deeper analysis of system functions, operations, architecture and of some measurable performances allows assessing that list. Therefore, a lack of clarity of that diagram concerns the *recursive process implemented* to define suitable requirements. That's the meaning of circular arrows added in Fig. 3.3. In practice, requirements are defined for the whole system, then applied to each subsystem, component and part, whilst a design process is started. At the beginning, only a conceptual design is performed, by resorting to a *functional modeling*, then functions are associated to some logical devices, providing those functions, but not yet associated to a specific technology, and a preliminary design could be performed. Nevertheless, this preliminary architecture needs to be verified and somehow tested, to be then specified in a synthesis of design, which includes more details and associates all those logical blocks to real parts, selected on market or produced by design. In principle, as the design goes deeper, the system layout could be better defined even in its smallest components. In this step it is crucial resorting to a *physical modeling*, based on mathematics, i.e. equations which allow predicting the main figures of performance of the system behavior.

This diagram often *doesn't cite the aftermarket operation*, including maintenance and service. It could be foreseen another cycle of operations which includes a first *training of operators*, the *system operation*, then a *monitoring activity* which leads to a continuous maintenance, an evaluation of performance and even an *optimization of the system configuration*. Those tasks basically motivate two current developments of the roadmap previously described for a straight product lifecycle management. *Connectivity* could help in monitoring the service of systems until their decommissioning, for instance like the growing up strategy of "Industry 4.0" describes and deploys. Moreover, instead of applying the Systems Engineering to single products, a newer implementation considers a *Line of Products* and focuses on how systems are changed, version by version of the same product, through a progressive configuration change, not limited to the single product development. It

might be associated no longer a pure Model Based Systems Engineering but even more a Model Based Product Line Systems Engineering.

3.2.3 The Spiral Diagram

Those limitations motivated the proposal of the new model of product lifecycle based on a spiral frame, to point out that the SE process is a recursive approach and each step needs an assessment, being performed by refining the previous one, step by step, circle by circle, particularly when requirements are concerned. This representation suggests that each step might take a look on the nearest part of the diagram previously covered, being deployed in the inner region, as in Fig. 3.4.

As the deployment goes towards the center of the spiral, the product reaches its complete development and activities are gradually performed and completed. As in previous case, main steps are identified, consisting of requirement analysis, functional analysis and modeling, physical modeling and final validation. As it looks

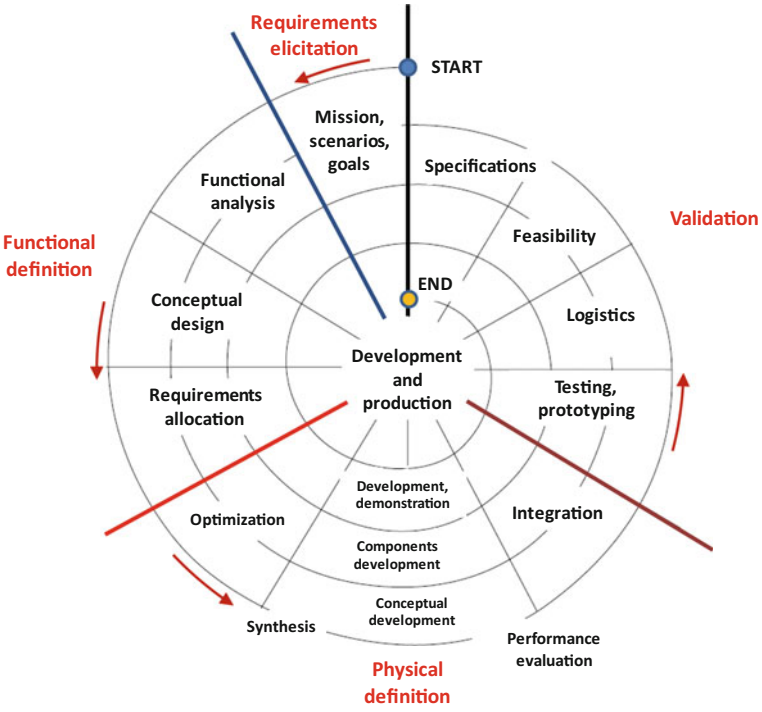


Fig. 3.4 Sketch of the "spiral diagram" applied to the product development in systems engineering

evident after a first turn, specifications are assessed and a better design of details could be performed.

Several operators in the field of Systems Engineering find still some lacks of information in this representation, which is surely fairly well detailed. In the technical domains related to the manufacture of material products, those people complain the two main issues should be integrated, since the early steps of the proposed development. In case of safety critical systems, i.e. of products associated to potentially severe failures and high risk, those activities should include not only a functional analysis, and related modeling, but even a parallel deployment of a *non functional analysis*, or even *dysfunctional*, being tailored to detect all the relevant requirements connected to the safety and reliability of the system. It is nowadays widely assumed that a late safety and reliability analysis performed on a definitive configuration of systems is ineffective to prevent severe failures and their catastrophic consequences. In addition, a complete design should evaluate, since its beginning, the *business model* to be applied for that product, to prevent high costs, infective marketing and unsuccessful disposal. Those reasons motivate to associate to the model of product lifecycle a defined process, which is currently object of assessment and refinement by several technical standards, some vendors of dedicated software and even academy. Before proposing a deeper investigation about process, the alternate approach of the *architecture framework* will be described.

3.3 The Architecture Frameworks

As the Reader could realize, the model of the product life cycle expressively looks at the system as a *product* to be designed and manufactured, and actions are performed with the aim of providing the complete system at the end of that *process*. By converse, if the point of view of the *customer* is mainly privileged, the focus might move upon the *validation*, i.e. on the correspondence between the product manufactured and acquired and the needs, or even the original idea of the commitment. This is crucial in a procurement activity of systems, based on some preliminary agreement. In this case, the same product could be seen as an *object* exhibiting some *capabilities* to be exploited in operation. Therefore, some *views* of the system might help in defining its characteristics and, consequently, in identifying some suitable process to be applied in design, manufacturing and testing. This interpretation leads to define a *framework* within which the designer should find the solution to be proposed to the customer and the related *architecture* of the analyzed system.

Many are the *architecture frameworks* proposed to define the main views of a system. Some of those are fairly popular in the literature. They were proposed by several actors, including public institutions, technical standards, communities working within the same technical domain or even some software vendor. Some examples are the:

- AF-EAF: Air Force Enterprise Architecture Framework.
- AFIoT: IEEE P2413 Architecture Framework for the Internet of Things.
- AAF: Automotive Architecture Framework.
- DoDAF: US Department of Defense Architecture Framework.
- ESAAF: European Space Agency Architecture Framework.
- MODAF: UK Ministry of Defence Architecture Framework.
- NAF: NATO Systems Architecture Framework.
- TOGAF: The Open Group Architecture Framework.
- UAF: the Unified Architecture Framework proposed by NO MAGIC.

The exigency of developing an architecture framework basically was found in the defense domain, to have a comprehensive guidance to enable an effective commitment and a consequent successful development of tailored products. Moreover, it provides a common information and communication infrastructure to connect customer and manufacturer. Somehow, as is it brightly claimed by Nordqvist, Edberg, Byström, and Gustfsson (2006) as far as connectors in computer science enable the interoperability among tools to create the right environment for the heterogeneous simulation, the architecture frameworks allow to create a sort of "intellectual interoperability" among people, especially when customer and system engineer meet.

The architecture frameworks, easily speaking, provide some guidelines about how describing the system architectures, while they do not drive the design activity itself. Daily practice suggests that the architecture frameworks support the decision-making process, allow defining some capability requirements, and enable the communication among the system suppliers, developers, producers and customers. As it was previously stated, the architecture frameworks are used to identify the needs and allocating them to the system.

Those goals were clearly detected by the ISO who proposed some standards about the architecture frameworks, like the ISO/IEC/IEE 42010 (2011), which practically harmonized and completed some contents of the Architecture Frameworks above listed. A brief description will be herein proposed just to understand what are the views proposed and how they could be applied to the system development, while a more detailed analysis of this subject could be found on the references related to the Architecture Frameworks above cited.

3.3.1 MODAF

In case of the MODAF, for instance, the architecture framework is based on six viewpoints, shown in Fig. 3.5. Within the viewpoints, up to 38 views are foreseen and enable each one to depict a particular aspect of the system.

The views describe some capabilities required to the system, motivating a specific selection of a suitable architecture aimed at providing some function. Just to make aware the Reader about the contents, viewpoints are briefly herein described.

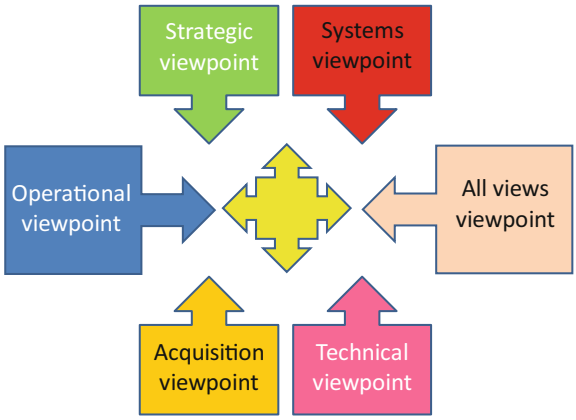


Fig. 3.5 Viewpoints of the architecture framework MODAF

The *Strategic Viewpoint*, for instance includes the capabilities:

1. *Vision*: how the goals shall be reached over time
2. *Taxonomy*: how to structure and decompose capabilities
3. *Phasing*: available capabilities for different time periods
4. *Clusters*: relationships among capabilities
5. *Systems Deployment Mapping*: deployment of systems and relations
6. *Function to Operational Activity Mapping*: mapping between capabilities and operations.

The *Operational Viewpoint* consists of seven views:

1. *High level operating concept*: graphical and textual descriptions of system context and operational performance associated to the scenarios.
2. *Operational node connectivity*: it describes some operational nodes and data exchanged between nodes.
3. *Operational information exchange matrix*: it is used to describe the information exchanged.
4. *Organization relationships chart*: it depicts the command structure.
5. *Operational activity model*: operational activities.
6. *Operational activity sequence and timing*: rules and constraints for the enterprise or business; operational states and event traces.
7. *Logical data model*: relationships among operational data elements.

The *Systems Viewpoint* has eleven views:

1. *System Interface Description*: systems and related interfaces and nodes.
2. *Systems Communications Description*: ports specification and protocols, port connectivity, connectivity clusters, which describe how individual connections are grouped into logical connections between nodes.
3. *Systems-Systems Matrix*: interface characteristics.

4. *Systems Functionality Description*: hierarchies of system functions and data flows.
5. *Operational activity to systems functionality traceability matrix*: relations between operational activities and system functions correlated.
6. *Systems data exchange matrix*: characteristics of data exchanged.
7. *Systems performance parameters matrix*: quantitative metrics of systems and hardware/software elements, interfaces and functions.
8. *Systems evolution description*: systems and architectures evolution over time.
9. *Systems technology forecast*: development of the supporting technologies.
10. *Systems functionality sequence and timing*: rules for the selection of the system architecture; state transition between two events; event traces, recorded and exchanged between system elements.
11. *Physical schema*: structure of data and properties of the system.

The *Technical Viewpoint* includes only two views:

1. *The technical standard profile*: it shows the standards and the constraints applied to the architecture.
2. *The technical standards forecast*: prediction of changes in standards and conventions defined into the technical standard profile.

The *Acquisition Viewpoint* just is based on a:

1. *Systems of systems acquisition clusters*: where the acquisition plans are described.
2. *Systems of systems acquisition programs*: which are focused on the dependencies between a main program and the projects of each system.

The *All (other or inclusive) Views* finally summarizes the information through two views:

1. *The overview and summary information*: they describe the whole architecture.
2. *The integrated dictionary*: it defines the terminology used within the architecture definition.

It is remarkable that many items pointed out by the views are associated and connected to the steps of product development previously described by the models of product lifecycle. Moreover, in many views several keywords linked to the definition of customer needs and to the elicitation of requirements appear. This demonstrates the intimate link between views and process. They look like rows and column of a same ideal matrix to be used to describe the contents of the system design and production.

3.3.2 UAF

As it can be immediately appreciated, the list of views and viewpoints might change case by case, depending on the technical domain, but it provides a real frame to

describe the overall properties of the system to be designed and validated after design. The architecture frameworks above mentioned basically adopt some different viewpoints and views inside those. The *Unified Architecture Framework (UAF)* recently introduced proposes a reference matrix which summarizes the viewpoints in rows, while columns refer to the main information to be shared and defined (Fig. 3.6). It clearly states the relationships between viewpoints and system properties or attributes, while it gives to the user a quite manageable tool for its application. Moreover, a certain merge of the most interesting and qualified contents of many other architecture frameworks was performed, thus leading to converge towards a unified approach, applicable in several technical domains.

The main milestones useful for the industrial product were all enclosed in the framework, particularly where the information were set up. Moreover, the key activity of simulation is used to connect processes, states and interactions. The requirements are clearly evidenced as a main item, as well as the glossary and a summary.

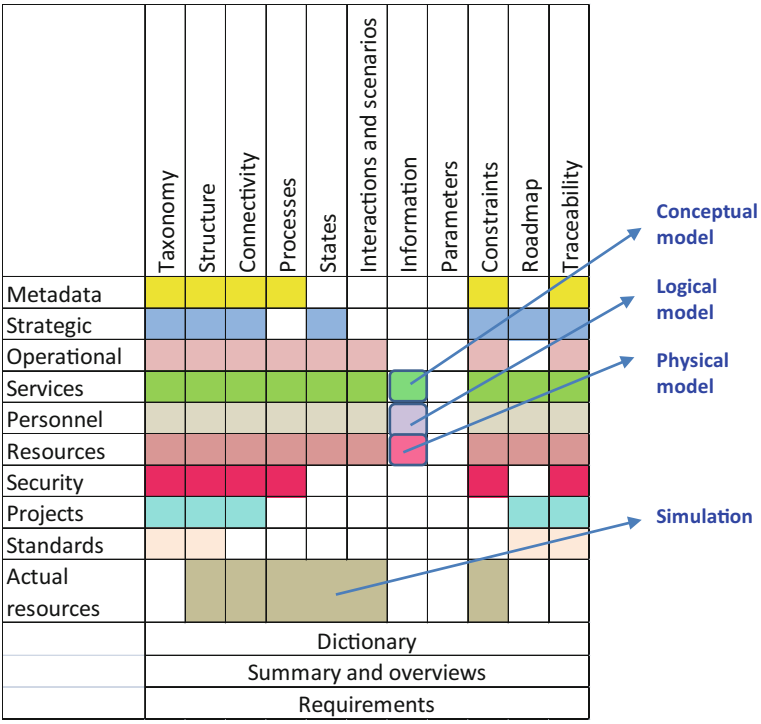


Fig. 3.6 Viewpoints of the architecture framework UAF

3.3.3 Framework and Process

It is worth noticing that a *frame* is just a common language, but contents have to be filled by analysis. This task is performed by following a *process* and resorting to a standard way to represent things, like only a *language* can support. After the elicitation of requirements, based on the needs identified, even through a preliminary selection of capabilities, a modeling activity is performed and some simulations are completed.

The model is a physical, geometrical, mathematical, procedural representation of a real, ideal or even virtual system. The simulation consists into the implementation of the models in some executable forms, which allows investigating the reactions of the system modeled to different events or simply over time.

According to the MBSE, the process is aimed at defining a model of the system which can be used to describe its behavior and architecture to allocate the requirements. The architecture framework could help in defining the details of the system, to establish the needs and all the relevant aspects to be considered in its development, to drive the requirements elicitation, eventually providing a suitable reference to validate the system. Functional and physical modeling activities should provide some suitable representations to deploy the *views* which are foreseen by the architecture framework, being in some domain the right and standard way to express and represent the result of the whole product development, or to check the compliance between the contents of the commitment as well as the system performance and properties.

3.4 The Industrial Implementation of the Methodology

In many technical domains involving industrial engineering, i.e. those aimed at finalizing the activity of product development to manufacturing, a simplified sketch of the above mentioned representations of the SE process is close to that depicted in Fig. 3.7.

According to the representation given in Fig. 3.7, industrial systems engineers are often prone to follow a main flow of activities, which is described in the middle of sketch and includes:

- A preliminary identification of *customers* and their *needs* together with an enterprise model.
- A preliminary elicitation of *requirements*, oriented to the system *mission*, *goals* and *use cases*, for given *scenarios*.
- A definition of system *functions*, based on requirements, use cases and *stakeholders* behavior, aimed at describing a preliminary *architecture* where *interfaces* are defined. In this case blocks are related only to functions and capabilities.

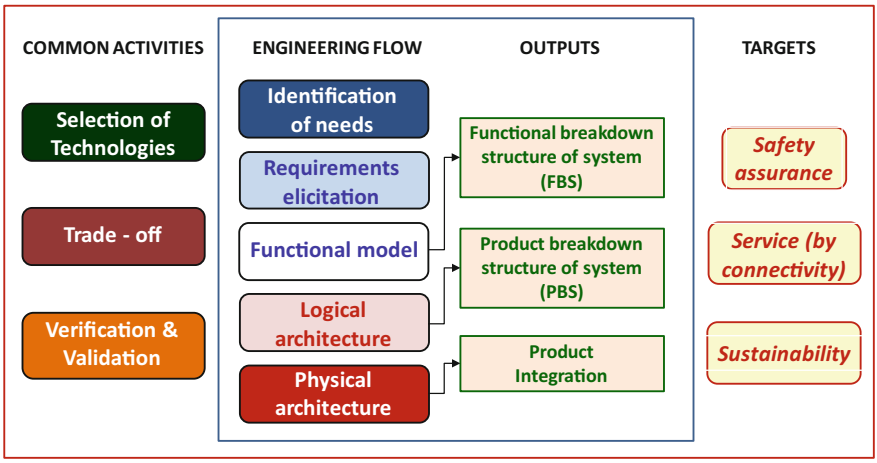


Fig. 3.7 Simplified flow of engineering used in some technical domains with aside descriptions of main activities and targets

- A definition of a *logical architecture*, which transforms the functional architecture into a *product breakdown structure* composed by logical blocks, nominal actors neither yet associated to a specific technology or configuration, nor to a commercial product.
- A *physical architecture*, where functions and logical blocks are finally related to a detailed selection of devices and subsystems.

This interpretation is based on the consideration that an intermediated layer like the logical architecture allows the designer uncoupling capabilities and functions foreseen in previous steps from a specific set of devices and components, exploiting a given technology. This strategy allows proposing different solutions, when available, and comparing their performances. The same result cannot be achieved only resorting to the functional analysis, because it is not assured that each function is performed only by a single system component, therefore the step between functional architecture and logical one usually allows decomposing the function allocation into a distribution of blocks, being closer to the final physical configuration of system, where each generic logical block is then substituted by a real device. It might be pointed out that not all the processes proposed in the literature accept and adopt that interpretation, but in some cases, from a preliminary description of system functions, the real components and parts are then associated to those functions, whilst a candidate architecture is drawn. So far, the intermediate step of a logical description of the function to identify a sort of "function maker", not yet corresponding to a real device, is an option, although one of the most effective.

Actually, when that interpretation is implemented, three main outputs are provided as described in Fig. 3.7, namely the *Functional Breakdown of System (FBS)*, the *Product Breakdown of System (PBS)* and the *Product Integration*, which define well the real steps of the information assessed in the product lifecycle development.

Moreover, some common activities are performed to proceed with the flow of activities above depicted. For each level of development, concerning system, subsystems, components and parts, designer investigates the available *technologies*, defines some alternate *solutions*, based on some *architectures*, then performs a *trade-off* analysis to select the best one. Once that system is defined through a design synthesis, a complete *verification* and *validation* activity is performed.

Several are the criteria used in those actions, but they could be summarized basically by the *safety* and *security* assurance, by the evaluation of *performance* in terms of capabilities, smartness, connectivity, reliability, availability and maintenance in service and by an overall *sustainability*, which includes cost, feasibility, environmental compatibility and, somehow, the quality of product.

Those items are compatible with the concept of *dependability*, being precisely explained for instance by Douglass (2016) and sketched in Fig. 3.8. Main steps of the SE process are there associated to some key activities of the product development, thus pointing out the need of defining several details like *layout*, *technology*, *integration* and final *architecture* before reaching the verification and validation, and through a *closed loop* of feedbacks always active until the disposal of system, even during its operation.

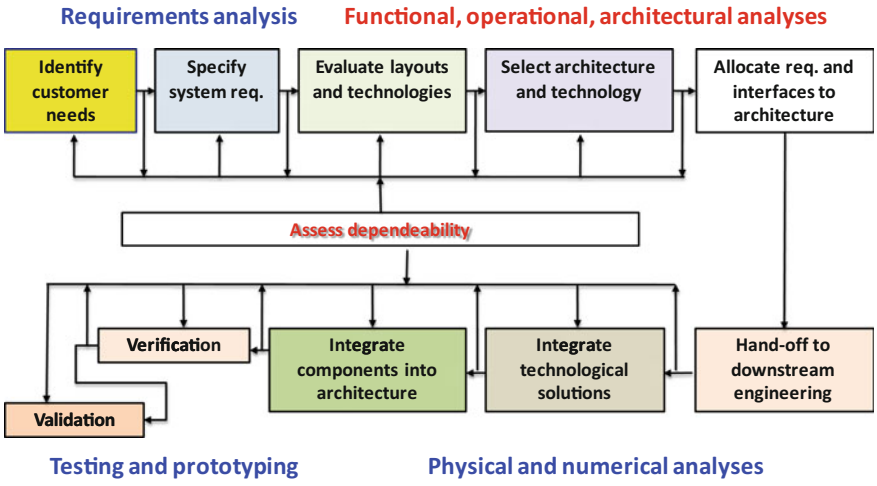


Fig. 3.8 A straight interpretation of dependability within the product lifecycle development as recently proposed by B. Powel Douglass

3.4.1 Key Issues of the SE Process

Those representations of the SE approach were proposed to make the Reader appreciating that in the literature and in some standards the interpretations given to this subject are slightly or substantially different, although some key concepts are kept by all the contributions. Nevertheless, once that main contents of process are caught, their implementation is never easy, if it cannot be driven by some guidelines and performed through some suitable tools. A *method* to deploy the process is strictly required, to face the problem of *how* implementing *what* the process describes. Before that a preliminary overview of *methodologies* currently proposed could be provided, which include also the methods to implement the process, it looks worthy explicating some crucial concepts and issues common to all of those.

A relevant result of the MBSE implementation concerns the time at which a certain subject in the system development is addressed and the tool used to perform the analysis. Those details are essential for the assessment of the SE approach and, somehow, for the existing competition among vendors of the dedicated software or even among companies offering the MBSE services.

Main targets of this discussion are:

- Definition of the system architecture and technology.
- Differences and integration of functional and physical modeling activities.
- Safety and security assurance and differences.
- The real contents of the verification and validation (V&V) process.
- The industrial baseline proposed for the system engineering.

Architecture and technology. Definition of some candidate *layouts* for the system development might suffer the problem of either:

- Resorting to some well-known architecture, being often legacy of the technical domain where designer works, without exploring any other new idea, thus reducing the effort towards innovation.
- Feeling a lack of ideas, similar to the fear of writer for the blank page, being associated to a real need of following some drivelines to proceed and excite new concepts and solutions.

This need is covered by the definition of the so-called *architecture frameworks*, which are used first to define the commitment of a new product, then to validate the proposed solutions.

Similarly, the selection of the most suitable *technology* to be applied in the product development might be affected by a certain trend of designer to resort to some and typical solutions, by avoiding to investigate the possibility of applying a completely different technology, perhaps typical for another domain. In this case, the tools of the MBSE exhibit a lack of a specific representation to import and digitalize a screening of available technologies, which could be reusable and complete. Nevertheless, in this handbook, some additional tools will be described,

although they are not yet included in some standard language of the SE, as the *technology charts*.

Functional and physical modeling. As previous sections described, there is a need for a preliminary *qualitative analysis* of the system development to define clearly goals, actors, functions and interactions, being poorly associable to some mathematical model. This first task is the goal of the so-called *functional modeling*. However, a complete refinement of requirements can be performed only by resorting to a *quantitative prediction of performance*, which can be made through a mathematical modeling of system, including a geometrical description of its layout, a dynamic analysis of its behavior and, eventually, of energy conversion and dissipation. This is the activity referred to as *physical modeling*, since the literature is prone to associate the concept of physics to the real nature of a material product and to the whole set of its characteristics. The two above cited models have to be connected each other and interoperated. *Interoperability* becomes a crucial task of the MBSE and of tools used to implement it, especially if it is required that an automatic exchange of data is performed.

Safety and security. In case of software engineering, a critical issue of design is the *security of product*, being meant as avoiding any change which might introduce a cause of failure, either due to an uncontrolled modification or to any access to the code not allowed. In many products of industrial engineering, a critical issue is *safety*, being finally interpreted as an assurance that any failure eventually occurring to system is never severe and dangerous for operators and for the integrity of the whole system.

It is worth noticing that safety is assured by standards through the application of suitable margins, when designing the system with respect to some limits and saturations. Nevertheless, a specific demonstration and evaluation of safety and reliability, which describes the probability of failure, are usually performed after that the system architecture is completely defined and sizing is finished. Some suitable tools are usually applied, as the *Fault Tree Analysis* (FTA) or the *Failure Mode and Effects Analysis* (FMEA).

A key issue of the MBSE is that functional modeling, being introduced to help the designer and to be interoperated with the numerical modeling, allows a simultaneous analysis of behavior in presence of failure, i.e. it resorts to a non functional or even *dysfunctional analysis* which can be performed in parallel to the functional one, thus allowing a refinement of requirements and of the system architecture, by considering its behavior affected at least by some typical failure conditions. This task will be herein particularly developed as it looks one of the most challenging issues of the application of the MBSE to the industrial product.

Verification and validation. Another significant concept within the MBSE approach is the so-called *V&V process*, which might be interpreted in some different ways, domain by domain. It can be remarked that the effectiveness of the whole approach has to be *measured* at the end of process, through some suitable *metrics* and by providing some evidences. Therefore, a first step consists of a *verification* being aimed at demonstrating that "you built the system right", according, for instance, to the ASME V&V 10.1 (ASME, 2012). The verification

activity practically should demonstrate that the modeled system corresponds to the original concept developed by the designer in terms of properties, attributes and performance, and it assures a complete coverage of requirements.

After this step a *validation* is performed to demonstrate that the system built corresponds to the models used to predict its behavior and, moreover, it fits the requirements allocated to each part, thus assuring a complete compliance to the customer needs. It should be confirmed that "you built the right system" (ISO15288, 2015). In case of a pure structural system, the difference between verification and validation could be appreciated in Fig. 3.9.

According to the sketch of Fig. 3.9, the verification activity is often based on a *numerical simulation* of the system behavior and it needs the creation of a *mathematical representation* of its geometry. Those models include all the design parameters useful to define the system architecture. A *computational model* is then used to describe the system performance once that the equations associated are solved. A first verification consists in assuring that equations written to describe the model of the system analyzed are correct. Moreover, after some calculations, it is required to verify that these are correct. Those verifications are suitable to demonstrate that modeling activity describes the system conceived, but are insufficient to give an evidence that they can really predict the system performance without a *mock-up*. *Validation* is typically based on the comparison between the prediction performed through the mathematical models and the real behavior of the prototype of the system.

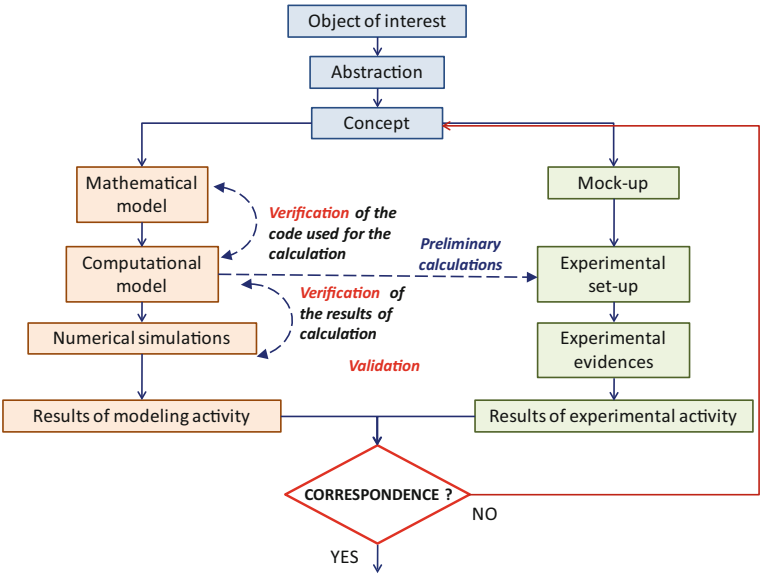


Fig. 3.9 Development of a mechanical and structural system according to ASME (2012)

As it could be appreciated in the above figure, definition of the quantitative modeling activities based on either a numerical representation or a direct construction of a physical mock-up to perform some tests is clearly described and some steps are even included. By converse, the first part of that description, being focused on the *concept of system* and *preliminary definition*, is still quite general and somehow poor, even inside a standard. That step of the system development can be suitably detailed and covered by the MBSE, thus balancing the two main sections of the overall product development.

The industrial baseline for product development. Practically speaking, in case of large and complex systems industry, it is prone to implement the MBSE within the product development by using its methodology in each step and by associating the models even provided by the SE as is shown in Fig. 3.10. It can be appreciated that the typical approach described by the V-diagram is recursively applied to each step. Key sub-processes are the *system integration*, meant as the assembly through a suitable harmonization of components to be compatible and concurrent in providing the required functions. Verification and validation are dominant activities as well as the so-called *Configuration Management*, being aimed at tracking and authorizing the changes all along the system development. Moreover, models described for each column compose the overall tools available to produce the views required by the architecture framework, according to the customer needs. They include different kinds of models and contents, as *requirements* which are based on verbal contents, the SysML or similar representations, based on *graphical meta-models* and finally *mathematical models*, either related to the geometrical description of system and digital mock-ups, or to some numerical analyses. A detailed information for production could be found in *Bill of Materials* or *BOM* and *production plans*.

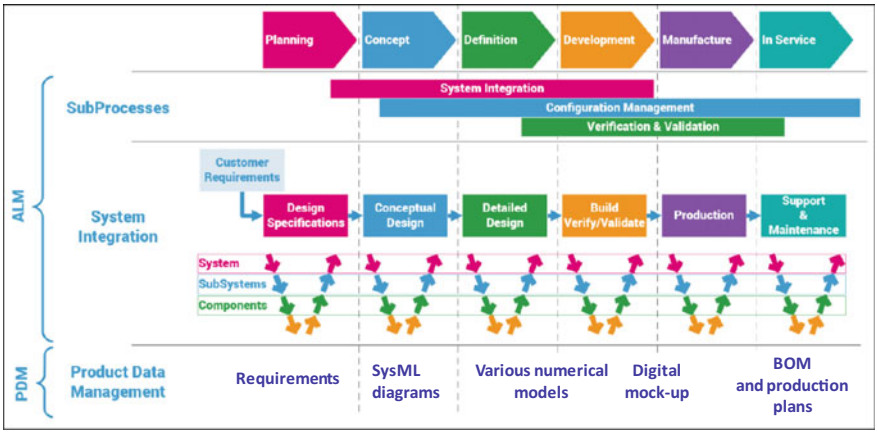


Fig. 3.10 Example of baseline for the product lifecycle development and association with different modeling techniques and artifacts

This handbook is aimed at showing through some test cases how modeling activity is performed by means of tools currently available within the SE and how the functional and physical analyses meet within the so-called *heterogeneous simulation*, through a real interoperability. Moreover, some typical difficulties arising along this process will be highlighted to describe some problems which have to be faced in applying the MBSE to a product coming from a material processing. In the meanwhile, the main benefits of this approach will be pointed out to motivate and instantiate its application to the industrial product development.

3.5 Overview on Known Methodologies to Implement the MBSE

Actually, several approaches have been developed during the last years to support the MBSE and implement the processes. A bright survey of those approaches was provided by Estefan (2008) and continuously revised and updated, for instance by the INCOSE (2017). A brief description of the leading MBSE methodologies is provided, to represent the scenario of interpretations currently available, but even the depth of information provided by the SE Community.

3.5.1 The INCOSE Object-Oriented Systems Engineering Methodology (OOSEM)

The OOSEM was developed since 1998 and further refined by the INCOSE (Friedenthal, Moore, & Steiner, 1999). It looks prone to apply the *V-diagram* as a model of the product lifecycle, but even other similar descriptions. Since 2006, it adopts the *SysML language* to support the specification, analysis, design and verification of systems. The OOSEM is a sort of *scenario-driven approach*, i.e. it integrates the process, being fully *object-oriented*, with the model-based approach.

A key focus is the definition of the system *goals*, *mission* and *operative scenarios*. Those are associated to a precise identification of the *stakeholders* and their *needs*, as well as to an effective classification of *requirements*, being followed by their elicitation. The definition of the *logical architecture* of system, being clearly distinguished from the constructional view, is assumed to be a milestone in applying the SE process, then a synthesis of the candidate physical architecture is performed. The *trade-off analysis* to optimize the system layout upon an evaluation of alternate solutions is a key task of the engineering methods. Moreover, the *validation and verification* of the system are used as essential step of the implementation of this method to assess the design activity. *Reusability* is often highlighted as a priority and, somehow, as a goal of the development.

3.5.2 The IBM Rational Telelogic Harmony-SE

The IBM Corporation developed the *Harmony-SE* process in 2006 (Hoffmann, 2011). It is very well known, even because of the popularity of H.P. Hoffmann, who taught the contents of this approach in several countries as he wrote in "*Systems Engineering best practices with the Rational solution for systems and software engineering—Deskbook*", distributed by IBM.

The process proposed is based on the same principles of the "V" life-cycle model, exploiting an iterative workflow that includes an incremental cycle through the three main phases of requirements analysis, system functional analysis, and design synthesis. The "Harmony process" supports the "Model-Driven Development" (MDD) approach, in which the model is the central work product of the development processes, based on the SysML structure diagrams. From the architectural point of view, a key issue is resorting to some central repositories to store the information and even to share among the users. The purpose of this methodology is to be designed as tool and vendor neutral, respectively. However, IBM provides a support of Harmony within the IBM Rational tools packages, therefore this methodology is often associated by users to the proprietary software tools.

3.5.3 The IBM Rational Unified Process for System Engineering (RUP-SE)

The RUP-SE was also defined by IBM (Cantor, 2001). Actually it extends the Rational Unified Process (RUP) of concurrent design and iterative development for Model-Driven System (MDS), by resorting to the spiral model and including object oriented principles. A main issue is that this approach emphasizes the business modeling, namely the first discipline of iterative tasks, to guarantee the conformance of the system requirements with business activity. It might be appreciated that this specific introduction of the *business model* associated to the product lifecycle deployment is a key issue of the SE implementation, although it is very often neglected in the literature or even in several practical applications. A suitable focus upon the business model to be implemented is a sort of constraint or boundary condition for a straight implementation of the model. Basically, it defines the product in terms of either series or single product (variety of versions, when multiple) time of permanence on the market and service. This methodology is supported by IBM with the RUP SE plug-in Rational Method Composer (RMC).

3.5.4 The Vitech Model-Based System Engineering (MBSE)

Approaching the SE without citing the Vitech Company (www.vitechcorp.com) currently seems as a lack of historical information about the development of the MBSE. Actually, Vitech is promoting a significant action of education, implementation and application of processes and methods of the MBSE. Particularly, the incremental approach to the MBSE involves the *Vitech's CORE*® environment as a system design repository, providing a connection among stakeholders, source requirements domain, behavior domain, architecture domain and verification and validation domain. The used language for technical communication is the System Definition Language (SDL) that is based on relationships, entities and attributes. This approach is supported by the Vitech CORE® suite. The approach is based on a proprietary model of product development, namely the *Onion Model*, organized by layers which allow the user passing from the source documents of the customer to the final specifications by including all the relevant actions of requirements analysis, system behavior, architecture and design, as well as verification and validation.

3.5.5 The JPL State Analysis (SA)

The California Institute of Technology, as the Jet Propulsion Laboratory (JPL) developed a method that defines model- and state-based control architecture. The states describe each condition reached by the system when evolving in time, while the models describe the evolution of the system states themselves (Rasmussen, 2015). This method enables the evolution of the system model during the project lifecycle, through an iterative modeling process and the state-analysis information. They are compiled in a SQL database and ensure the fulfillment of the software requirements, when applied to this product development.

3.5.6 The Object-Process Methodology (OPM)

The Object-Process Methodology (Dori, 2011) is a holistic system paradigm based on Objects, namely things that simply exist, and Processes, which are seen as a transformation patterns applied to the objects. This methodology defines the system development, lifecycle support, and evolution, in three main stages: requirement specification, analysis and development, and implementation. The OPM combines the Object-Process Language, OPL (process-oriented approach) and Diagrams, OPDs (object-oriented approach) supported by the OPCAT software environment.

3.5.7 The Architecture Analysis and Design Integrated Approach (ARCADIA)

The ARCADIA (ARChitecture Analysis and Design Integrated Approach) is a Model-Based engineering methodology for systems, hardware and software architecture design, developed by Thales approximately between 2005 and 2010. Three mandatory interrelated activities are recommended and play the same role in terms of importance: Need Analysis and Modeling, Architecture Building and Validation, Requirements Engineering. The ARCADIA is supported by a standard modeling tool (the Melody Advance/Capella) that relies on the UML/SysML languages (Roques, 2016).

3.5.8 The Systems Modeling Process (SYSMOD)

The SYSMOD Systems Modeling Process (Weilkiens, 2016) is a user-oriented approach for requirements engineering and system architectures. It includes the following activities: stakeholders identifying, requirements elicitation, system context definition, requirements analysis (e.g. with use cases), domain model definition, and system architecture definition by levels (functional, logical, physical). The SYSMOD is tool-vendor independent.

3.5.9 The Alstom ASAP Methodology

Like the ARCADIA the Advanced System Architect Program (ASAP) is expression of the Systems Engineering application to a specific technical domain, as the transportation and railways systems (Ferrogalini, 2015). It was directly developed by Alstom and aims at reducing the system complexity by improving the quality of system specifications. According to this approach, textual requirements are initially deployed on model elements, then are further specified and refined. The modeling approach is focused on a *operational vision*, which deals with objectives and missions; *functional vision*, which concerns the strategy to perform the mission; and *constructional vision*, which applies to elements required to perform the functions. Alstom adopts the standard SysML language to implement the ASAP.

3.5.10 Synthesis About the Methodologies

Previous overview points out that a real standardization was not yet reached within the SE, although several standards already support this activity. By converse, it is

true that a suitable convergence could be found, since in terms of main activities all of the cited methodologies substantially agree, despite of a variety of details in the implementation based on software, theoretical tools and languages. It might be perceived that different points of view, i.e. different thinking heads, assume that the core actions in the SE implementation concern the system as it could be designed, manufactured, and even sold. That variety of interpretations helps at the very beginning in analyzing all the relevant issues of the product development, but creates a number of solutions, poorly compatible with a straight unified approach, to be shared and applied among the operators. A standardization is definitely needed, as it was preliminarily developed and studied by 68 Partners within the European Project ARTEMIS-JU "CRYSTAL", stating for "Critical Systems Engineering Acceleration" which was developed since 2013–2016 (www.crystal-artemis.eu). As it was demonstrated by that project, for assessing a standard procedure to implement the SE within a community of users, a preliminary acceptance of a common process is required. Then, a more difficult agreement about the *engineering methods* is necessary, although the most difficult task usually concerns the operation of a common *platform* building, consisting of a chain of interoperated software tools.

3.6 A Reference Process: The ISO/IEC 15288

As in previous sections some main methodologies were recalled, it is worth noticing that the product development *process* is clearly depicted in the main SE references, but sometimes the activities foreseen at the very beginning are not so trivial like a user could wonder. A straight procedure to catch the real needs of customer, for instance, was never actually assessed, although it is a daily activity of the system engineer. Nevertheless, the contents of the *ISO/IEC 15288 Standard* clearly introduce some major details of process. It describes what is basically required to do for developing a new product. This is the point of view is of the entire enterprise, thus enlarging quite a lot the scenario of roles and competences involved, with respect to the technological process. It suggests a preliminary classification of operations in product development, which includes four kinds of process:

- Enterprise processes.
- Project processes.
- Technical processes.
- Agreement processes.

As a matter of facts, those processes define the main roles of the operators involved in the product development within the company and even some typical areas of competence. It can be appreciated that all the relevant issues are considered, not only the pure technological assessment of product, but even the interaction between manufacturer and stakeholders, as well as financial and management matters (Fig. 3.11).

ENTERPRISE	AGREEMENT	PROJECT	TECHNICAL
Enterprise Environment Management	Acquisition	Project Planning	Stakeholder Requirements Definition
Investment Management		Project Assessment	Requirements Analysis
System Life Cycle and Management	Supply	Project Control	Architectural Design
Resource Management		Decision-making	Implementation
Quality Management		Risk Management	Integration
		Configuration Change Management	Verification
		Information Management	Transition
			Validation
			Operation
			Maintenance
			Disposal

Fig. 3.11 The processes of the systems engineering according to the standard ISO/IEC 15288

The *Enterprise Processes*, for instance, deal with the Enterprise Environment Management, which identifies some procedures to support the product life cycle management. Finances related to the PLM are object of the Investment Management, while the System Life Cycle Process Management is strictly focused on the product. In addition, the Resource Management covers the human and material resources involved, while the Quality Management assures that product fulfills the requirements of quality.

A key role is played by the *Agreement Processes*, which are based on the Acquisition of products and services and on the Supply Process, aimed at delivering the product.

The project management is based on the *Project Processes*, which namely deal with the Project Planning, Assessment and Control Processes. A specific Decision-making Process is listed and includes the actions performed to choose between solutions. In this class appear the Risk Management and the Configuration Management Process, already cited in previous sections. Moreover, an Information Management Process is used to control and share the information among stakeholders.

The *Technical Processes* are composed by eleven items, which are more related to the activities described by the *V-diagram*, for instance, although some of the processes above cited clearly appear in the baseline steps just described. Particularly, the Stakeholder Requirements Definition, the Requirements Analysis,

the Architecture Design cover the first issues of the ALM. The Implementation, the Integration, the Verification, the Transition and the Validation Processes are more related to the right arm of the *V-diagram*. The Operation, Maintenance and Disposal Processes drive the manufacturer in the last activities of PLM.

The same ISO/IEC 15288 Standard identifies the sequence of steps for a generic baseline which looks like:

Concept → *Development* → *Production* → *Utilization* → *Support* → *Retirement*

As it could be immediately appreciated by the Reader, the Standard ISO/IEC 15288 covers all the relevant issues of the product development. Due to its contents, the view offered is larger than the contents of the industrial baseline, since all the actions to be promoted to support those activities are even cited. This specific property motivates the adoption of the standard as a framework to support the implementation of the SE. However, the processes there defined do not allow immediately to access to some helpful drivelines to perform a preliminary definition of system goals and related architecture. This goal is achieved through the *architecture framework*, being able to express several views of the system, thus driving the designer to consider all the relevant issues, by different points of view, like when looking at a physical object.

3.7 The Engineering Methods

As it was previously pointed out, despite the huge development of the SE concerning the model of product lifecycle, the approach and the process, right now a clear standardization about the so-called *engineering methods* aimed at defining how implementing the process was not yet reached. It might be referred herein that a pervasive activity was recently performed in Europe by industry and academy within the frame of several projects to develop a common standardized methodology of SE, including a deeper investigation about the engineering methods to be applied. As an example, the above cited ARTEMIS JU Project "CRYSTAL—Critical Systems Engineering Acceleration" focused on four technical domains like automotive, aerospace, railways and health care to deploy some common glossary, methodology and platforms, based on a tested tool chain, to show, through some industrial demonstrators, the feasibility of such standardization. As the authors could directly realize, an agreement among several users upon the methods is never easy, especially when the technical domain provides a consolidated experience and tradition, being different company by company. Nevertheless, at least at higher level, when a common *Reference Technology Platform* was set up, a convergence could be reached, as it shall be herein briefly described to show the Reader some typical items of the engineering methods and allow catching the difference between process and method.

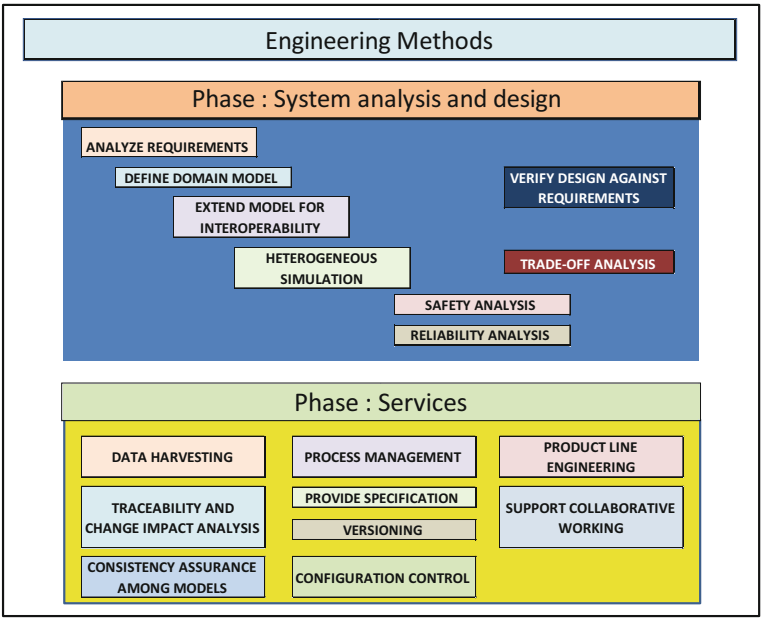


Fig. 3.12 Example of engineering methods defined for the aerospace technical domain within the frame of the ARTEMIS JU EU Program "CRYSTAL"

As Fig. 3.12 shows a straight implementation of the SE process consists of some phases or issues, like those herein described. A first list includes some methods concerning the system analysis and design. Each label actually is associated to a precise list of *activities* to be performed, and it is linked to the use of some *tool* or *language*. It might be appreciated that items on the left side of the dark area should be executed in series, since they concur to the trade-off analysis and the verification described on the right side, being also required inputs for the safety and reliability analyses. By converse, if one looks at the general issue of the common services to be exploited all along the product lifecycle development, labels describe once again some actions, but they are separated activities, to be executed either in series or in parallel in time.

Some key issues are contained in those methods. In the first phase, the effort foreseen to pass from the functional models of the technical domain to a larger simulation including both physical and functional models, i.e., to the so-called heterogeneous simulation is appreciated. This step corresponds to make tools interoperable. A crucial task is the process management, which is based on the use of a platform of tools. Cooperation introduces some issues like security of access and data storing and sharing. Consistency and versioning must be carefully assured through a defined and known strategy. Changes introduced by the allowed users must be listed, motivated, tracked and suitably linked to the continuous updating of the system configuration. As it shall be further discussed in this handbook, tools and

strategy for a configuration change management is strictly required in a collaborative environment for the product development, together with some technicalities of tools to allow the collaborative work.

As soon as one compares Figs. 3.11 and 3.12 a doubt could arise about the similarity of some definitions. Requirement analysis and verification appear in both, but a difference holds. In each process, a main activity is foreseen in terms of people involved, step of development, time to execute the described item. In engineering methods, the same labels directly bring to actions, tools and outputs. As an example the method "Analyze requirements" is here discussed.

A preliminary elicitation of requirements should be already done when the System analysis and design starts. So far, a preliminary critical review of requirements is performed to analyze their contents.

A *pre-condition* for that engineering method is that:

- Requirements were already defined as a result of the elicitation.
- They were stored in a Requirement Management Tool, including their contents and characteristics.
- An additional information about the requirement quality information was stored in a different database.

As an *input* for the analysis each requirement should:

- Be expressed through a native language in some sentences, which clearly describe the statement.
- Include an identification code (ID).
- Indicate the status and the version.

The *activity* of analyzing the requirement according to the defined engineering method consists, for instance, of following actions:

1. The *Requirements Analysis Tool* (RAT) gets the list of requirements from the *Requirements Management Tool* (RMT), by connecting to it first, then launching a request of service, and the RMT eventually applies some filtering to select only the requirements object of the analysis.
2. A list of requirements is assembled and sent to the RTA by the RMT.
3. The RTA receives the requirements, then selects the requirement to be analyzed in detail, then perform the analysis.
4. The RTA eventually modifies the requirement and applies some changes.
5. The RTA launches a service to send the requirement back to the RMT.
6. The RMT receives either a new version of requirement or a new requirement, and stores it.
7. The new requirement is included or the original requirement is updated and records describing version, status or the identification code are accordingly updated.

As a *post-condition* for this engineering method is that each requirement is either approved in its original form or modified.

The *output* of this activity is similar to the input:

- Requirements are expressed through a native language, they declare a statement
- An identification code (ID) is associated
- Version is updated
- State could be now checked
- Further analysis could add some properties like a type or classification.

The above example simply pointed out as the method includes a precise set of activities performed by dedicated tools almost like in a protocol of experiments or an algorithm in calculations. This preliminary work of decomposing the complexity of the system through process, method and model helps in facing some really complex systems, but even some steps of the product development. The system designer should appreciate, as a challenging issue of the SE, the combined activities of functional/dysfunctional analysis and of the physical behavior prediction, being modeled through the functional and physical modeling respectively, as a heterogeneous simulation. Nevertheless, implementing the heterogeneous simulation might look rather difficult without a method. Within the frame of the same project CRYSTAL that issue was investigated and a simplified approach could be included in the description of engineering methods.

As a *pre-condition* for a simulation:

- Models should be available in functional and physical (mathematical) forms.
- Goals of simulation need to be defined in advance, together with a test plan.
- Since models could be developed in different languages, a preliminary screening about the interoperability formats must be provided.
- Each model should be identified by an ID.
- Model version is declared.
- A description of simulation is usually added.
- Properties of the inputs required by the simulation should be provided.

The *activity of heterogeneous simulation* consists in:

1. Selecting first some suitable criteria for each simulation run.
2. Configuring the simulation architecture, thus applying to functional and physical behavior, to the environment.
3. Checking the available simulation environments.
4. Selecting the simulation models (e.g. IBM Rhapsody®—SysML functional model and Simulink® numerical model).
5. Acquiring the simulation environment and resources (computational time or supports).
6. Running the simulation.
7. Collecting the simulation results.
8. Evaluating those results.
9. Eventually debugging the model.
10. Approving results and reporting.

As a *post-condition* of the simulation, several outputs are foreseen:

- Simulation results for a defined set of metrics (e.g. power consumption, power generation, vibration reduction…).
- Model quality reporting and identified issues with simulation.
- Simulation model, with a list of properties representing the results of the simulation.
- Eventually, list of properties defined by the interoperability Standard applied to run the combined heterogeneous simulation.

Similarly, other key engineering methods like verification against requirements or trade-off of system layouts can be specified by following that approach, i.e. through a preventive definition of required actions and artifacts to start, contents of the activity, outputs and additional products.

3.8 The Languages for Systems Engineering

Deploying the process is quite easy once that methods could be completely defined. Nevertheless, if the numerical modeling of systems can resort to mathematics, either to describe its geometry, e.g. through some technical drawings, or to predict its behavior, e.g. in terms of dynamics, stress distribution, electric and magnetic fields configuration or any other typical object of engineering sciences, the functional modeling suffered for long time the lack of a suitable tool to describe all of typical contents of that investigation. Luckily, the development of meta-models based on a graphical and intuitive language of symbols and blocks could help in making the functional model, speaking the designer and communicating the users. A preliminary sketch of some popular languages used within SE is herein proposed. A deeper information could be found later on, when test cases will be fully developed.

It is known that, in software engineering, the design by objects was successfully supported by the so-called *Unified Modeling Language* (UML), which was then elaborated, enriched and adapted to the system design as *System Modeling Language* (SysML). Nevertheless, more recently, new updating and evolutions of those languages are proposed to more properly fit some specific applications and domains. Within the industrial engineering, a lack of entities in the original UML and SysML was clearly found. So far, some research groups are currently working on both the UML and SysML to introduce some additional tools and somehow shaping those standards for a more straight use in mechanics, electromechanics and mechatronics. The *Interdisciplinary Modeling Language* (IML) or the *Automation Modeling Language* (AutomationML) could be examples of those new generation of communication tools. It is true that their main goal is not explicitly the Systems Engineering, but much more the technical domains where are currently developed. Nevertheless, their appearance in the literature of industrial systems engineering was carefully considered as a symptom of incomplete standardization, as the development of the *Lifecycle Modeling Language* (LML), since 2013, can confirm.

Currently, the SysML is still widely used within the international community of SE, thus making suitable to focus the attention of this handbook on it, more than on other languages, like those previously cited. However, a short overview is herein proposed to catch the context and some limitations in the development of the SysML.

3.9 Unified Modeling Language—UML

All of the above cited languages share each other some typical characteristics and contents. They were basically developed within the software engineering to be a general purpose tool to develop the software and modeling. The UML was originally proposed and developed by Booch, Jacobson, and Rumbaugh (2005), since 1994, but in 1997 was defined as a standard by the Object Management Group (OMG), being a worldwide recognized organization within the object-oriented method, which grew up as a appreciated approach to design. More recently, in 2005 it was approved by the ISO as a standard.

A main goal of the UML is making standard a notation which cannot resort to mathematics or any other formalized notation to express some artifacts or objects, as functions, activities, stakeholders or logical architecture of system are. As a language, it provides a defined notation, based on standard diagrams and modeling elements, together with suitable semantics and rules to connect the model elements and give an ordered structure to the described system. Obviously, it was conceived for developing software, therefore the proposed diagrams were tailored to this purpose. Briefly speaking, a clear description through a graphic meta-model of the activities foreseen for the software operation, of its architecture, including single components and routines, of rules for running and of the interactions occurring among elements through some interfaces (and even with the user), was required. Those goals could be reached by defining structured diagrams, whose frame is described in Fig. 3.13. As it looks evident, this structure highlights the behavior and the architecture of the analyzed system, as well as their interactions with time, space and functions performed. It might be immediately appreciated that this

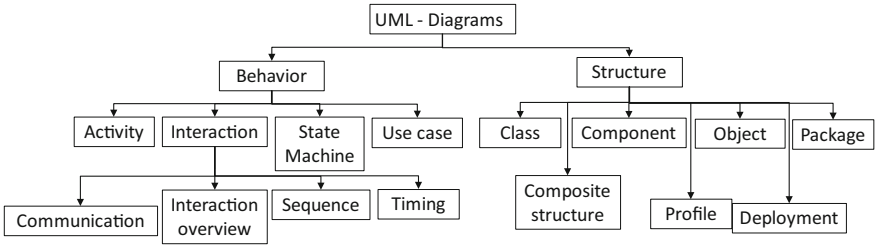


Fig. 3.13 Typical diagrams of the UML

interpretation was greatly simplified and simultaneously enhanced by the SysML, to adapt this language to other kinds and more general systems. Therefore, the meaning of each diagram described in Fig. 3.13 will not be analyzed right here, as most of those diagrams are even used within the SysML, being object of next section.

3.10 System Modeling Language—SysML

The *System Modeling Language* (SysML) is one of the most well-known languages that support the specification, design, analysis and verification of systems. According to the definition of the Object Management Group (OMG), SysML is "*a general-purpose visual modeling language, built up as a UML profile to target systems engineering. However, the SysML does not provide systems engineers a way to model both design alternatives and the design objectives for optimization purposes*" (OMG, 2017).

The SysML was developed since 2001, starting from a propulsion given to this initiative by the INCOSE. It looks like a customization of the UML to adapt the notation to the SE. Moreover, in 2003 a group of industrial managers and tools vendors organized the so-called SysML Partners, born as a spontaneous association aimed at defining a specification for the new language, as it was preliminary done in 2004. In 2006, the OMG adopted the SysML as a standard notation, after a collaborative action of refinement performed by several contributors.

The SysML actually is an extension of the UML previously described, which includes modeling *blocks* instead of modeling classes. It supports the modeling activity of requirements, structure, behavior, and parameters to provide a robust description of a system, but also of its components and of the environment. Particularly, the SysML defines a notation aimed at describing how the concepts of MBSE could be visualized as graphical or textual elements, through a simple diagram. Each block is the basic unit of structure of this model and represents an element of the system. Through a set of diagrams, being conceived to give several points of view on the same system, this language allows modeling the system structure and behavior, as well as the requirements definition and the specification of some performance parameters.

Four main typologies of diagrams can be distinguished within the SysML, namely:

- *Requirements diagrams.*
- *Structure diagrams.*
- *Behavior diagrams.*
- *Parametric diagrams.*

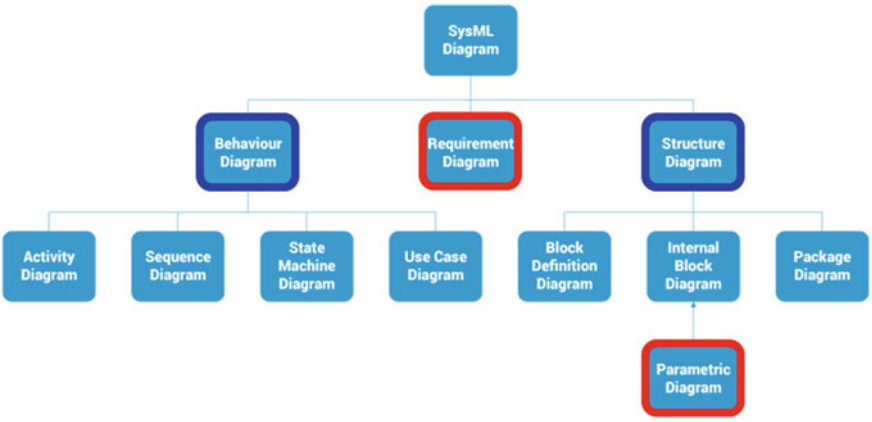


Fig. 3.14 Diagrams of the SysML

In addition, behavior and structure diagrams include several other ones, aimed at showing different contents of the whole information to be collected about the system definition.

It can be remarked that requirements and parametric diagrams were introduced in addition to those already present within the UML. They play a relevant role to connect the system model to the preliminary elicitation of requirements and, on the other side, to a quantitative modeling of the system behavior and architecture, aimed at performing a numerical simulation. A sketch of the structure of diagrams foreseen by the SysML is depicted in Fig. 3.14.

Since those diagrams are widely used to implement the process of SE, a brief and preliminary description is herein proposed, while a direct application to the test cases shall show better their peculiarities and details.

3.10.1 Requirements Diagram

The SysML defines a visual and graphical representation of textual requirements usually collected in form of sentences and classified by a requirements manager tool separately. To integrate and allocate the requirements, a direct image within the digital model of the system is needed. This diagram allows introducing the requirements inside the domain managed through the SysML and performing some associations between themselves or with other model elements, managing their changes, through a structured and hierarchical environment. As it shall be deeply described in next chapter, each requirement must be allocated and satisfied, and shall be described by a textual statement. It is worth noticing that each requirement is numbered and referenced, with attributes and properties. The requirements diagram is even used to clearly define the requirements *hierarchy*, how they are

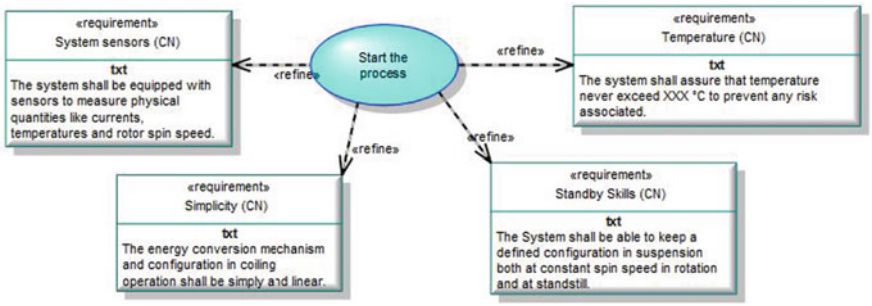


Fig. 3.15 Example of contents of the requirements diagram

derived, satisfied and verified, thus providing a suitable gate between the requirements elicitation and the system modeling. Figure 3.15 shows an example of requirements diagram, where each block includes the requirement identification and statement as well as a link to the use case to which is associated.

3.10.2 Behavior Diagram

The system behavior is modeled within the SysML by emphasizing all the relevant inputs and outputs, to define functions, goals, interfaces and actors, through a sequence of actions, operating conditions, and even as a function of time. The so-called behavior diagrams include the *use case*, the *state-machine*, the *sequence* and the *activity* diagrams.

Use Case Diagram (UCD). A *use case* is a description of observable results assuming a measurable value, for one or more actors, when using the system or interacting with it. The Use Case Diagram (UCD) basically describes the neighborhoods of the system, being highlighted as a rectangle in the middle of the diagram (Fig. 3.16), how the system can be used and operated, through several modalities represented by elliptical call-outs, and the interaction between the system and some external actors, to be referred as *stakeholders*, in relation to each use case. Stakeholders can be human users, depicted as a dummy (e.g. the cockpit crew in figure) or even other systems, directly interacting with the system analyzed, being depicted either as a dummy or as a rectangle.

State Machine Diagram (STM). The *State Machine Diagram* (STM) depicts the discrete behavior of system, through some finite states, reached after a transition induced by a dynamic sequence of events, during which operating conditions change. Each transition is usually activated as the system reaches a certain measurable threshold of operation or behavioral condition, thus leading to the occurrence of a defined *event*, while invariant condition is held for a given *state* of system operation.

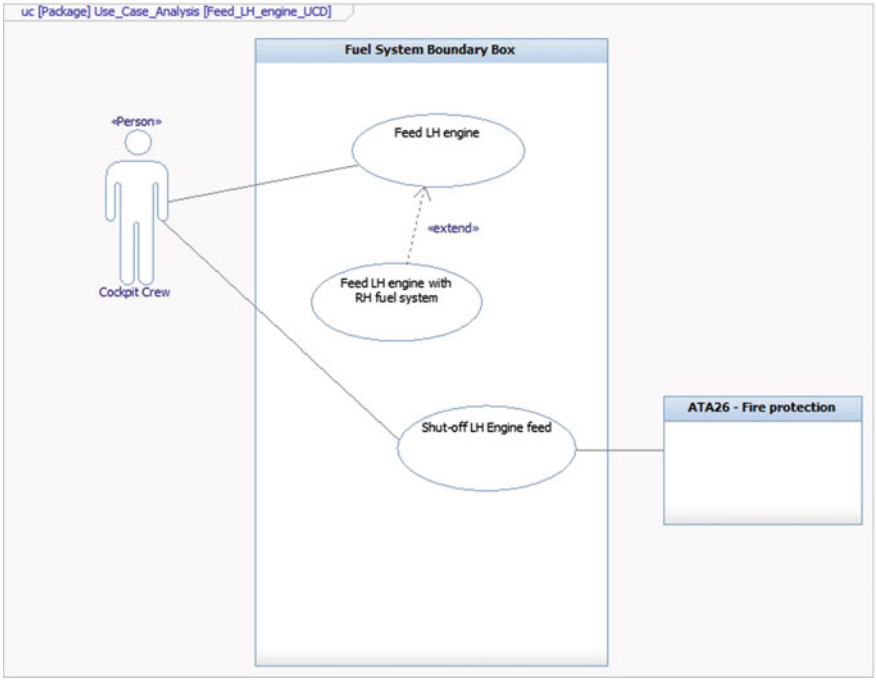


Fig. 3.16 Example of use case diagram

In Fig. 3.17 a starting operating condition is defined on the upper part of the diagram, as an entrance, and is worth noticing that the system depicted shall come back to that point, to exit, at the end of the whole cycle of operation foreseen for the depicted use case. States correspond to the labels, which catch the system operating in a temporary stable condition, while arrows describe a transient condition, with variable parameters which correspond to an event, motivating the switch between two subsequent states. In some case the state includes other sub-states which are activated in sequence, as in Fig. 3.17 the inner rectangle.

Sequence Diagram (SD). This diagram describes the interaction among the *actors* involved in the system operation, as a function of *time*, by focusing on the sequence of either *messages* exchanged or *actions* applied between actors and system. The events occurrence is depicted along the lifeline of system operation, therefore the diagram shows how processes operate and in which order.

Figure 3.18 shows an example of Sequence Diagram. Several actors are involved in the described use cases and appear on the upper part of the diagram, like heads of columns. Time is assumed to be running from the upper to the lower part of the diagram. Therefore, each horizontal arrow describes a message, or an action or even a flow of material or power from one actor to another one and to the system, being expressively located on the right side as last column of the diagram. If the

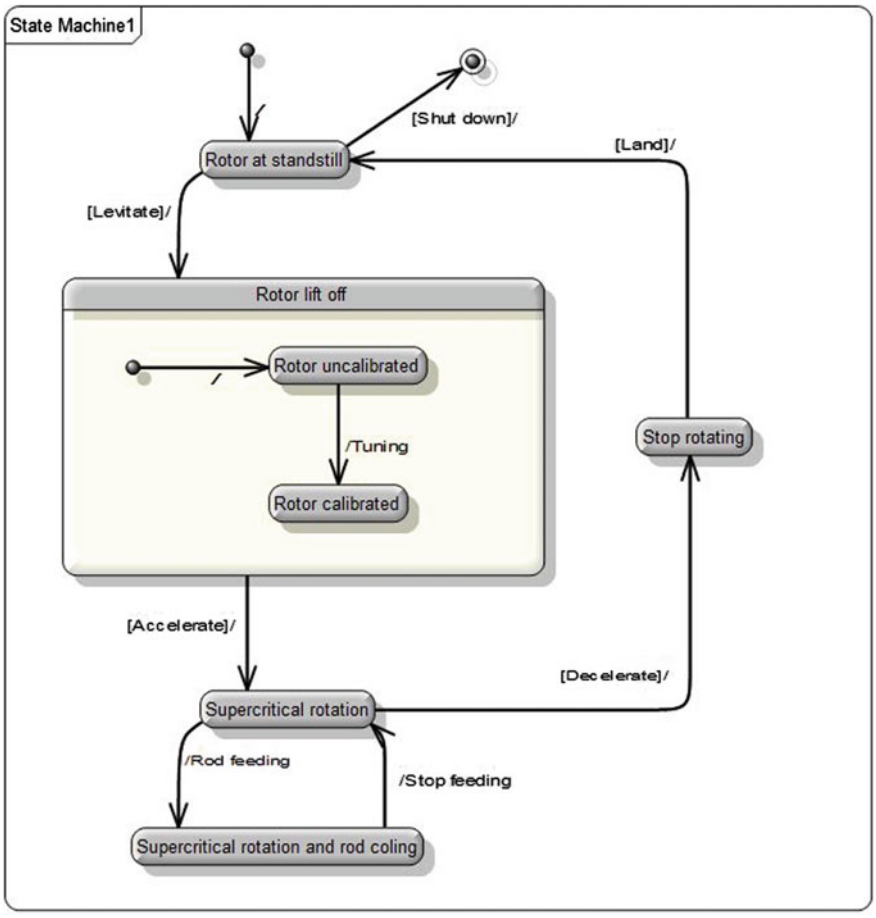


Fig. 3.17 Example of state-machine diagram

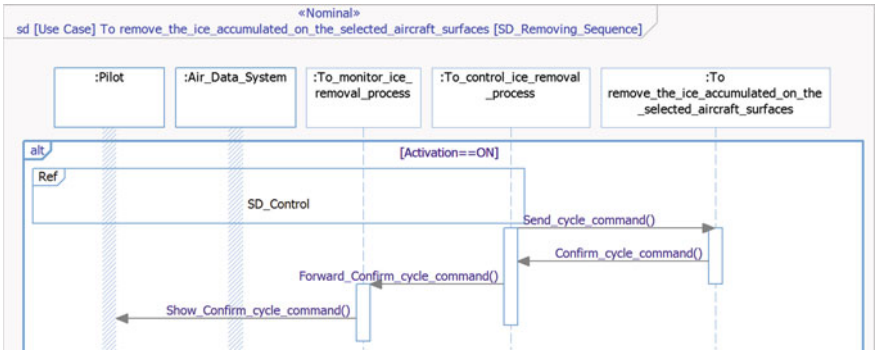


Fig. 3.18 Example of sequence diagram (detail)

diagram is read from the top to the bottom, a sequence of operations is identified both in terms of actor and time at which each one is performed.

Activity Diagram (AD). In this case the main subject is the system itself and its behavior is associated to *activities* and *operations* performed through a sequence, which is here uncoupled with a specific evaluation of time, while it is detailed whether each activity is simultaneous (*parallel tasks*) or subsequent (*tasks in series*) with respect to any other one, thus identifying inputs, outputs, interfaces and flows. Some partition groups could be defined as a set of activity or operation nodes, based on a set of criteria. They often correspond to some organizational units. It might be noticed that all of activity diagram definitions already used in the UML even apply to the SysML.

In Fig. 3.19 a simple example of Activity Diagram is proposed. Action starts at the upper point, each activity looks like a label inside the diagram, while arrows describe the flow of activities. They are performed in series considering the upper, middle and lower blocks, while are done in parallel in the middle set of activities. In this example, a logic action is foreseen after the three blocks in parallel and allow deciding whether the last activity should be started or a loop is required.

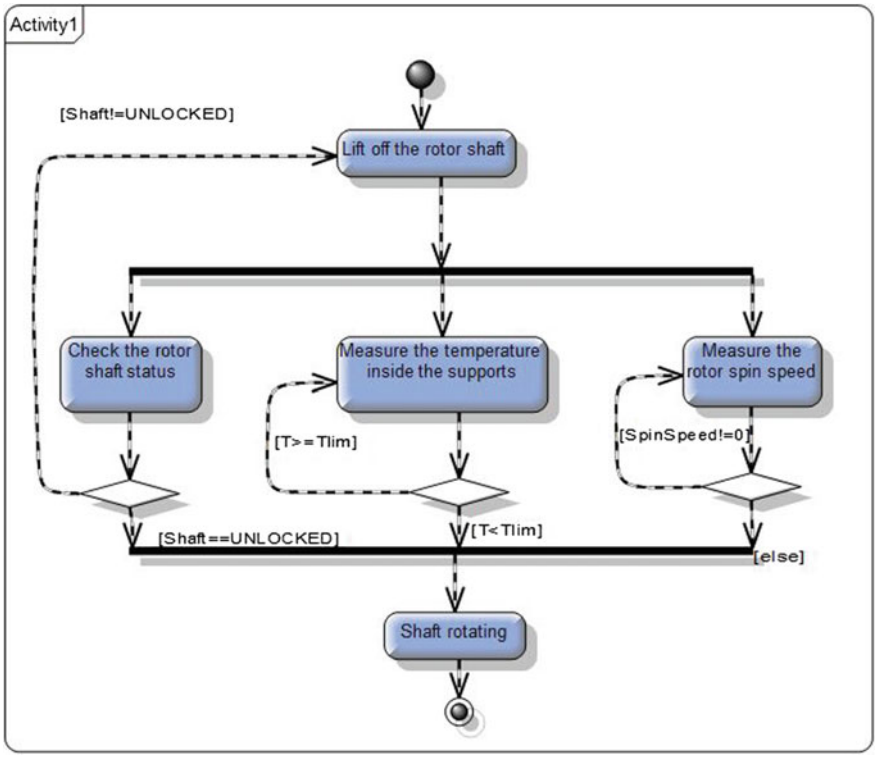


Fig. 3.19 Example of activity diagram (detail)

3.10.3 Structure Diagrams

The *architecture* of the analyzed system is investigated through the structure diagrams, sometimes referred to as architectural or constructional. They include the *Block Definition Diagram* (BDD), the *Internal Block Diagram* (IBD) and the *Package Diagram*. The *block* is the fundamental modular unit, being always precisely defined and used to introduce a logical or conceptual or physical entity, like hardware, software, data or even a person or a facility. In some cases, the block corresponds to an entity that flows through the system or is located into its natural working environment. Each block includes principally attributes (state variables), operations (behavioral procedures), constraints, ports (exchange of flows with the external world), and parts (sub internal blocks). Blocks are often used to describe reusable components, i.e. they can be used in many different systems just by adapting the model developed through the SysML.

Block Definition Diagram (BDD). This diagram provides a black-box representation of a system, as a composition of a main block connected to a set of other blocks, through a well-defined *hierarchy*. Main focus of that description are features and *structural relationships* of blocks composing the diagram. This approach leads to describe the *structure* and the *architecture* of the analyzed system.

It is remarkable that, in comparison with the UML2, the SysML BDD redefines the class diagram by replacing the classes with blocks and introducing the flow ports. The flow port is a new definition, typical of the SysML. It describes what can pass through a block (in and/or out), i.e. data, matter, or even energy… In the example depicted in Fig. 3.20, the structure of a rotor on magnetic suspension is

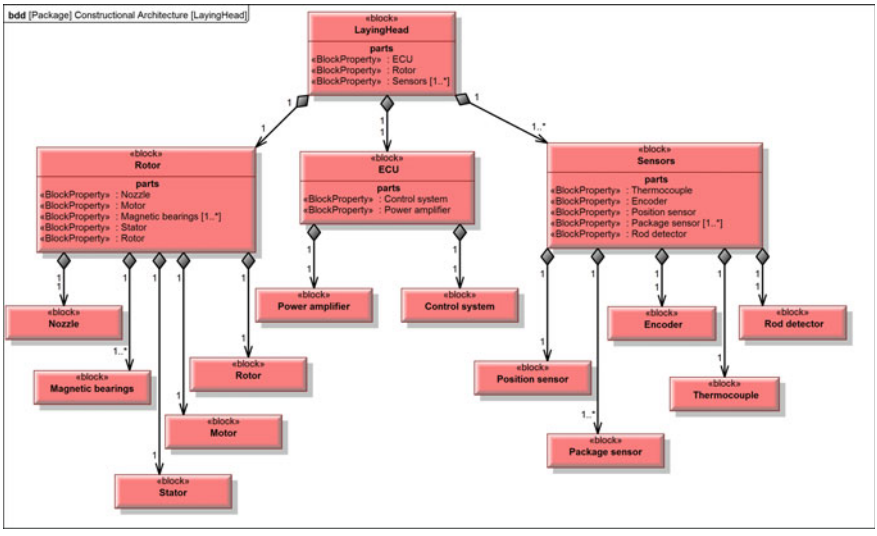


Fig. 3.20 Example of block definition diagram

described as a tree of main modular elements composing the whole system. Interactions mainly flow from the main module towards the smaller components.

Internal Block Diagram (IBD). It is mainly used to define the internal structure of a system. It describes how the internal parts are interconnected, by resorting to *ports* and *connectors*, and what kind of *flows* are activated between them. Actually it depicts the internal structure of a block, by clarifying the interactions occurring with some other parts of the system and specifying the contents of a block of the BDD. If one compares this diagram to the same used within the UML2, in this case the SysML IBD redefines the composite structure diagram by supporting *blocks* and *flowports*.

As an example of IBD, Fig. 3.21 shows a simple diagram related to a single use case. It is meant that the left side actor inputs a message to start the process. A check is performed. Functional elements are described by parts, whilst arrows on the border of each part define the flow direction. Links are used to connect the elements. It is worth noticing that arrows are encapsulated into defined ports, being pointed out through some small rectangles. Different functions are allocated to those parts, and therefore the hierarchy of operation between functional blocks is clearly stated. The actor on the right side receives the output of this activity (Fig. 3.21).

Package Diagram (PD). The aim of the Package Diagram is limited to the modeling activity itself as it represents the internal organization in terms of packages and elements. Using the Package Diagrams allows organizing the model in several views and to analyze the system at different levels of abstraction. The same diagram is also used within the UML language (Fig. 3.22).

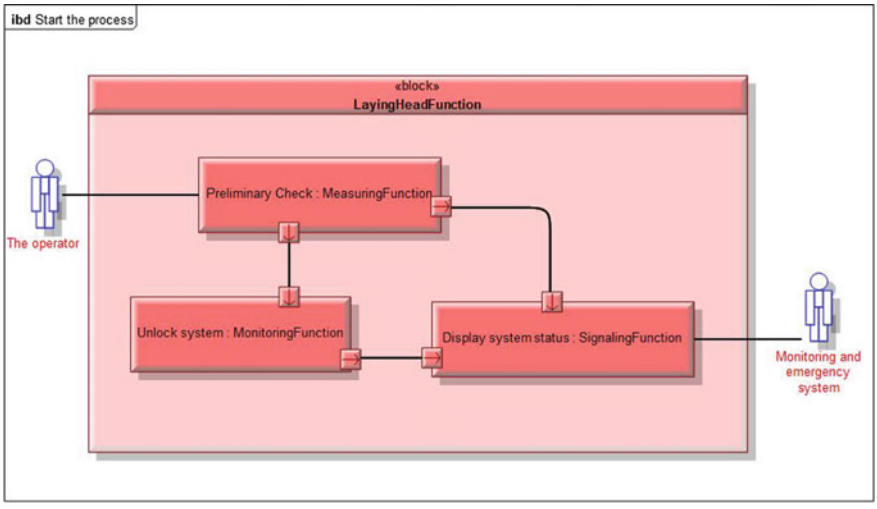


Fig. 3.21 Example of internal block diagram

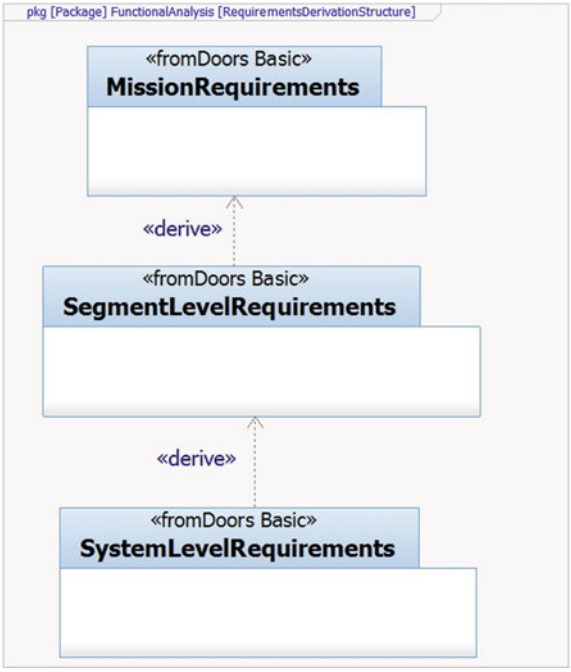


Fig. 3.22 Example of package diagram

3.10.4 Parametric Diagram

The Parametric Diagram is used to model some constraints that affect the value properties of blocks. It enables the integration of SE with the design modeling. It defines the constraint properties and constraint parameters, as well as their mutual relationships. In practice, it might be used for a preliminary quantitative analysis of system performance. It is not present in the UML.

In Fig. 3.23 the proposed example shows the classic example of calculation of deceleration of a motor vehicle while breaking, starting from the breaking equation, passing through the relation between acceleration and force, then to the computation of the required distance to stop the vehicle. Each block receives some parametric inputs, elaborates a calculation and provides some preliminary results, to verify the requirements allocated. Links are used to describe the path followed and blocks to represent rules and computations.

All those diagrams allow modeling the system and perform the functional analysis, according to the methodology applied. Nevertheless, it is clear that a more quantitative modeling could be required for a deeper investigation, being provided by several numerical models available in the engineering science. When the two

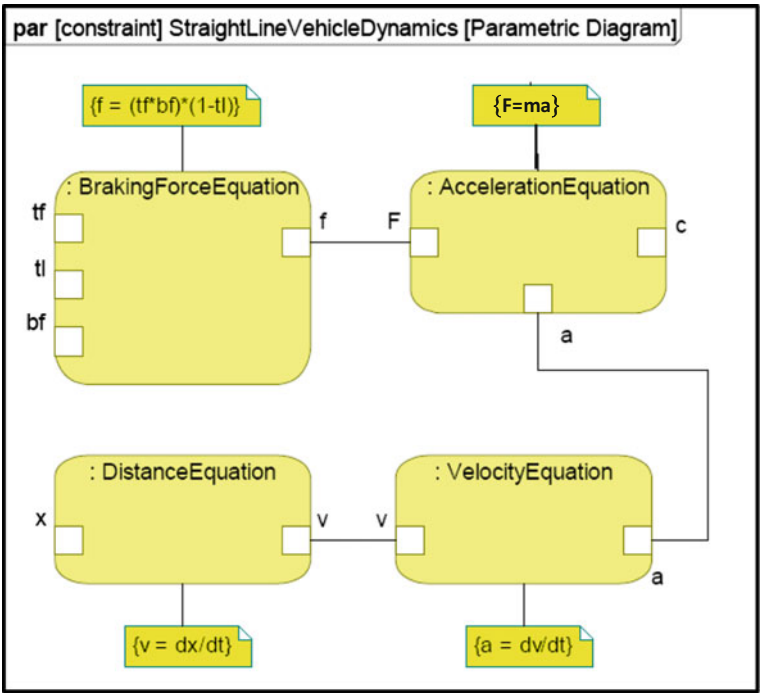


Fig. 3.23 Example of parameter diagram

worlds of functional and numerical modeling, or better physical in this context, are effectively connected to share data and information, and both are linked to requirements, a complete heterogeneous simulation of the system behavior can be performed and the product design finally assessed.

References

ASME. (2012). *An illustration of the concepts of verification and validation in computational solid mechanics.* ASME V&V 10.1.

Booch, G., Jacobson, I., & Rumbaugh, J. (2005). *The unified modeling language user guide* (2nd ed.). Addison Wesley.

Cantor, M. (2001). *RUP SE: The rational unified process for systems engineering*. The Rational Edge, Rational Software.

Dori, D. (2011). *Object-process methodology: A holistic systems paradigm.* Springer Science & Business Media.

Douglass, B. P. (2016). *AGILE systems engineering*. Waltham, MA, USA: MK Morgan Kauf-mann.

Estefan, F. A. (2008). *Survey of the model-based systems engineering (MBSE) candidate methodologies.* The INCOSE MBSE Initiative.

Ferrogalini, M. (2015). MBSE in rail transportation—Product families and product lines. *INSIGHT INCOSE*.

Friedenthal, S., Moore, A., & Steiner, R. (1999). *A practical guide to SysML, the system modeling language.* The MK/OMG Press.

Hoffmann, H.-P. (2011). *Systems engineering best practices with the rational solution for systems and software engineering, deskbook (model based systems engineering with rational rhapsody and rational harmony for systems engineering).* The IBM Software Group.

INCOSE. (2017). www.incose.org.

ISO15288. (2015). *Systems and software engineering—System life cycle processes.* ISO/IEC/IEE15288.

Nordqvist, T., Edberg, A., Byström, H., & Gustfsson, M. L. (2006). *Systems engineering, architecture frameworks and modeling and simulation*.

OMG. (2017, May). *The OMG systems modeling language.* vs. 1.5: http://www.omg.org/spec/SysML/1.5/PDF/.

Rasmussen, B. (2015). *Session 1: Overview of state analysis (internal document)*. Pasadena, CA, USA: Jet Propulsion Laboratory, California Institute of Technology.

Roques, P. (2016). MBSE with the ARCADIA method and the Capella tool. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016).* Toulouse, France.

Weilkiens, T. (2016). *SYSMOD—The systems modeling toolbox—Pragmatic MBSE with SysML.* Lulu.com.

www.crystal-artemis.eu. (2016). CRYSTAL EU project: www.crystal-artemis.eu.

www.vitechcorp.com. (2017). Vitech Corporation, Primer on the Model Based Systems Engineering.

Chapter 4
Systems, Customer Needs and Requirements

Abstract A preliminary and relevant step of the whole process includes a complete collection of customer needs and a suitable elicitation of requirements. This activity is never trivial and requires to be suitably formalized. This is done by resorting to the test cases of this handbook. In the meanwhile, the role of stakeholders is even investigated and defined. Requirements are then defined, classified and categorized for a better implementation of the MBSE. A direct implementation of requirements analysis is performed for the two examples.

4.1 A Couple of Examples to Understand

The Systems Engineering includes so many contents and practical issues that one might find difficult understanding its deployment without a direct application to a intuitive test case. Therefore, in this handbook, two real applications are proposed to show the Reader how the discussion of some main contents of this approach can be easily understood, even in case of a product coming from a material processing, as it happens in mechanical and mechatronic engineering.

The two cases selected have a different profile. One could be referred to as *didactic*, since it comes from a real industrial application, but it was developed just preliminarily, i.e. only up to define together with the customer some general issues of design. It might be considered as a simplified model, but it looks suitable for a preliminary understanding of topics explained in the handbook. By converse, a second test case, to be referred to as *industrial*, was selected to show some typical issues of the system integration within an industrial framework, aimed at providing a system model suitable to be assessed in tight cooperation with all of partners of a manufacturing consortium and to deal with the actual depth of technical problems faced during the product development.

The didactic test case is basically a rotating machine for shaping the steel wire rod produced by a rolling mill. The industrial case is related to the integration of an Ice Protection System (IPS) on a civil aircraft for passengers' transportation. Those two examples were actually developed by the authors in a real industrial context,

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_4

thus experimenting some crucial issues of the MBSE activity. Moreover, in case of the IPS, the main issues are the design and integration of a system within the larger complexity of the aircraft. The behavior of this system is typically related to the ice accretion and either how ice can be cracked and expulsed (in *de–icing*) or how its formation upon the aerodynamic surfaces can be avoided (in *anti–icing*). Two interesting features of this example concern the need of a trade–off among different technologies for selecting the system configuration and the mechanism of anti/de–icing and the strong interaction between the modeling of the system and of the environment in which it works, being related to both the flight and the weather conditions. By converse, the rotor, used as a coiling system for shaping the steel wire rod, shows three interesting topics. It is clearly a system connected to a system–of–systems like the mill plant is, thus exciting some investigations about the interaction among subsystems. The rotor exhibits a structural behavior which must be foreseen and properly simulated, through a *physical modeling*. Solutions proposed in that case were strictly related to structural mechatronics and active vibration control, thus giving the possibility to investigate the real meaning of smartness and some typical requirements associated.

Considering the methodological contents of those examples, in the didactic one a clear reusability can be exploited if, for instance, the rotor is assumed to be actively suspended through magnetic bearings and, instead of the coiling system, the model is applied to a flywheel for energy storage. In the industrial test case a significant issue was the final process of SE, implemented to cope with the different approaches applied by several partners of the consortium involved in this design activity.

4.1.1 Didactic Test Case: A Coiler for Wire Rod Production

Several mechatronic solutions are currently investigated to reach a higher level of innovation in industrial equipment, by resorting to a smart energy conversion. A *mechatronic* approach is just applied to deploy the MBSE in the test case of the coiling system for steel wire, even and simply referred to as *coiler*. This example can show how the intrinsic complexity of mechatronic design of systems is nowadays often faced through the SE. Even the meaning of *smartness* requires often to be clarified and suitably interpreted. It involves the dynamic behavior of systems.

Some preliminary information about the operation of storage of the steel wire rod applied to the steelmaking plant are introduced in Table 4.1, to describe some general details of the system.

The technical problem can be briefly described as Fig. 4.1 shows. Wire rods and bars are cut and coiled at the end of the steelmaking plant, after rolling, in the finishing area, just before to be stored and delivered to customers. The wire rod reaches the end of the line fairly fast to assure a large production per hour, up to almost 130 m/s. Within few meters it is required that it stops and is completed coiled. Basically, the translational motion has to be converted into a rotational

Table 4.1 Typical properties of the coiler system for steel wire rod

Description	Value	Units
Maximum speed	130 (current record) to 150 (perspective)	m/s
Diameter range of wire rod or bar	4–65	mm
Utilization of the product line	Up to 90% of time	
Coil maximum weight	3 (min diameter) to 4 (max diameter)	tons
Coil maximum outer diameter	1–1.25	m
Rolling maximum temperature	900	°C

http://smt.sandvik.com/it/applications/wire-rod-rolling/ as is on December 11th 2016

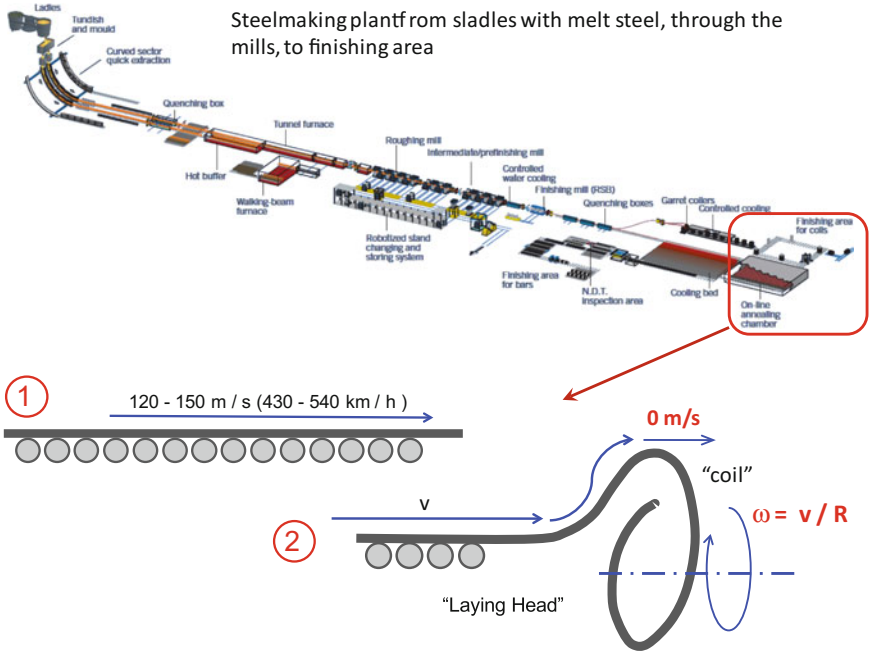


Fig. 4.1 Principle of coiling of the steel wire rod in rolling mill

motion, before stopping. To perform this action, a so–called *laying head* pipe is used, thus allowing to shape the wire rod into a coil and transforming the translational speed v into a rotational speed ω, which depends on the radius of coil. Practically speaking, a rotating and tubular shaft is equipped at one end with a sort of conical head to allow the wire rod inletting into the rotor and coiling over the surface of the conical head, thanks to the effect of rotation at fairly high speed. To assure a continuous delivery, the circumferential speed of coils should be at least equal to the translational speed v. Coils are then stored and transported to the stock, before delivering. The *laying head* is a crucial device in this operation, since it

rotates quite fast and is largely loaded, thus undergoing some operating conditions which induce a severe wear of materials and unbalance for the rotor and the suspension system, to be carefully designed and monitored in service.

4.1.2 *Industrial Test Case: De-icing or Anti-icing System for a Commercial Aircraft*

The industrial example belongs to the aeronautical engineering. It concerns the development of the Ice Protection System of a commercial aircraft classified as Very Light Business Jet (VLBJ). The process herein proposed is aimed at showing only a preliminary high level development of this subsystem, but the depth of analysis will be sufficient to deploy the MBSE approach as it does within a company. It has to be noticed that the example describes some typical issues of this activity, but it does not correspond to the real certified process deployed by the manufacturer nor by the suppliers, since they are covered by non-disclosure limitations. Nevertheless. some typical requirements of the whole aircraft can be introduced in Table 4.2 to develop the models of the de–icing or anti–icing system.

The above mentioned aircraft must fulfill the requirements of the Airworthiness Agency to receive the Airworthiness Certificate before entering into service. The de–icing system is strictly required to operate the aircraft during the flight in all of environmental conditions, seasons and altitudes.

Moreover, a selection of technologies can be evaluated to design this system, basically by resorting to a pneumatic device equipped with inflatable boots, electrical systems based on electro–thermal energy conversion, or even on a heat exchange realized by driving a flux of warm air near to the aerodynamic surfaces.

Table 4.2 Main properties of the Very Light Business Jet (VLBJ) used as an example for the deployment of the MBSE process

Description	Value	Units
Maximum Take–Off Weight (MTOW)	Up to 4500	kg
Autonomy (maximum operational range)	Up to 2000	km
Take–off field length	Up to 1100	m
Landing distance	Up to 1500	m
Ceiling	12500	m
Never Exceed Mach	0.65	Mach
Thrust (for each engine)	Up to 7	kN
Max Payload	Up to 600	kg
Passengers	Up to 4	
Pilots	2	

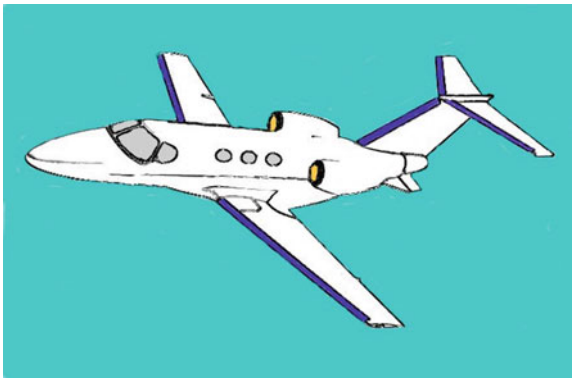


Fig. 4.2 Rough sketch of the Very Light Business Jet (VLBJ) used as an example for the deployment of the MBSE process

This selection requires a preliminary functional and operational analysis to define limitations and performances of each solution.

As Fig. 4.2 shows the de–icing system basically is composed by some devices distributed on the aerodynamic surfaces of the leading edge of wings, horizontal and vertical stabilizer, as well as on engine intakes. The interaction with other systems is quite critical, since a certain amount of power should be used, reducing the level available for all the other functions and for propulsion and to avoid the ice accretion or to remove the ice.

4.2 Implementation of the MBSE

In the following sections and chapters, the two above mentioned test cases will be used to provide a direct example of implementation of the MBSE. Obviously, the main figures described in the above sections are usually found as a result of the exploration, although in this case they were disclosed to make the Reader aware about the technical domains of the proposed examples. By the way, a challenging issue in the implementation of the MBSE consists expressively in avoiding as much as possible to resort to some known set of parameters, which describe the state–of–the–art of the product object of the development process, and in making the process as far as possible blind with respect of the tradition of the technical domain.

The didactic test case will be developed by resorting to some software kindly supplied by PTC, to develop the requirement analysis and the whole ALM activity, up to the design synthesis. A mechatronic solution based on the active magnetic suspension technology will be developed and its limitations and advantages even discussed. In this example, rather than the trade–off activity, the system integration will be investigated and the verification, validation and testing, to be performed through both numerical and experimental methods, will be considered. As it was previously described, a short overview on a straight reuse of models for a different device aimed at storing the energy through a flywheel system will be introduced.

The industrial case will be used first to describe the customer needs and the standard requirements in a large production, then to investigate the steps of both the functional and physical analyses. Some software tools will be used, particularly the dedicate tool kits kindly supplied for this redaction by the IBM Ltd. In addition, some examples of interoperation with Matlab/Simulink® will be even discussed. The main goals of this implementation concern the trade–off of the system configuration, the refinement of requirements and the integration within the aircraft model, with the aim of exploring some issues related to safety and reliability.

4.3 Identification of the Customer Needs

4.3.1 Needs Versus Requirements

As the so–called "V–diagram" even shows, a first action performed in deploying the MBSE is a bright identification of the *customer needs* and of all the *stakeholders*. How this action can be done looks never trivial, while is crucial for the following steps of product development. A typical mistake in applying this methodology consists in assuming that needs and requirements are coincident. Actually, *needs* define the product to be *validated* at the end of process, while *requirements* specify, at different levels, their implications in terms of functions, operations and architecture, to cite the most important issues, in relation with several boundary conditions, and particularly the technical standards. It could be told that customer needs are somehow expectations roughly expressed, while requirements are like a translation into a precise, complete and unambiguous technical communication written in a proper language. Requirements have to be *allocated* on functions, then on subsystems, components and parts. A clear *traceability* is assured, but elicitation of requirements might be assessed in some steps, through a refinement which can be performed by resorting to the functional and physical analyses. A unique action of requirements elicitation is never complete nor possible, in presence of complexity, therefore a recursive approach has to be implemented. This process usually drives first to a preliminary design of product, then a further investigation of detailed requirements can be performed.

4.3.2 Looking for the Customer Needs

The customer needs are often identified through a direct *interview*, but even through a preliminary definition of a *commitment document*, like in technical domains related to defense. A standard guide to catch the needs of the customer is not yet assessed, although some experts are able to follow a main driveline to select and collect the real needs among a number of preliminary data. Nevertheless, the

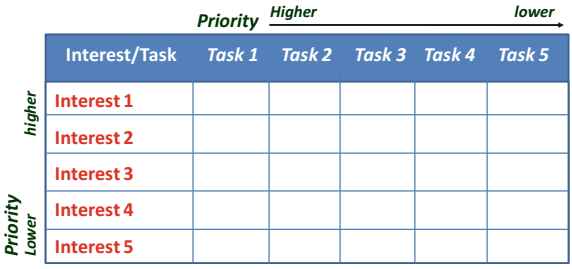


Fig. 4.3 Example of classification of interests per task in needs identification

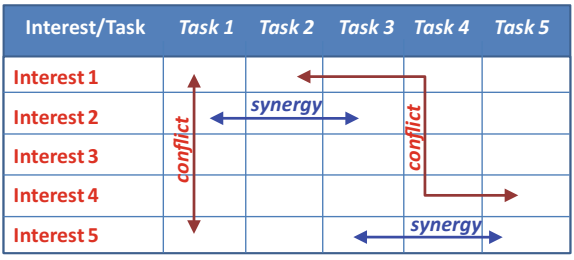


Fig. 4.4 Example of conflicts and synergies in interests of stakeholders

architecture frameworks previously described can help in defining the needs by looking at the *system capabilities* to be exploited. They can be explored by resorting to an assessed approach like the UAF, through the *matrix of capabilities*. However, if a commitment document is not required, a simpler description can be performed by connecting each task of the interaction between stakeholder and system to the real interest associated, through a classification which points out the level of priority perceived. The above mentioned approach might lead to a structure like that depicted in Fig. 4.3.

If the different tasks and interests respectively are analyzed, the needs could be better identified. Moreover, potential conflicts or synergies might be even detected (Fig. 4.4).

4.3.3 A Systematic Approach to the Identification of Needs

It should be immediately realized that this action could be poorly driven by a precise list of possible items. Nevertheless, this is never true, if a *business model* for the product under development is foreseen. Particularly, it is important to distinguish:

- New version of an existing product and completely new systems never appeared on the market.
- Single product from production in series.

- Quality products aimed at satisfying a quite selected audience of customers, perhaps even keeping the price not so low, such as for mass products, where quality is related to the number of series manufactured but at lower cost and price.
- Time to market and time on market of this product.
- Service, maintenance, distribution items.
- Level of connectivity after market with manufacturer.

All those options could identify a certain target of market, which is related to some specific needs, being typical of a selected population of customers. In this selection, there are some relevant issues to be considered, that might help the identification of needs.

As it was recently expressed in some popular examples (Michelli, 2016) there are some key issues to be considered in this detection of customer expectations.

- Assumption that *customer needs are essential* in product development and never aside of the product lifecycle.
- Creation of a *vision about the customer experience*, by defining the current state of a product, the new expectations and the future targets.
- Interpretation of the *urgency of a given expectation* to assess a suitable time to market and assigning a *priority* to the need.
- Consideration of the *impact of that need on the current organization* of the design and production.
- Direct *confirmation of the stakeholders* about the consistency of the customer needs identified.
- Assessment of a *systematic approach* to innovate the product according to the customer needs (and the SE is this kind of approach).
- Listing and consolidation of the needs defined by *writing* all for next steps of development.
- Simultaneous identification of some *metrics* to check the correspondence between the final product assessment and the customer needs through a detailed validation.

Those items may be used to drive the interview to the customer. In case of a product coming from a material processing, obviously, some typical issues to be explored concern:

- *Layout* and *architecture* of the new system.
- *Materials.*
- *Context* where the system shall be operated, mission, scenarios and goals.
- *Quality* of product.
- *Safety* and *security*.
- *Cost*, *feasibility* in manufacturing and *sustainability*.

4.3.4 Source of Needs

The criteria above mentioned meet a second kind of categorization which looks helpful in revealing the level of importance of each need and the possibility of negotiation.

Needs are basically related to:

- *Customer.*
- *Technical standards.*
- *Constraints*, typical of the technical domain or of the manufacturer's practice.
- Key features to innovate the product (*innovation targets*).

The priority of needs is consequently defined. The customer expectations and the innovation targets have the highest priority because of their strategic effects, but are often negotiable. The practice of the technical domain looks somehow less influent, but in several cases it represents the state–of–the–art of that industrial field, thus introducing a number of limitations, which might be even and simply induced by the constraints imposed by the facilities owned by the manufacturer. The requirements of technical standards play the role of law, which must be fitted by the manufacturer, without exception, because of the *product liability*.

The above mentioned identification of need source can help in perceiving the level of innovation associated to the new product. A very rough interpretation could see in each innovation target a strong pull effect towards a new generation of the designed product, while constraints imposed by the technical domain or directly by the manufacturer look like a brake to a straight innovation activity. In some cases the customer needs are quite different from the known requirements of standards, which might represent the tradition. Provided that this interpretation cannot have an absolute impact on the actual indexation of the added value of the new product, a preliminary impression could be get from the number of needs detected in each of those four categories and depicted like in Fig. 4.5. That representation allows measuring the balance between innovation and tradition in the system development, simply by comparing the number of needs related to each type.

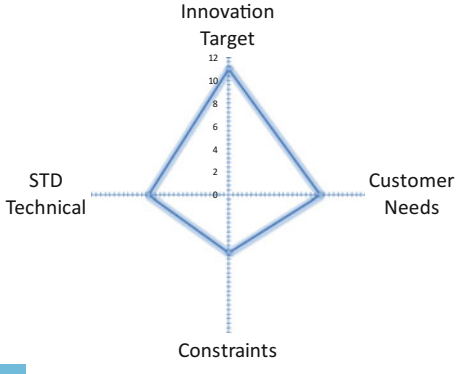


Fig. 4.5 Innovation plane of needs expressed for the new product

4.4 The Stakeholders

To easily cope with the identification of needs, a first action to be performed at the beginning of the whole process is defining who are the *stakeholders*, being usually defined as "either people or systems having some kind of interest with the system analyzed" (Walden, Roedler, Forsberg, Hamelin, & Shortell, 2015). If they are clearly identified, an immediate realization of the interface connecting the system to each one allows exploring the systems *functions* and somehow even its *neighborhoods*.

It is worth noticing that stakeholders could be at higher levels the customers themselves, thus driving the identification of needs. In some other cases they may be operators and systems usually connected to the main system to make possible its operation.

In software engineering, for instance, interests and links are easily detected, since usually the stakeholder sends either a request or a command to the system, which then replies by giving a feedback based respectively on a data transmission or an action. In a industrial context, a typical matter of misunderstanding in systems manufactured through a material processing is the role of some actors, who do not have a specific interest in the system, but are simply connected to it. This case is quite often found, as in the two test cases. They could be referred to as "*shadow stakeholders*". They appear, but their relevance is sometimes higher when a dysfunction occurs than in nominal behavior.

A simple example is the tank used to store the fuel into the aircraft. The fuel system doesn't ask the tank to store the fuel, but just sends it to the tank, during refueling for instance. Therefore, the tank seems free of interests in the system operation, and it could be either included inside or kept outside the neighborhoods of the fuel system. When it is interpreted as a specific subsystem, the connection between fuel system and tank doesn't require a real request to start the storage and it is never based on a pure transmission of data, but on a physical motion of fuel towards and from the tank. However, in dysfunctional analysis, a damage in the tank structure might cause a spillage of fuel and a severe risk of fire as well as a problem for the flight autonomy of the aircraft. In this case it is not so difficult generalizing the role of that stakeholder, since instead of having the tank asking to be filled by fuel in refueling, it is the fuel system that performs this operation as well as it autonomously performs the defueling. Nevertheless, in deploying the MBSE, this interaction often makes system engineers somehow confused.

The role of the electric power supplying is even more relevant in many systems. Very often professionals are prone to say that power supplying is never a stakeholder, but just a system being connected and naturally supplying energy without a specific request, thus neglecting to include it into the system model. It is very well known in steelmaking plant that power supplying not only is critical for the need of energy, but even because of the dynamic behavior of the network connected to the plant, which might affect its performance and safety, due to some anomalous behavior, like the phenomenon of flicker, whose symptom is a large fluctuation of voltage and current.

4.4.1 Didactic Test Case: Needs and Stakeholders

To apply the above proposed concepts a first test case is herein discussed. The owner of the steelmaking plant listen to the main customer. After a preliminary interview, a scenario about the needs could be defined as follows.

> A modular system has to be integrated with the plant, at the end of the rolling mill train, being able to introduce a suitable angular speed to convert the translational motion of the wire rod into a rotational motion. This motion has to be compatible with the speed of the production line. The system has to preserve the wire rod from any damage and to prepare it for a safe packaging in coils, to be stored and then delivered. Modularity of the coiler system consists of having a first "laying head", being dedicated to the spinning of the wire rod and a "coiler", being devoted to store the coils and to build up the package. The system has to be easily and safely operated. It is required that it can rotate up to a given angular speed, within a defined unbalance condition constraints, without structural failures. It is even required that the laying head can stop, in case of emergency, and that it behaves as an adaptive and actively controlled system, able to change its configuration, within some defined limits of operation, to face to some abrupt variation of the working condition, associated to the production of the rod. Moreover, this system has to be autonomous and standstill configuration should be possible, in pause. Additional constraints are maximum weight and volume, environmental compatibility, power consumption, maintainability, reliability, process monitoring… A suitable lubrication is required or even no one, no wear, no contamination…

As it clearly seems from previous investigation the customer needs are expressed in terms of qualitative targets, somehow insufficient to define a list of specification to design the new product. Therefore an immediate elicitation of requirements is useful to transform that information into a *corpus* of real technical specifications.

Before proceeding with this action, a preliminary identification of some main stakeholders is strictly required. This can be done by interacting once again together with customer. A first impression of the main idea committed to this example is provided in Fig. 4.6. The system looks like the assembly of two modules as the real coiler and the storage device. This shall be immediately matter of discussion, since system neighborhoods need to be defined. It is possible to consider that it is

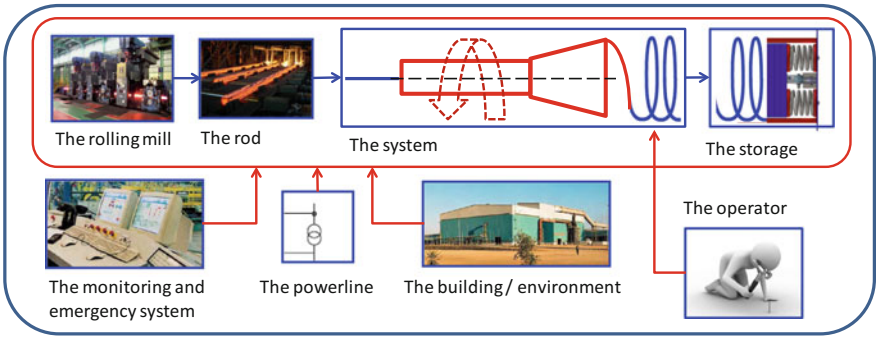


Fig. 4.6 Impression of some main stakeholders of the didactic test case

interacting with the rolling mill, as a part of it, with the rod, which is processed, but even with the main monitoring system, located at the control tower, with the power line, with the environment, corresponding to the building, with roofs, frames and floor, and with some operators, in different operating conditions, even in maintenance.

In this example, it was decided to limit the system under development to the real coiler, referred to as laying head, while the storage device could be considered as an additional stakeholder, quite close to the system. According to some remarks previously written among the stakeholders, the powerline and the building might look like as a shadow–type. Particularly, the floor where the system is constrained does not ask to be connected, but it has a relevant influence on the system behavior in terms of vibration, or even seismic power transmission, and of electrical grounding and risk of short–circuit, as it shall be further explained in next sections.

4.4.2 Industrial Test Case: Needs and Stakeholders

In case of the Ice Protection System, for a commercial aircraft, some needs can be clearly defined since the beginning of the product development.

> The environmental and weather conditions on ground and in flight might induce the ice accretion phenomenon and significantly affect the aircraft behavior, especially in terms of aerodynamic efficiency. Therefore, in case of an anti–ice system, the main need is that an effective action supported by a given amount of power could inhibit the ice accretion on all the most important surfaces of the aircraft and a monitoring system could eventually warn about any abnormal accretion; by converse, in case of a de–icing system, the ice accretion should be easily and quickly detected and localized, and, upon either an automatic or direct command of the crew, the system should remove promptly and effectively the ice, without any risk for the aircraft, the other systems and the people. Power used to perform either the anti or de–icing operation should be compatible with the overall power balance of the aircraft, i.e. with the other main functions of flying, maneuvering and on–board systems operating. Additional needs concern maximum weight, electromagnetic compatibility, maintainability, reliability, availability and safety as well as cost and sustainability (no contamination of the environment)…

In this case, some typical stakeholders are those of the technical domain of aeronautical engineering:

- *Pilot*: the system is operated by the pilot under a direct command or even automatically, but always keeping the option of a direct control on its activation and deactivation. Interest of pilot is controlling the system operation.
- *Environment*: the ice accretion is directly connected to the weather and environmental conditions. In this case a bright interest cannot be defined as is, but environment might cause and affect the ice accretion on the exposed surfaces of the aircraft, thus allowing considering that a sort of interaction is the deposition of ice on the aircraft structure.

- *On–board flight management and control system*: it needs to monitor the ice eventually deposed on the aircraft structure and to forecast its accretion.
- *On–board air data system*: it acquires and sends data useful to operate the anti/or de/icing action.
- *On–board power supplying system*: depending on the technology applied, some power is needed by the system to operate and this might provide the required amount (for instance electric power…).
- *On–board actuation supplying system*: the anti/or de/icing operation is surely based on a flow of air, heat, current… thus requiring a suitable connection to the on–board system able to provide it. In case of boots the system shall use a pneumatic actuation, based on some air coming from the turbofans, while, in case of electro–thermal heat generation, the electric system shall provide the required power.

4.5 The Role of Requirements in the Product Development

Converting the needs identified into some clear and executable requirements is a key issue of the concept design activity. It was demonstrated[1] that the largest part of failures occurring in technical systems are due to:

- Incompleteness of requirements.
- Lack of user involvement.
- Unrealistic expectations.
- Uncontrolled changing of requirements.

While evident benefits are associated to:

- Clear statement of requirements.
- User involvement.
- Realistic expectations.
- Change configuration and requirement management.

According to that analysis the Systems Engineering suitably drives the product designer to reach all the goals previously described:

1. *Elicitation of requirements* is performed iteratively by resorting to both functional and physical analyses, after a preliminary definition, through meta-models and numerical models, until that a suitable definition could be found.
2. Requirements are written by carefully considering the *customer needs* previously identified.

[1]Scientific American, September 1994 and Standish Group 1995, 1996.

3. *Feasibility of product* corresponding to needs and requirements defined is preliminarily checked, then deeply investigated through the verification and validation process.
4. Changes are tracked and a *configuration change management* is nowadays implemented to avoid any anomalous modification which might introduce a risk of failure.

As the Reader might immediately realize, requirements have a precise role within the lifecycle product development:

- *Project planning*: requirements are often used to define a path to be followed during the deployment of the design activity, since the need of allocating their contents to functions, elements and parts intrinsically motivates the designer to find the product configuration by checking that all requirements could be covered and allocated.
- *Risk prevention and management*: requirements usually describe and prevent some unsafe, failure or damage events, thus allowing to perform a suitable risk and safety analysis.
- *Acceptance*: they are written to allow the product to be positively evaluated in tests, homologations and even when it is released on the market.
- *Trade-off of layouts*: among different configurations proposed by the designer during the project deployment requirements allow selecting the most suitable one, depending on how it fits their prescriptions.
- *Change control*: changes are acceptable only when fitting requirements, thus assuring to be compatible with safety, quality and cost control.

4.5.1 Definition of Requirement

The specialized literature offer many definitions of requirement, as a starting issue of design. Among all, the one proposed by the IEEE in 1998 might provide a significant overview upon his main role:

> Requirement is a statement that identifies a product or process operational, functional or design characteristic or constraint which is unambiguous, testable or measurable and necessary for product or process acceptability.

As a statement requirement is a *textual expression*, even sometimes described into a tabular form or diagram, defining a *concept* being *traceable* and manageable. Product is something built in response to requirements and which has to be used to reach some goals, while process is a procedure for using things being built up for this purpose. Particularly, in material processing, process is a sequence of actions performed to shape, create, handle and make things to be then used.

It can be immediately noticed that the main difference between need and requirement, according to the INCOSE handbook (Walden et al., 2015), within the Systems Engineering is that:

- *Need* is some capability of system desired or required by a stakeholder.
- *Requirement* is a formal structured statement that can be verified and validated within the life cycle product development.

To proceed with the so–called *elicitation of requi*rements, it is worth noticing that some issues have to be considered. First at all, *contents* of requirements are relevant for a complete definition of system properties, however a preliminary *classification* of requirements helps in this activity and even in their allocation. A crucial issue is the *granularity* of requirements, i.e. how deep is the level of information concerned. It is important that this level is the same for all the requirements or, better, that different levels are defined to go deeper in detail, by organizing requirements as master, son, grandson, like in a waterfall of information. Very important is the *syntax* to be used to assure that requirements are clear, simple, never ambiguous.

It is often referred to "*smart*" the complete list of attributes of requirements, since SMART corresponds to an acronym stating for *Specific, Measurable, Achievable, Relevant and Traceable*.

Those attributes specifically mean:

- *Specific*: it must concern only one aspect of the system design or performance and must be expressed in terms of need, never of solution.
- *Measurable*: performance is expressed objectively and quantitatively and an exact pointing requirement can be tested thus verified.
- *Achievable*: it must be technically achievable at costs considered affordable.
- *Relevant or Realistic*: it must be appropriate for the level being specified.
- *Traceable* or *Time bound*: lower level requirements must flow from, and support, higher level requirements, but even realization should occur in the due time.

4.5.2 Classification of Requirements

As it was previously said for needs, even in case of requirements it is never so trivial listing their contents easily and completely. An important task, particularly in case of material product, is defining different classes to include each requirement into a specific *category* or *class*. This helps quite a lot the investigation, but very often it doesn't seem so easy.

A preliminary and very well-known classification is based on three main categories:

- *Functional requirements*: they answer to the basic question "what the system shall do in its operation?", thus identifying functions and flows of activities.
- *Operational requirements*: they answer to the basic question "what kind of usage is foreseen for this system?", thus leading to define the external actions applied and the main usages of system.
- *Constructional (or architectural or even structural) requirements*: they answer to question "what the system shall be made of?", thus focusing on the decomposition and the configuration of system.

Those three main categories are always considered in the requirement analysis, but very often they do not cover the needed number of visions, which may better describe the system. Therefore, a preliminary useful activity performed by the designer is listing a suitable number of classes, being related and strictly sufficient to give a complete impression of the system within the context of a typical technical domain it belongs. To go deeper into this investigation a better knowledge of the three main classes is required, as is depicted in Fig. 4.7.

In the specific case of product coming from a material processing some classes have to be considered, as they are listed in Table 4.3. As it could be easily noticed,

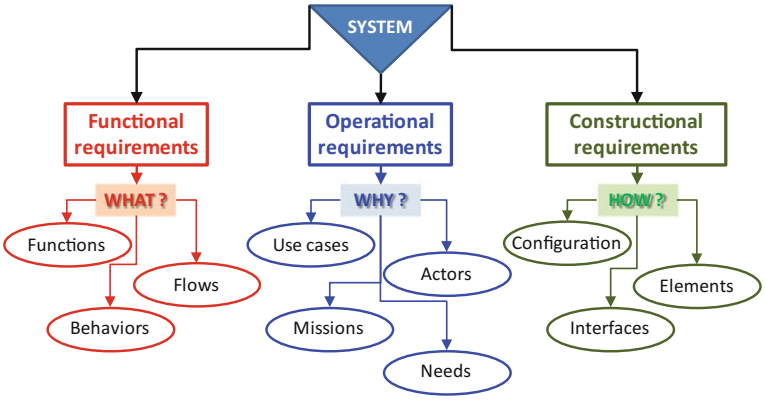


Fig. 4.7 Main classification of requirements and related contents

Table 4.3 Example of classification of requirements applied to a material product according to the SEBoK guide (Pyster et al., 2012)

Classes of requirements	
1. Functional req.	7. Adaptability req.
2. Performance req.	8. Physical constraints
3. Usability req.	9. Design constraints
4. Interface req.	10. Environmental conditions
5. Operational req.	11. Logistical req.
6. Modes and/or States req.	12. Policies and regulations
	13. Cost and schedule constraints

the classes and subclasses usually introduced are many. Some key issues are always considered, as the performances of system, its modes and states during the operation, the safety and, where applicable, the security. Typical of material product are the usability, which defines all the characteristics to make handling, usable and manageable the system, but even the interfaces which allow integrating the different parts of the system to reach the final assembly. Moreover, an important target is the packaging, together with the requirements for transportation, which might significantly change the contents of requirements.

The description of Table 4.3 points out that, after a preliminary rough distinction among functional, operational and constructional, every project needs to deploy a more complete list of classes, to help the product development. This activity is quite related to the technical domain involved. Each domain tends to resort to some traditional schemes of requirements, although in several cases the model based approach was never applied yet. Those classes can be further discretized as a tree, thus allowing including some relevant goals like *reliability*, *availability*, *maintainability and safety* (*RAMS*) and the product quality.

4.5.3 Syntax and Attributes of Requirements

As a verbal statement, the requirement should be written by applying a standard syntax and avoiding as much as possible what might depend on the designer who wrote it. Moreover, to be transformed into a manageable digital record, it is strictly important that contents could be simple and schematic. In Systems Engineering those goals are achieved by resorting to a fairly standard description of requirements like the one depicted in Fig. 4.8. Several elements are included. After the

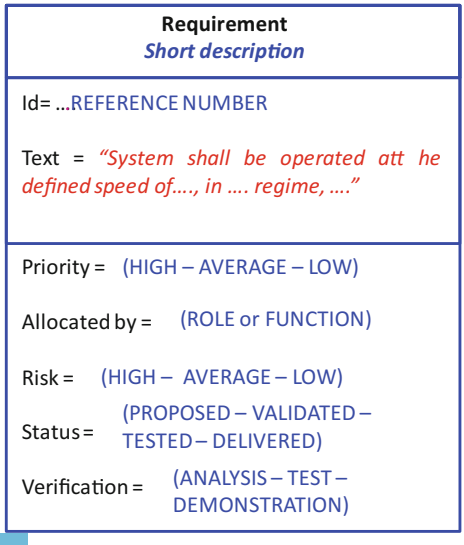


Fig. 4.8 Example of main attributes of a requirement

text, which describes the main statement and an *identification number* or *code* which allows allocating and linking the requirements to the other artifacts of the MBSE, the *priority* of requirement is defined, together with the *risk* associated in case of lack of fulfillment. *Allocation* is even declared as well as the *status* of requirement within the product development process and, finally, the level of *verification*. Authors are often enclosed into a tabular representation of the requirements.

A *syntax* is also applied to make clear the text of requirement. Among a number of rules, just to introduce this topic, few of those might be considered.

A requirement must state:

- *Who* is responsible.
- *What* shall be done.
- Or *how* something shall be done.
- Or even *under which constraints* something shall be done.

The usual format of requirement is "*who shall what*" and active forms are preferred to passive ones. In previous test case, the following statement can be found: "The system shall adapt the switch–off time to the rate of ice accretion". Therefore the literature suggest of using:

- For requirements "…*shall*…".
- For facts and declarations "…*will*…".
- For non-mandatory provisions "…*should*…".

By converse the use of "…must…" is definitely deprecated. Those rules allow the Reader catching the real purpose of sentences when they are read.

4.6 Tools for Writing Requirements

The requirement analysis basically resorts to three tools already developed and available on the market: a *manager* software, some *quality* and *authoring* suites and some software for *functional analysis*, where requirements are imported and allocated to functions and system elements. This one will be deeply described in following chapters, while the two first above cited are briefly herein introduced.

4.6.1 Requirements Manager

Listing requirements seems quite simple in the provided examples, because of the fairly small number of items. In complex systems this activity can be fairly heavy and rather difficult. The effort associated with the organization of the requirements specification, together with the classification of the statements themselves and their

management during the design process requires a systematic approach for the definition, collection and update of requirements, which is usually faced with a model–based approach, especially for the design of complex systems. Therefore some helpful tools are currently used to enhance the effectiveness of design in describing and collecting the requirements. This generally leads to the adoption of a centralized requirements database, capable of keeping trace of changes and updates for different classes and at several levels. This hub is conceived to allow accessing from the design tools, that shall be used to characterize the system architecture and its behavior, starting from high level requirements and gathering at the same time new requirements, generally at lower levels, which are the results of system modeling activities.

As a consequence, the main features of such a database shall be:

- *Robustness*, when identifying uniquely the requirements and storing their changes during the project, by reducing some mistakes eventually arising from the user input.
- *Flexibility* in terms of customization of classes and levels of requirements, depending on the involved engineers and the stakeholders needs.
- *Simplicity*, to allow a high portability of the environment where requirements are defined, which shall be user friendly.
- *Reliability*, to guarantee a low probability of loss of data or generation of wrong traceability paths.
- *Interoperability*, with the widest range of tools, in order to be able to communicate with the largest possible tool chain during the whole system lifecycle.
- *(Two–ways) connectivity*, assuring a smooth access to the data in both directions (input/output).
- *Security*, usually implemented with a strong authentication strategy of the users and a specific identification of roles to segregate data, to assure that each profile may have access only to the information needed, while protecting the confidential data being out of the scope of that role within the project.

An example of implementation regarding the creation of a requirements database in different contexts, through the IBM Rational DOORS® tool, will be described in this handbook and applied to the test cases. Some of the aforementioned characteristics will be highlighted, showing the practical implication they have within the tool. The definition, collection and updating of requirements is performed in some steps, starting from a high level requirements specification, being the first result of the elicitation process driven by the stakeholders needs.

It might be remarked that the structure of the requirements specification and some basic features of the tool will be herein shown, while the updating activity and the traceability process will be described later, i.e. within the development of the system design, through the Operational, Functional, Logical and Physical analyses, as are described in Chaps. 5–8.

The IBM DOORS® and its updated platform IBM DOORS Next Generation or DNG® allow structuring the information related to the requirements elicitation and

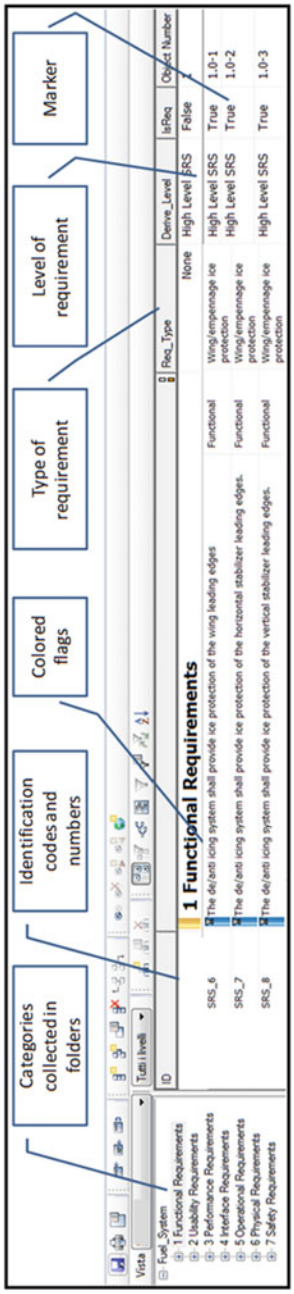


Fig. 4.9 Example of requirements written inside a requirements manager as the IBM DOORS®

Property	Description
Selectivity	Capability of assessing the system properties depending on the working conditions
Self–diagnosis	Existence of intrinsic parameters which detect a failure condition
Self–tuning	Skill of performing an internal calibration
Sensitivity	Relation between cause and effect in the coupling (i.e. linear, non linear)
Shape–ability	Capability of modifying the system shape for different needs
Self–recovery	Possibility of reaching a saturation without failures
Simplicity	Simplicity of the energy conversion mechanisms, of the configuration
Self–repairing	Skill of recovering a stable and working condition after a saturation
Stability	All of the possible stabilities of the system operation
Standby skills	Possibility of keeping a defined configuration
Survivability	Capability of avoiding failure modes
Switch–ability	Possibility of operating at different levels of energy / power, if the architecture of the system allows
Smart-nect-ness	Capability of being interconnected with other systems and of fruitfully exchanging data and information

Fig. 4.10 Typical requirements related to the smartness of systems as they are identified in the literature

even converting into a word processor like Word®. It is worth noticing that the classes of requirements are evidenced in folders. Some colored flags allow the user detecting the status of each requirement (Fig. 4.9). As it can be seen, the main interface contains the tree of contents on the left side, while the so-called *objects* appear on the right. Objects represent the row of the workspace, which basically are the statements. Each object may be characterized by several *attributes* representing the column of the requirements area. They include the identification codes/numbers, being like a name for each requirement to be used in allocation, a flag suggesting the status of requirements, the statement itself, a label declaring the class, the level and some optional items as markers and check numbers.

If one looks at the digital content of that file, the so–called "System Requirements Specification" or SRS is a crucial product of the model based approach. It might be realized that both the statements and how they are written within the frame of the requirements manager tool are issues for the information related to the set of requirements.

The SRS is relevant because:

- It collects the system requirements.
- The list of requirements is uniquely ordered and numbered item by item.
- It is digitalized and can be easily shared with other users.
- It can be imported in another working environment.
- It applies a standard form to be easily understood by users.

If the requirements management is performed quite straightly through the tool, a critical engineering activity is the definition of the specification architecture, i.e. of the *classification of requirements*. It depends on the context and on the engineering domain in which the design process is developed. Classes are very important not only to define the requirements but often even to organize people and resources within the design group, especially when it doesn't correspond to a Company but more to a consortium of manufacturers, as it happens in the aerospace domain. This issue is herein practically deployed for the two test cases to show the Reader how an assessment of classes and categories is performed.

4.6.2 Requirements Quality and Authoring Suites

Another set of tools is nowadays proposed for the requirement analysis. It includes some software for managing and checking the quality of written requirements, for authoring and writing or to perform a semantic analysis. An example is proposed by the REUSE Company. The main assumption is that writing some complete, correct and consistent requirements is rather difficult and long. To reduce the time spent in developing the project, those tools help in facing three main issues of requirements:

- *Correctness*: they check that requirements are easily readable (short, simple), unambiguous and cited properties are measurable.
- *Consistency*: the text is checked to verify the consistency between statements, models and requirements.
- *Completeness*: it is checked that all needs are covered by the requirements and that they are suitably allocated on functions and components.

Those actions are provided by an *analyzer*, which performs the quality control, according to the policies defined by the user. It analyzes the text, its structure, the semantic information, the units and all eventual redundancies. An authoring service is then activated. It supports the writer in real–time, by checking the vocabulary and the grammar consistency, but even the completeness of statements. Finally, a semantic analysis is performed to apply the specific domain vocabulary, by connecting the ontologies and suggesting the patterns to be used as a template for writing.

4.7 Requirements Refinement and Assessment

The daily practice demonstrates that in complex systems all the above described activities are never sufficient to give a complete impression of the system to be manufactured. They are supported by several effective tools, but resorting to some iterations from the concept stage to validation is always required. Therefore, the list of requirements composed in a very early concept design is usually helpful to start a functional analysis, but never coincides with the final, assessed during the product development. A challenging issue of the Systems Engineering is a direct, digital and automatic link between requirements, functional and physical analyses. It makes possible to run a fast process of refinement, which allows assessing the requirements, the system configuration and a preliminary validation, as it shall be described in next sections.

4.7.1 Didactic Test Case: Classification and List of Requirements

Classification. The mechatronic system consisting of a suspended and controlled rotor connected to a laying head actually is a quite particular industrial equipment, even within the steelmaking activity. Few references are found to identify some typical examples of requirements analysis and classification, respectively. Moreover, the existing configurations of that machine basically resort to a classic mechanical construction instead of active supports. Therefore, after a preliminary interview with the manufacturer, which allows identifying some needs, the system requirements are tentatively listed by assuming the sources above described as a reference for writing the SRS. Then, a classification was defined, as detailed in Table 4.4.

In this case, classes are quite close to those described in Table 4.3. It is worth noticing that system needs have to be integrated within the rolling mill plant, thus requiring a deep attention in the design activity to some logistical issues as the interfaces, the information management, the installation and maintenance as well as the transportation, which are already distinguished at high level, i.e. in that list. Physical characteristics actually include constraints and environmental

Table 4.4 Example of classification of requirements applied to the didactic test case

Classes of requirements (Spoiler)	
1. Operational Requirements	7. Physical characteristics
2. Functional Requirements	8. System Safety
3. Performance Requirements	9. Information management
4. Constructional Requirements	10. Installation and maintenance
5. System modes and states	11. Transportation
6. System interfaces	12. Cost

compatibility issues. Design constraints are somehow expressed by the Constructional requirements. Safety and cost are directly indicated, although in every project they appear as main issues of design.

Preliminary elicitation of requirements by source. The elicitation of requirements was performed by resorting first to their source. This approach allows collecting a preliminary list, which is just slightly larger than the one collecting customer needs, but already converted into technical and measurable contents, as it looks in next tables (Figs. 4.11, 4.12, 4.13 and 4.14). Particularly, a first list concerns the needs expressed by the customer during a preliminary interview and by some additional communications. It looks quite complete, although contents are not so homogeneous. Some requirements related to the technical domain are then listed. They are defined directly with the manufacturer as macroscopic exigencies of fulfillment. Finally, a set of requirements related to the *smartness* of system are added. The last could be defined by considering some typical attributes of the mechatronic systems, as they are described in the literature and summarized in Fig. 4.10.

In the above figures, several requirements are described. It could be immediately realized that customer was prone to identify either a quantitative requirement, based on a numerical reference, or a qualitative requirement, focused on a certain attribute or action. SR1, SR2, SR6, SR10, SR11 and SR13 for instance (in Fig. 4.11) all include a numerical target, while SR3, SR4, SR5 and others provide just a statement. Different levels of information are foreseen, but they are not subdivided into layers in terms of *priority* or *granularity*. Some are very general as the SR20 (against the risk of fire), some other ones are quite specific as the SR28 (linear conversion of energy). A more detailed description is provided by requirements belonging the list of technical issues. In this case the practice of this domain suggests some specific issue for the design activity as, for instance, the need of rotating in supercritical regime the rotor to benefit of the natural self–alignment (SR102). Smartness requirements are defined by resorting to the list of Fig. 4.14 and applied to the technical context. As it could be easily realized, this template suggested a number of pertinent and useful items. Identification numbers of requirements in the example reveal that designers included some other statements, which were then deleted, after a preliminary screening (identification numbers are not in progressive order). Nevertheless, to avoid any mistake, the listed requirements were not numbered again, but they kept their initial identification code.

Listing of requirements by class. In an advanced step of elicitation, it is preferable resorting to *classes* to organize the statements collected from different (here just three) sources. In the example, the applied categories correspond to those introduced in Table 4.4. However, a lack of a more general category just aimed at defining the system purpose is found. Therefore, the requirements manager tool introduced a first generic class for some very high-level statement, to allow a better distribution of contents. The structure of the document looks like in Fig. 4.15.

As Fig. 4.15 shows, to assess the requirements list is useful applying some typical rules. Each requirement has to be identified by a suitable *code or identification number*. This code is assigned as soon as the record is open and never

ID	CUSTOMER NEEDS
SR1	The system shall have a maximum weight of WWW kg.
SR2	The system shall not exceed the maximum volume of DDD (or DD x YY x ZZ) m^3
SR3	The system shall assure a complete electromagnetic compatibility (ECM)
SR4	The system shall be easily transportable
SR5	The system shall be compatible with the environment (no production of pollution or contamination)
SR6	The system shall be able to shape a steel wire rod of XX diameter into coils of WW diameter.
SR10	The system shall be able to stop the rod within a distance of XXX meter from the rolling mill way out.
SR11	This system shall not exceed a power consumption of XYZ Watt.
SR12	The system shall be able to keep a defined configuration in suspension both at constant spin speed in rotation and at standstill.
SR13	The system shall assure that temperature never exceed XXX °C to prevent any risk of fire associated.
SR14	The system shall be able to apply a suitable angular speed to convert the translational motion of the wire rod into a rotational motion, compatible with the speed imposed by the production line.
SR15	The system shall be able to shape the rod to allow a faster removal and disposal.
SR16	The system shall preserve the rod from any damage and prepare it for a safe packaging in coils, to be stored and then delivered.
SR18	The system shall reduce the wear of material and of bearings in coiling operation.
SR19	To increase its safety the system shall suitably react to any abrupt variation of the working condition, associated to the production of the rod.

Fig. 4.11 Requirements of the didactic test case after the elicitation based on the customer needs (Part 1)

ID	CUSTOMER NEEDS
SR20	The system shall prevent any risk of fire.
SR21	The system shall be able to rotate at regular regime of angular speed, within a defined unbalance condition constraints, without structural failures, and stably.
SR22	The system shall be equipped with a control system able to prevent any risk of catastrophic failure.
SR23	The system shall be able to operate under the inputs of the main supervision control system of the steelmaking plant.
SR24	The system shall provide a measurement of the amount of material coiled and packaged.
SR25	The system shall be able to safely stop in case of emergency.
SR26	The system shall be a dedicated unit (module) which has to be easily integrated with the steelmaking plant, at the end of the rolling mill.
SR28	The energy conversion mechanism and configuration in coiling operation shall be simply and linear.
SR29	The system shall be equipped with sensors to measure physical quantities like currents, temperatures and rotor spin speed.
SR35	The system shall provide a continuous monitoring of the axial and radial displacements of the rotor shaft.
SR36	The system shall be able to detect the presence of the wire rod inside.
SR37	The system shall provide all data related to the rotor shaft position, vibration, currents, temperature to the overall supervision control system for warning and security management.

Fig. 4.12 Requirements of the didactic test case after the elicitation based on the customer needs (Part 2)

ID	TECHNICAL REQUIREMENTS AND DOMAIN PRACTICE
SR102	The system shall operate easily and safely in supercritical regime above the rotor critical speed.
SR103	The system shall operate easily and safely in supercritical regime below the instability threshold to achieve a good self-centering.
SR104	The system shall be able to assure a rotor balancing condition corresponding to balancing class ABC.
SR105	To reduce the wear between materials the system shall be suspended without direct contact.
SR106	The system shall either use special lubricants to prevent any risk of fire or operate without any lubricant.
SR107	The system shall prevent any abrupt dissipation which might induce a severe increasing of temperature.
SR108	The rotor system balancing, stability and dynamics will be controlled in operation.

Fig. 4.13 Requirements of the didactic test case after the elicitation based on some technical standards

ID	SYSTEM SMARTNESS
SR201	The system shall be able to modify its shape for different needs (in this case variable shape of the head nozzle).
SR203	The system shall provide balancing after a preliminary centering performed through an internal calibration at standstill.
SR204	The system shall be able to tune its properties depending on the working conditions: rotor spin speed (related to that of incoming rod) and power (set up to a defined maximum).
SR208	The system shall be able to reach a saturation without failures (in this case if currents are too large shutdown must be automatically done on bushings).
SR209	The system shall be able to recover a stable and working condition after a saturation. Suspension shall be possible after a shoutdown.
SR210	The system shall be able to avoid the condition for failure occurence: uncontrolled vibration, instability, severe unbalance, heating, accidental stop of rod delivery will be prevented.
SR211	The system shall be able to detect a lack of power supplying from the main powerline and to activate the backup system.
SR212	Configuration of the active suspension shall assure a linear response of the system to an imposed displacement through a suitable coupling phenomenon (electromechanical).
SR214	Actions of suspensions shall be linearized to allow a simpler control.

Fig. 4.14 Requirements of the didactic test case after the elicitation based on the system smartness

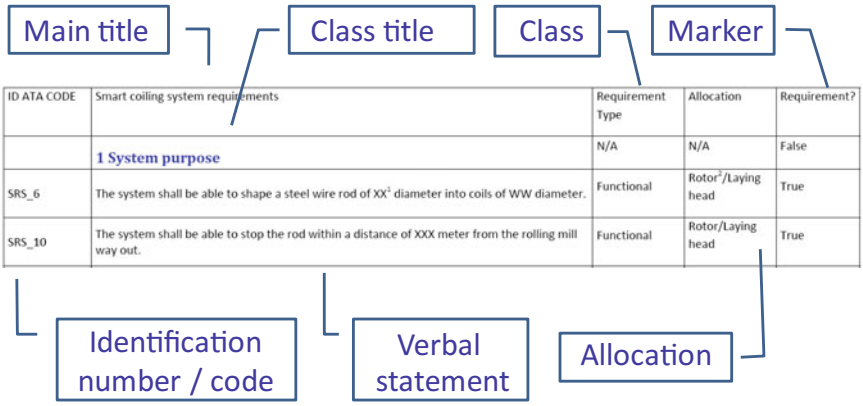


Fig. 4.15 Structure of the list of requirements (example)

changed after, thus avoiding that any reordering of the list might cause a mismatch in allocating and linking the requirements to functions. It is not considered a matter if the number sequence is never complete nor perfectly ordered, since, independently from the value, the correspondence between code and statement must be unique, whenever the list is accessed by users. This allow linking to this statement, during the allocation, all the following entities developed. The *category* or *class* of requirement is explicitly declared. All the requirements belonging a certain class are usually listed together, under the same title. To make the automatic detection of relevant digital contents easier a marker distinguishes comments and titles from requirements.

As that organization was applied to the test case, the requirements were listed like in following tables (Figs. 4.16, 4.17, 4.18, 4.19 and 4.20). They are distributed within the classes previously defined. To point out two main challenging issues of the application of the MBSE, in this description two additional *objects* are shown. First at all, since it was pointed out that the requirements assessment needs an iterative approach, in those figures a list of requirements previously divided by source is replicated by resorting to classes, but some additional ones, in different color, are inserted. Actually, it should be realized that there is never a perfect elicitation of requirements, but just either an incomplete or a complete list. Only after performing the operational and functional analyses, and allocating the requirements, a suitable compromise between a lack of information and a too deep description of system specification is usually found. In the example, some new requirements are added just to show the Reader how those analyses could improve the original list assessed at the beginning of the project. The SRS_27, for instance highlighted the need of autonomy of the system, being initially poorly stated. The SRS_302, SRS_303 and SRS_304 define more in details the mechanical and electrical interfaces, after that the analyses pointed out a problem of grounding the system and of stably fixing against the risk of short-circuit and of seismic vibration. Those results will be identified by resorting to the functional analysis described in

ID ATA CODE	Smart coiling system requirements	Requirement Type	Allocation	Requirement?
	1 System purpose	N/A	N/A	False
SRS_6	The system shall be able to shape a steel wire rod of XX[1] diameter into coils of WW diameter.	Functional	Rotor[2]/Laying head	True
SRS_10	The system shall be able to stop the rod within a distance of XXX meter from the rolling mill way out.	Functional	Rotor/Laying head	True
SRS_15	The system shall be able to shape the rod to allow a faster removal and disposal.	Functional	Rotor/Laying head	True
SRS_5	The system shall be compatible with the environment (no production of pollution or contamination)	Constructional	General	True
	2 Functional requirements	N/A	N/A	False
SRS_14	The system shall be able to apply a suitable angular speed to convert the translational motion of the wire rod into a rotational motion, compatible with the speed imposed by the production line.	Functional	Rotor/Laying head	True
SRS_18	The system shall reduce the wear of material and of bearings in coiling operation.	Functional	General	True
SRS_25	The system shall be able to safely stop in case of emergency.	Functional	Rotor/Laying head	True
SRS_203	The system shall provide balancing after a preliminary centering performed through an internal calibration at standstill.	Functional	Rotor/Suspension	True
SRS_211	The system shall be able to detect a lack of power supplying from the main power line and to activate the backup system.	Functional	General	True

[1] Values, figures and other information will not be explicitly written because of a non disclosure restriction, but substituted by letters.
[2] Including its control system

Fig. 4.16 Requirements of the didactic test case ordered by class (Part 1)

ID ATA CODE	Smart coiling system requirements	Requirement Type	Allocation	Requirement?
SRS_36	The system shall be able to detect the presence of the wire rod inside.	Functional	General	True
	3 System modes and states	N/A	N/A	False
SRS_12	The system shall be able to keep a defined configuration in suspension both at constant spin speed in rotation and at standstill.	Functional	Rotor/Laying head/Suspension/Stator	True
SRS_13	The system shall assure that temperature never exceed XXX °C to prevent any risk of fire associated.	Operational	General	True
SRS_27	*The system shall be autonomous, i.e. its operation will not depend upon the energy supplied to the steelmaking plant, at least for a given time of XX hours.*	Operational	General	True
	4 Operational requirements	N/A	N/A	False
SRS_21	The system shall be able to rotate at regular regime of angular speed, within a defined unbalance condition constraints, without structural failures, and stably.	Operational	Rotor/Laying head	True
SRS_102	The system shall operate easily and safely in supercritical regime above the rotor critical speed.	Operational	Rotor/Laying head/Suspension/Stator	True
SRS_103	The system shall operate easily and safely in supercritical regime below the instability threshold to achieve a good self-centering.	Operational	Rotor/Laying head/Suspension/Stator	True
SRS_104	The system shall be able to assure a rotor balancing condition corresponding to balancing class ABC.	Operational	Rotor/Laying head/Suspension/Stator	True
SRS_108	The rotor system balancing, stability and dynamics will be controlled in operation.	Operational	General	True

Fig. 4.17 Requirements of the didactic test case ordered by class (Part 2)

ID ATA CODE	Smart coiling system requirements	Requirement Type	Allocation	Requirement?
SRS_209	The system shall be able to recover a stable and working condition after a saturation. Suspension shall be possible after a shoutdown.	Operational	Rotor/Suspension/Stator	True
	5 Performance requirements	N/A	N/A	False
SRS_11	This system shall not exceed a power consumption of XYZ Watt.	Operational	General	True
SRS_204	The system shall be able to tune its properties depending on the working conditions: rotor spin speed (related to that of incoming rod) and power (set up to a defined maximum).	Operational	General	True
SRS_214	Actions of suspensions shall be linearized to allow a simpler control.	Operational	Suspension	True
	6 Constructional requirements	N/A	N/A	False
SRS_1	The system shall have a maximum weight of WWW kg.	Constructional	General	True
SRS_2	The system shall not exceed the maximum volume of DDD (or DD x YY x ZZ) m^3	Constructional	General	True
SRS_22	The system shall be equipped with a control system able to prevent any risk of catastrophic failure.	Constructional	General	True
SRS_26	The system shall be a dedicated unit (module) which has to be easily integrated with the steelmaking plant, at the end of the rolling mill.	Constructional	General	True
SRS_29	The system shall be equipped with sensors to measure physical quantities like currents, temperatures and rotor spin speed.	Constructional	General	True
SRS_105	To reduce the wear between materials the system shall be suspended without direct contact.	Constructional	Suspension	True
SRS_106	The system shall either use special lubricants to prevent any risk of fire or operate without any lubricant.	Constructional	Rotor/Laying head/Suspension/Stator	True
SRS_201	The system shall be able to modify its shape for different needs (in this case variable shape of the head nozzle).	Constructional	Laying head	True

Fig. 4.18 Requirements of the didactic test case ordered by class (Part 3)

ID ATA CODE	Smart coiling system requirements	Requirement Type	Allocation	Requirement?
	7 System interfaces	N/A	N/A	False
SRS_23	The system shall be able to operate under the inputs of the main supervision control system of the steelmaking plant.	Constructional	General	True
SRS_302	*The system shall be equipped with a platform to be easily picked up and moved*	Constructional	Stator	True
SRS_303	*The system shall be connected to the power line by suitable connectors*	Constructional	General	True
SRS_304	*The system shall be constrained to the plant platform and fixed*	Constructional	Stator	True
	8 Physical characteristics	N/A	N/A	False
SRS_3	The system shall assure a complete electromagnetic compatibility (ECM)	Functional	General	True
SRS_28	The energy conversion mechanism and configuration in coiling operation shall be simply and linear.	Constructional	General	True
	9 System safety	N/A	N/A	False
SRS_16	The system shall preserve the rod from any damage and prepare it for a safe packaging in coils, to be stored and then delivered.	Functional	Rotor/Laying head/Suspension/Stator	True
SRS_19	To increase its safety the system shall suitably react to any abrupt variation of the working condition, associated to the production of the rod.	Functional	General	True
SRS_20	The system shall prevent any risk of fire.	Functional	General	True
SRS_107	The system shall prevent any abrupt dissipation which might induce a severe increasing of temperature.	Functional	General	True
SRS_208	The system shall be able to reach a saturation without failures (in this case if currents are too large shutdown must be automatically done on bushings).	Functional	General	True
SRS_210	The system shall be able to avoid the condition for failure occurrence: uncontrolled vibration, instability, severe unbalance, heating, accidental stop of rod delivery will be prevented.	Functional	General	True

Fig. 4.19 Requirements of the didactic test case ordered by class (Part 4)

ID ATA CODE	Smart coiling system requirements	Requirement Type	Allocation	Requirement?
SRS_308	*The system shall avoid any dispersion of electric charge upon the main plant*	Functional	General	True
SRS_310	*The system shall be suitably shut up into a inaccessible space to avoid any injury to the operators*	Constructional	General	True
	10 Information management	N/A	N/A	False
SRS_24	The system shall provide a measurement of the amount of material coiled and packaged.	Functional	General	True
SRS_35	The system shall provide a continuous monitoring of the axial and radial displacements of the rotor shaft.	Functional	Rotor/Suspension	True
SRS_37	The system shall provide all data related to the rotor shaft position, vibration, currents, temperature to the overall supervision control system for warning and security management.	Functional	General	True
SRS_309	*The system shall be connected to a main network for a continuous data exchange (IoT)*	Constructional	General	True
	11 Installation and maintenance	N/A	N/A	False
SRS_306	*The system shall be easily installed and connected to the main system*	Constructional	General	True
SRS_307	*The system shall be equipped with removable housing to allow a fast maintenance*	Constructional	General	True
	12 Transportation	N/A	N/A	False
SRS_4	The system shall be easily transportable	Constructional	General	True
SRS_305	*The system shall be equipped with a locking device for the rotor to allow a safe transportation*	Constructional	General	True
	13 Cost	N/A	N/A	False
SRS_301	The overall cost of the system shall be compatible with the cost of the main rolling mill plant where is installed.	Operational	General	True

Fig. 4.20 Requirements of the didactic test case ordered by class (Part 5)

next chapters, but it can be realized since now that a deeper detailed specification can be only reached through the process proposed by the Systems Engineering. Moreover, the physical analysis which shall be developed in next chapters allows defining components and parts of the system architecture, which are crucial to perform the allocation of requirements as is herein preliminary described. From this point of view the Reader can compare the first set of Figs. 4.11, 4.12, 4.13 and 4.14 and the second one, including Figs. 4.16, 4.17, 4.18, 4.19 and 4.20 to perceive, at least preliminarily, the difference occurring between the initial and the advanced elicitation of requirements.

If one analyzes in detail the last set, it could be remarked that:

- Designer was prone to use two levels of classification, i.e. a high level corresponding to the three well known categories of Functional, Operational and Constructional requirements, to have an immediate impression of the main distribution of the related information, but even a second level of classes distinguishing statements for safety, performance, cost etc. That approach might be questionable, but is compatible with the practice of the MBSE.
- As soon as this list was tentatively filled, some classes were poorly described, thus inducing a deeper elicitation of requirements to add those written in italic. This was a direct consequence of the classification of requirements.
- Sentences and statements were written as the user directly suggested, but somewhere the syntax and the vocabulary might be a little bit imprecise or at least fairly rough, as it shall be herein further discussed.
- A preliminary allocation was attempted, although it was based on a very schematic architecture for the system, by resorting, for instance, to the concept of rotor, based on a real rotating part (rotor shaft), a fixed part (stator), a suspension system to connect those two elements (suspension) and the laying head as a main component, according to the original idea of the coiling system.

The figures shown in this section were exported from the IBM DOORS® database. As it will be clearer within the description of the requirements analysis for the industrial case study, they are organized in several *formal modules* which help organizing the structure of the SRS and the classification of the requirements themselves. An important issue concerns the internal traceability of requirements. When the SRS is discretized in modules, a great accuracy must be assured during the design activity in order to define how the requirements contained within the modules are related to each other. The implementation of dedicated matrices or *link modules* is the best way of registering this kind of information directly inside the requirements tool. Since this step a correlation between the customer needs and the requirements is looked for. The link modules used to connect the Customer Needs to the actual requirements specification are usually described by a sort of matrix as depicted in Fig. 4.21.

In different tools, the above Link Modules are defined by several graphical forms, but they resort always to a matrix to perform this allocation. Therefore, it looks fairly easy performing an impact analysis of the involved requirements in

CUSTOMER NEEDS VS REQUIREMENTS	Modularity	Motion conversion	Synchornization with mill	Damage prevention	Wire rod packaging	Safety	Self-alignment	Balancing	Emergengy stop	Adaptive configuration	Active control	Autonomy	Stand still	Configuration (weight, volume…)	Environmental compatibility	.....
SRS_1														X		
SRS_2														X		
SRS_3															X	
SRS_4	X															
SRS_5															X	
SRS_6		X														
SRS_7																
SRS_8																
SRS_9																
SRS_10			X						X							
SRS_11												X				
SRS_12							X	X					X			
SRS_13				X		X										
SRS_14		X	X													
SRS_15					X											
....																

Fig. 4.21 Link matrix used to connect the Customer Needs to the system requirements

case of both derivation and change processes. These internal traceability links are usually poorly exposed outside the specification, but are automatically modified by the tool in case of updating, i.e. when a requirement changes the related link still holds, at least until that the requirement is deleted. This approach assures the consistency inside the specification, but there is an alternate option, which shall be described in the industrial test case, where the SRS is organized as a federation of dedicated SRS, being related to each subsystem of an aircraft.

Actually, this preliminary work does not assure that the requirements identified are complete, consistent and correct. A further investigation is needed and is performed by deploying the functional analysis of the system.

4.7.2 Industrial Test Case: Classification and List of Requirements

The same activity can be applied to the industrial test case. Some attempts were required to assess the list of *classes* of requirements described in Table 4.5, according to the practice of aeronautical engineering.

Table 4.5 Example of classification of requirements applied to the industrial test case

Classes of requirements (Aircraft)	
1. Operational Requirements	6. Safety Requirements
2. Functional Requirements	7. Cost Requirements
3. Performance Requirements	8. Maintainability and testability Requirements
4. Physical Requirements	
5. Installation Requirements	

As in previous example, the selected classes are at least the high-level categories through which the SRS is written, but they already give an impression about the structure defined for the product development. Once again, the operational requirements are aimed at defining mission, goals, scenarios and context as well as the environment where the system shall operate, while the performance requirements describe both the service of the on–board Ice Protection System (IPS), and include some measurable parameters. Safety and cost play a significant role in this domain, because of the intrinsic nature of safety critical device. Moreover, as an on–board equipment, safety and cost greatly affect the integration and the selection of the commercial components to be applied to build up a redundant system. This motivation applies even to the specific classes concerning the installation first, then the maintainability and the testability. Physical requirements define the architecture, the construction and the packaging issues.

After a preliminary screening, a list of requirements for the industrial test case could be assessed as proposed in next figures, by resorting to a structure close to the one described in Fig. 4.15. The requirements elicitation started from the main system, i.e. the aircraft, then it was developed for all the subsystems.

Some typical issues of the requirements elicitation are clearly pointed out. Since the derivation of requirements starts from the main system, as the aircraft, the Technical Standards play a relevant role, when they define a reference scenario of weather conditions for the system operation, associated to the flight activity. An evidence of this influence is provided by the SRS_3, where several details about the flight mission are defined. It includes not only qualitative statements but even a quantitative information.

A preliminary assumption about the architecture of the aircraft to develop the IPS is even required, to identify the main functions of the subsystems. It could be realized by reading the SRS_6 and SRS_7 which define where the protection shall be applied, for this specific aircraft. It might be noticed that this need is related to the appearance in the baseline of the product lifecycle development, depicted in Chap. 3, of several "V–diagrams", being applied to the main system and to subsystems. At this step of the concept design, the main system should be sufficiently defined to allow a straight deployment of the design for every subsystem. It is worth noticing that some *Non-functional* requirements appear in the list. Actually, they specify physical properties of the system, although they are indirectly defined by the sizes of the parts of the aircraft where they are applied. This kind of requirement must be never confused with other ones which are related to a *dysfunctional*

behavior of the main system, as, for instance the SRS_4, which considers the event of a failure of the aircraft engine. The safety and maintainability requirements include several figures of merit, to be carefully defined and derived from the technical domain, the industrial practice of the manufacturer and possibly form the Standards.

The requirements specification for the industrial test case is organized by resorting to a dedicated project folder of the IBM DOORS®, containing the aforementioned *Formal Module*. This is a sort of template, being organized like a written document and implemented in a table–like format, where rows are the *objects* (i.e. the requirements) and the columns are the *attributes* (i.e. the additional information associated with each requirement) (Figs. 4.22, 4.23, 4.24 and 4.25).

Objects are classified in two categories:

- *Object Heading*, which are used to organize the structure of the specification by introducing chapters, sections and paragraphs; they are never requirements but they are just used to split the requirements in families and groups.
- *Object Text*, which are requirements, containing formal specifications, expressed through a verbal language.

Attributes include different data, like the identifier, some information about the author, the history (creation, updates etc.…) and every additional data defined by the user, making this interface extensively customized. The definition of such attributes is never random. A straight relation between the requirements list and a suitable *view* of the system is always looked for. This view consists of the *Formal Module* look, i.e. it is based on the displayed *attributes*. It usually contains:

- *ID Code*: the identifier of requirement is the unique numerical code which allows recognizing each requirement during the different steps of design.
- *System Requirement*: the text containing the specification expressed by each requirement. A colored bar allows recognizing at a glance the level of its development.
- *Requirement Type*: this attribute indicates the family to which the requirement belongs. It confirms the information provided by the *Object Heading*, with more detail.
- *Allocation*: this column allows expressing a preliminary hypothesis about the component or part where the requirement shall be allocated during the system design. This attribute looks a useful tool to drive the development of each requirement, especially in engineering. Since the final system will have parts categorized under a standard or a regulation, this attribute allows relating the Standard issue to the requirement. In aeronautics, the on-board systems and their components are uniquely identified through the ASD S1000D (former the ATA 100) numbering reference, being used to provide a common platform for the development of different types of aircraft.
- *Marker*: this attribute is a pure service input, aimed at identifying whether the current item corresponds, as an Object, really to a requirement. It is expressed through a boolean operator “true/false”. This attribute is very important for the

ID Code	IPS Requirements	Requirement Type	Allocation	Requirement
	1 Operational Requirements	N/A	N/A	False
SRS_1	The Ice Protection Sub-system shall permit aircraft operation without restriction in icing condition specified by applicable regulation.	Operational	General	True
SRS_2	The ice protection system shall be operated under all flight phase from take-off to landing when ice protection is required and taking into account the temperature envelope to which the AC is exposed.	Operational	General	True
SRS_3	The airframe ice protection system shall be operated under all flight phase from take-off to landing, when ice protection is required and taking into account the following AC performance: •Climb Rate = 3000 ft/min •Time to climb (@FL350) = 12 min •Climb Speed = 250 kts •Operating Altitude = 35000 ft •Max Cruise Altitude = 41000 ft •Max Cruise Speed (@35000 ft)= 340 kts •Typical Mission Range = 1200 nm •Cruise leg distance = 900 nm •Cruise leg time (@ 35000 ft) =2.8 h •Descent Speed = 200 kts •Descent rate = 2000 ft/min •Descent time (@200 kts) = 17 min	Operational	General	True
SRS_4	The Ice Protection Sub-system shall permit aircraft operation without restriction in icing condition within One Engine Inoperative condition.	Operational	General	True

Fig. 4.22 Requirements of the industrial test case (Part 1)

ID Code	IPS Requirements	Requirement Type	Allocation	Requirement
	2 Functional Requirements	N/A	N/A	False
SRS_6	The de/anti icing system shall provide ice protection of the wing leading edges.	Functional	Wing/empennage ice protection	True
SRS_7	The de/anti icing system shall provide ice protection of the horizontal stabilizer leading edges.	Functional	Wing/empennage ice protection	True
SRS_8	The de/anti icing system shall provide ice protection of the vertical stabilizer leading edges.	Functional	Wing/empennage ice protection	True
SRS_9	The de/anti icing system shall provide ice protection of the engine inlets.	Functional	Air intakes	True
SRS_10	The ice-protection system shall prevent ice formation on the pitot and static tubes.	Functional	Pitot and static	True
SRS_11	The ice-protection system shall prevent ice formation on the Air Data Sensors (AoA sensor).	Functional	Angle of attack sensor	True
SRS_12	The ice-protection system shall prevent ice formation on the Windshields and Side windows.	Functional	Wingshields and side windows	True
SRS_13	The ice-protection system shall prevent ice formation on the Horns.	Functional	Antenna/ Radome	True
SRS_14	The ice-protection system shall prevent ice formation on the Water Waste.	Functional	Water line	True

Fig. 4.23 Requirements of the industrial test case (Part 2)

ID Code	IPS Requirements	Requirement Type	Allocation	Requirement
SRS_15	The system shall detect ice condition.	Functional	Ice detection	True
SRS_16	The system shall be able to measure the ice thickness level.	Functional	Ice detection	True
SRS_17	The system shall be able to measure the ice thickness accretion rates.	Functional	Ice detection	True
SRS_18	The ice protection system shall provide the monitoring of its health status.	Functional	Control and indicators	True
SRS_19	The ice protection system shall provide the monitoring of its operating status.	Functional	Control and indicators	True
	3 Safety Requirements	N/A	N/A	False
SRS_20	The system shall be designed to assure the safety levels prescribed by applicable regulation.			
SRS_21	The system shall be design considering redundancy for actuation and control equipment	Safety	General	True
SRS_22	The system shall allow the pilot to visually check the ice accretion level in case of failure of the ice indicators	Safety	Control and indicators	True
	4 Performance Requirements	N/A	N/A	False
SRS_23	The system shall be conceived in order to support different modes of operation to adapt to the severity of ice accretion.	Operational	General	True
SRS_24	The ice protection system shall have a maximum power consumption of 65 kW, independently from the selected configuration and/or from the operating conditions.	Performance	General	True
	5 Physical Requirements	N/A	N/A	False

Fig. 4.24 Requirements of the industrial test case (Part 3)

ID Code	IPS Requirements	Requirement Type	Allocation	Requirement
SRS_25	The total mass of the ice-protection system shall not exceed 150 kg.	Non Functional	General	True
	6 Installation Requirements	N/A	N/A	False
SRS_26	The ice protection system shall be installed on the wing with the following geometry data: •Wing Aerodynamic Reference Area 19.51 m^2 •Mean Aerodynamic Chord 1.7 m •Span 13.16 m •Aspect Ratio 8.8 •Taper Ratio (Tip Chord/Root Chord) 0.52 •Root Chord 2.05 m •Tip Chord 1.06 m •Front Spar Position (% chord) 15 •Rear Spar Position (% chord) 70 •Relative thickness (Root/Tip) 12 % / 6 %	Non Functional	Wing/empennage ice protection	True
SRS_27	The ice protection system shall be installed on the horizontal stabilizer with the following geometry data: •Tail plane Reference Area 5.25 m^2 •Mean Aerodynamic Chord (MAC) 1.2 m •Span 5.26 m •Aspect Ratio 5.3 •Taper Ratio (Tip Chord/C/L Chord) 0.44 •Root Chord 1.65 m •Tip Chord 0.73 m •Relative thickness (Root/Tip) 8 % / 4 % •Front Spar / Rear Spar Position 10% / 60%	Non Functional	Wing/empennage ice protection	True
SRS_28	The ice protection system shall be installed on the vertical stabilizer with the following	Non Functional	Wing/empennage ice	True

Fig. 4.25 Requirements of the industrial test case (Part 4)

implementation of the following phases of design, as it will be shown in next Chapters. It makes easier separating requirements and other objects during the analysis.

An example of the IBM DOORS® Formal Module with the aforementioned view is shown in Fig. 4.26.

As the Reader could realize, the above cited organization becomes useful in case of an industrial product development, where several subsystems have to be integrated. Actually, the requirements written by means of the described structure are carefully used within the MBSE design process through a straight *traceability* to establish a bright link between requirement and model objects. It is known that the requirements management tools offer many possibilities to implement the *internal traceability* among the requirements within the specification (Fig. 4.27).

Some relations can be established among objects of the modules, to specify dependencies from and to the selected requirements. Those relations can identify either a derivation or a simple influence that requirements apply each other. They allow keeping trace of changes and even of the consequences of each modification, directly within the specification. The IBM DOORS®, for instance, offers an option to define some specific *Link Modules* to summarize the topology of these links, directly within a single matrix view, similar to the example of relation between need and requirement of Fig. 4.21. Specific *linksets* are established between the *Formal Modules* (or even a single one), to specify from which module the link is generated and to which one is pointing. Each Link Module can summarize this kind of information for the different linksets. A sort of map is provided between requirements.

The activity of requirements elicitation is often rather difficult despite of the simple look of the lists of statements above depicted. Resorting to all of sources actually relevant for the application, defining the classes to order the list, introducing a suitable hierarchy of requirements take time and need to be carefully done. Nevertheless, once that this activity is finished a well driven path to the analysis is disclosed to the user, who can easily proceed in investigating the operations to be performed through the system, the functions to be exploited and the most suitable layout to be built up to fit those requirements and assure the overall integrity of the system, as is introduced in next chapters.

ID Code	IPS Requirements	Requirement Type	Allocation	Requirement
	geometry data: •Fin Reference Area 3.48 m^2 •Mean Aerodynamic Chord (MAC) 1.9 m •Span 1.75 m •Aspect Ratio 1 •Taper Ratio (Tip Chord/Root Chord) 0.62 •Root Chord 2.25 m •Tip Chord 1.4 m •Relative thickness (Root / Tip) 17 % / 10 % •Front Spar / Rear Spar Position 15% / 50%		protection	
	7 Maintainability, Testability Requirements	N/A	N/A	False
SRS_29	The system shall be designed in order to minimize the maintenance work load and skill level, guaranteeing the objective of increased A/C availability	N/A	N/A	true
	8 Cost Requirements	N/A	N/A	False
SRS_30	The Cost for purchase and installation of the de-icing system shall not exceed 70000 Euro	Performance	General	True

Fig. 4.26 Requirements of the industrial test case (Part 5)

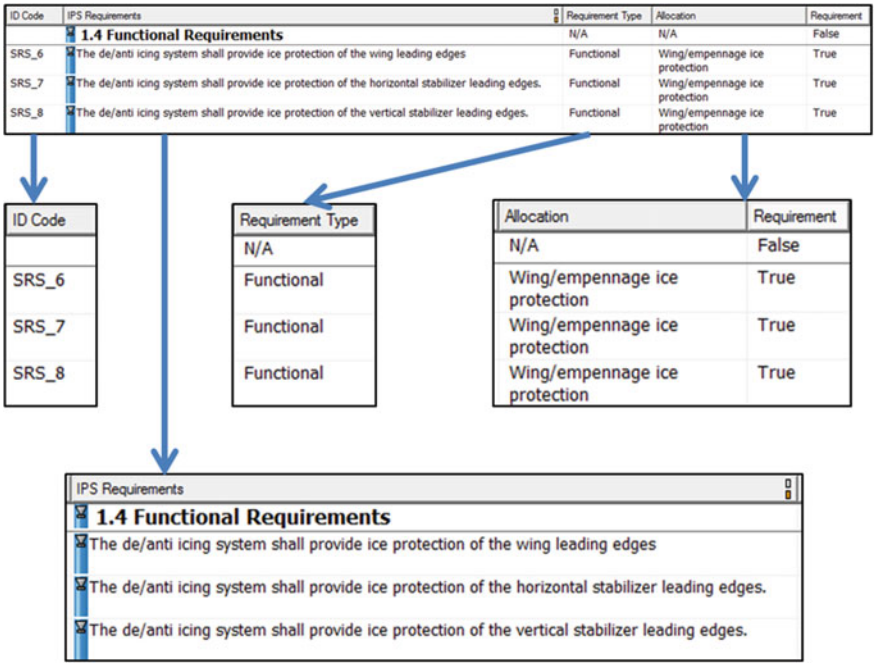


Fig. 4.27 Detail of the IBM DOORS® view within the main Formal Module containing the system requirements (industrial test case)

References

Michelli, J. A. (2016). *Driven to delight*. New York: McGraw Hill.

Pyster, A., Olwell, D., Hutchison, N., Enck, S., Anthony, J., Henry, D., et al. (2012). *Guide to the Systems Engineering Body of Knowledge (SEBoK)*. Hoboken, NJ, USA: The Trustees of the Stevens Institute of Technology.

Walden, D., Roedler, G., Forsberg, K., Hamelin, D., & Shortell, T. (2015). *Systems engineering handbook of INCOSE* (4th ed.). Wiley.

Chapter 5
Operational Analysis

Abstract Since this chapter, the different analyses performed by following the MBSE approach are proposed and deeply described. In the Operational Analysis, the system mission, context and scenarios are basically studied. The aim of the chapter is showing first how the Operational Analysis is deployed by resorting to some SysML diagrams. It is then applied to the so–called didactic test case, by resorting to a first software tool. It is then applied to the industrial test case, by exploiting some other tools, to show at least briefly, some differences. At the end of chapter, the main items of the Operational Analysis are resumed.

5.1 Goals and Tasks

The *Operational analysis* is meant to determine the system *mission*, the operational *context* and some *scenarios*, which describe the boundaries of the system activity.

Once that requirements are defined, the *context* of the system operation and its *boundaries* need to be identified. The system is assumed to be like a "black box". The operational analysis somehow describes what happens around that black box, and who acts, and it investigates what are the needs and the intended uses of the system, in operation. The major aim of this design step is motivating even the existence of the system under design. A deep understanding of the user needs in system operation is accomplished, and the system contributes to those needs are determined. Therefore, within the models developed by the operational analysis, the *environmental context* and the associated *operational scenarios* are derived. This action includes the identification of *actors*, defined by INCOSE as "entities that interact with the system through a *function*". The actors even correspond to some roles played by users or by any other external element, which interacts with the analyzed system. They receive and provide some *services*, within the evolving environment, depending on the context.

The *contexts* and the *use cases* help the systems engineer to check whether the product life–cycle development is comprehensive, complete and consistent. Moreover, some different behaviors and events, potentially produced by the

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_5

environment, are investigated. New contexts, actors, and use cases may be added and need to be considered. Therefore, the system must be either compliant or reactive to the environment evolution. Particularly, system architects must decide and predict what the system will be liable to.

This architecture might seem a little bit vanishing, if a deeper and structured definition is never added, but it is provided by the literature and by handbooks. The *environment* is restricted to a set of various *contexts*, being made of *actors* interacting with the system and playing a specific *role*. Performing the operational analysis means depicting what are the contexts of the environment, which actors are active in each one, and what is their role.

To implement that investigation, a *context vision* is provided. It focuses on the environment of the studied system, being often referred to as "*System of Interest*" (SOI), on the operational contexts, and on the relevant actors. It is crucial defining the *information exchanged* between the system and its environment, to characterize the *use cases*, where system, actors, information exchanged and functions are clearly stated. Furthermore, the system developer should circumstantiate the *operational data model* that specifies the physical structure of the exchanged information, and the operational requirements related to the operational model elements (use cases, information, contexts).

5.2 The Operational Analysis Deployed Through the SysML

Some diagrams can help the system engineer to analyze the context, during the Operational Analysis, as the *System Context Diagram* (SCD), the *Use Case Diagram* (UCD) and the *Sequence Diagram* (SD).

The *System Context Diagram* can be described in the SysML language through a Block Definition Diagram (BDD) or a UCD. It describes the external entities interacting with the system, there shown as a black–box. Since it provides a high-level view of the system, usually it is the first diagram defined. It is able to depict the environment of the system, the external elements or the actors to be taken into account, during the development of system requirements. Figure 5.1 shows a SCD described, for instance, by a UCD.

The relations between the system and the external entities are even better described by a *Use Case Diagram*, which defines the use cases that the system is going to perform, specifying its operational context. In Fig. 5.2 an overview of the UCD syntax is proposed.

The *use cases* are the functionalities of the modeled system and its goals. They are represented as an oval, with the name of the use case written inside, and are associated to some users, who are shown as some stick figures, with a label. The users of the system are described as *actors* and play a specific role in the system operation. They interact directly with the system or through another actor.

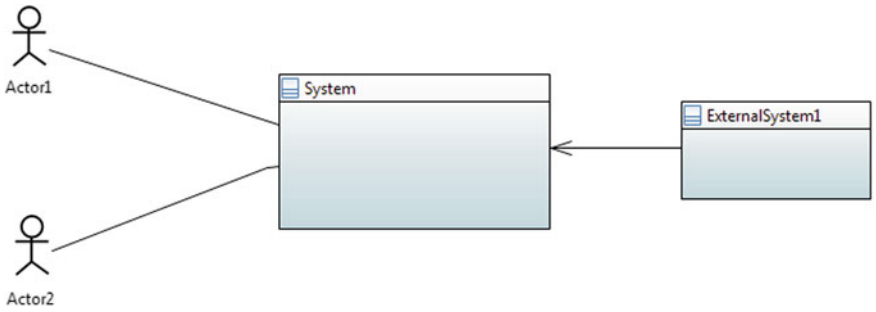


Fig. 5.1 Elements of a typical System Context Diagram (SCD)

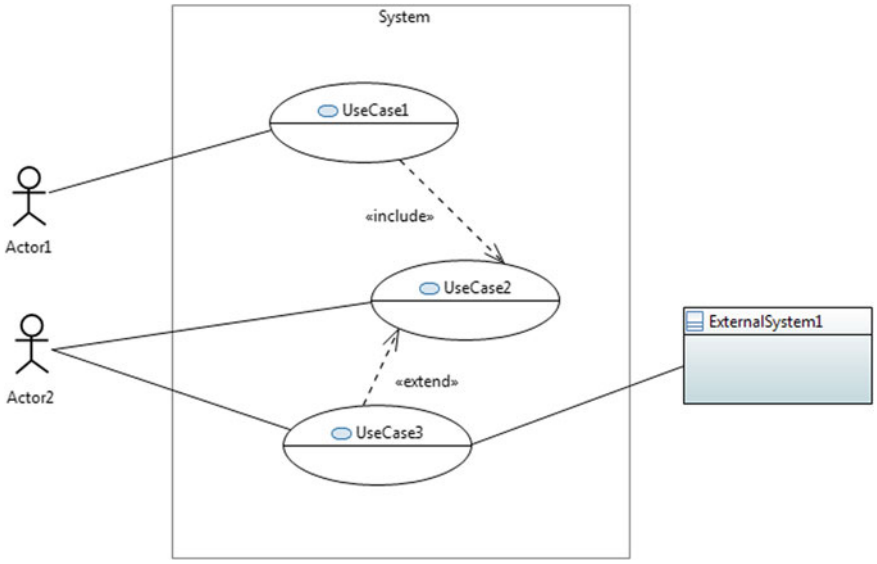


Fig. 5.2 Elements of a typical Use Case Diagram (UCD)

Moreover, the boundary of the system can be graphically represented to highlight that the depicted actors are external users and/or systems.

Besides actors and use cases, an important element is the *relationship* between them. Four types of Use Case relationships are described in the SysML language:

- "Association" between actor and use case.
- "Refine", "Include", and "Extend" between two use cases.

The *association* corresponds to a simple line, drawn between an actor and one, or more, use case. It states that the actor performs the linked use case.

A relation between use cases is described by arrows. A secondary use case can *refine* the main one, connected to an actor, by adding some properties or

functionalities. It might *extend* a use case functionality, with some exceptional behavior. The use case can *include* another one, if this performs a sub–function.

The *Sequence Diagram* is a time dependent representation of the data exchange, in terms of *messages*, among the entities of a use case. It is a very common representation, for the identification of interfaces between system and environment, during both the operational and functional analysis, respectively. Its implementation is oriented to input/output relations, based on request/answer structure, and it can be used also to translate the contents of the *Activity Diagrams* (Chap. 6) into a time dependent *sequence of operations* (hence the name). This is an interesting way to represent the flow of actions and operations, through many scenarios, to be further analyzed in terms of messages exchanged among the use case entities. Since each Sequence Diagram can be considered a description of a specific flow, the whole set of diagrams may represent the different situations in which the system shall operate. The *lifelines* in the SD (Fig. 5.3) allow describing the actors' interfaces and the actions performed, as a *sequence of messages* among the roles involved in the use case. In the meanwhile, the communication among different use cases can be described as well, if the cross–relations between *objectives* expressed by the *actors* (i.e. the *use cases*) are shown.

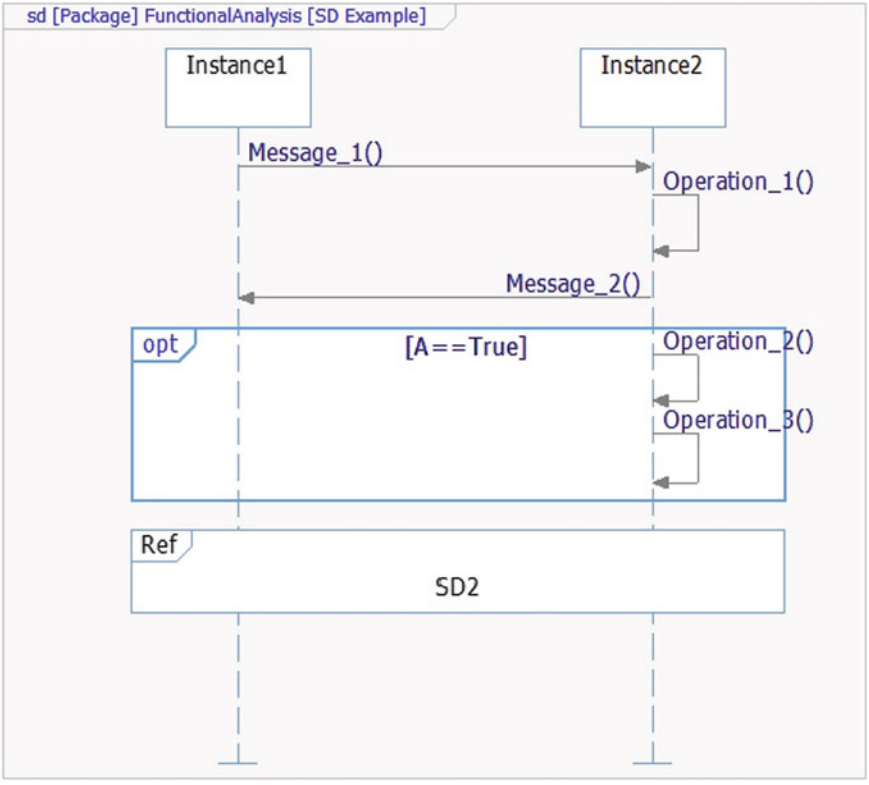


Fig. 5.3 Elements of a typical Sequence Diagram

Some tools use the *Sequence Diagrams* to formalize the operations defined in the Use Case and Activity Diagrams, to create a sort of database, to be exploited during the process. An example of SD is shown in Fig. 5.3. Two instances belong the same scenario. The Instance n.1 sends a message to the Instance n.2, that executes an operation and gives a feedback. Eventually, an optional set of operations can be selected, if the condition shown into the box is verified. At the end of the diagram, a reference to another SD is used to execute a predetermined set of operations. This is very often deployed to re–use the elements already defined or to enhance the integration within the diagrams.

The *Sequence Diagram* can be enriched with several SysML elements. It can represent various entities, exchanging messages and operations. Somehow it looks the most flexible diagram of the whole set of diagrams herein presented. It is helpful in characterizing the *white box* architecture of the system, which describes subsequent phases, by replacing the use cases with blocks representing the system's parts.

5.3 Implementation and Operational Context

The main contents of the Operational Analysis are now described by resorting to the test cases introduced in previous Chapter. The laying head system will be analyzed through the PTC Integrity Modeler®, while the IPS will be described through the IBM Rational Rhapsody®. The approach applied in both the test cases is quite similar, despite of some features of the software tools. The aim of this section is deploying the Operational Analysis tasks, and focusing some contents of the SysML diagrams. The requirements traceability is assured in both the examples by connecting the models to the database, where requirements are stored and managed by the IBM Rational DOORS®.

5.3.1 Didactic Test Case

In a material processing domain, no strict rules are followed during the operational analysis, because of the peculiarity of the analyzed system. Particularly, within the operational context of a smart steelmaking plant, the definition of system boundaries and of actors is never simple. Many actors interact with the system continuously, even without a trigger command or request. This situation unfits the formal definition of actor previously proposed, as an entity with a legitimate interest in the system, interacting through a function. From this point of view, it might be immediately realized that a straight application of the SE to the industrial product development needs to be conformed to the exigencies of a technical domain slightly different from those where this methodology was originally developed and assessed.

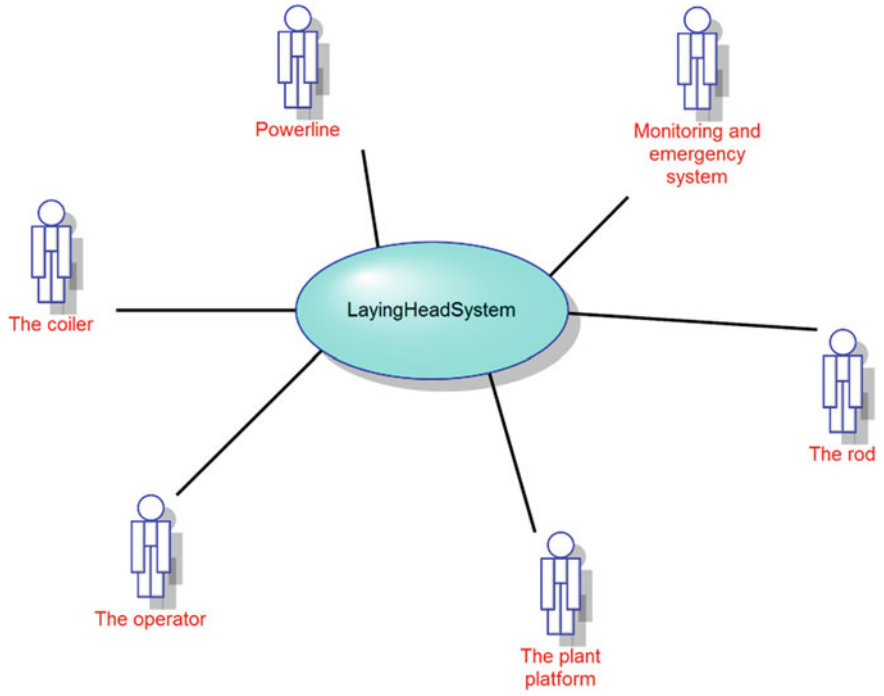


Fig. 5.4 The Context Diagram of the didactic test case

The above mentioned difficulty suggests to identify first the system boundaries, and all the actors interacting with the system. A Context Diagram is therefore drawn for the test case, as in Fig. 5.4.

The object of this analysis is the laying head system, that is making the wire rod spinning, and is as subsystem of the steelmaking plant. The operational analysis starts by identifying the actors of this system:

- The operator—person who operates the steelmaking process.
- The coiler—devices storing the coils of wire rod and building up the package;
- The monitoring and emergency system—central control unit which stops the laying head, in case of emergency, and behaves as an adaptive and actively controlled system.
- The rod—it is simultaneously a passive and active subject, which is shaped by the system, but affects its dynamics through the contact and the distribution of inertia while passing through the holed shaft.
- The plant platform—it is the floor required to bear the system, and to provide a reference for the electric grounding; it might be even a source of noise and seismic vibration.
- The powerline—it is a connected system, required to feed energy; it might be a source of irregular behavior in dynamics if the power feeding suffers some phenomenon like the flicker.

Starting from the different roles played by the actors during the system operation, a first set of use cases has been identified. Since the manufacturer intends to develop a mechatronic system, that shapes the steel rod as a coil and stores it, many customer needs assessed during the Requirements Analysis are related to the rod suspension and rotation, as well as to its shaping. Thus, in addition to the obvious services, like "Start" and "Stop the process", the system shall also suspend, rotate, shape and deliver the steel rod.

The three tasks related to the rod, plus the delivery, should be considered uncoupled, but, since they can be performed in sequence, the complexity of the system architecture can be reduced. For this reason, this test case considers the coupled use cases "Suspended and rotated" and "Shaped and delivered".

In Fig. 5.5, the main missions of the system derived from the collected customer needs are depicted. It can be noticed that not all the actors have been considered since the beginning. According to the definition of use case, the bubbles should represent how the actors actively interact with the system, but, in this case, some passive or *shadow* actors are considered as well. It means that they are never directly asked to perform a function, but the system simply involves them in its behavior, by exploiting their interaction.

The object of study in this case is a mechatronic system, therefore further details of the Operational Analysis deal with this specificity. According to the Requirements Analysis, more use cases need to be represented in this Use Case Diagram, including the power supply and the calibration of the system, as well as the ground supporting the Laying Head.

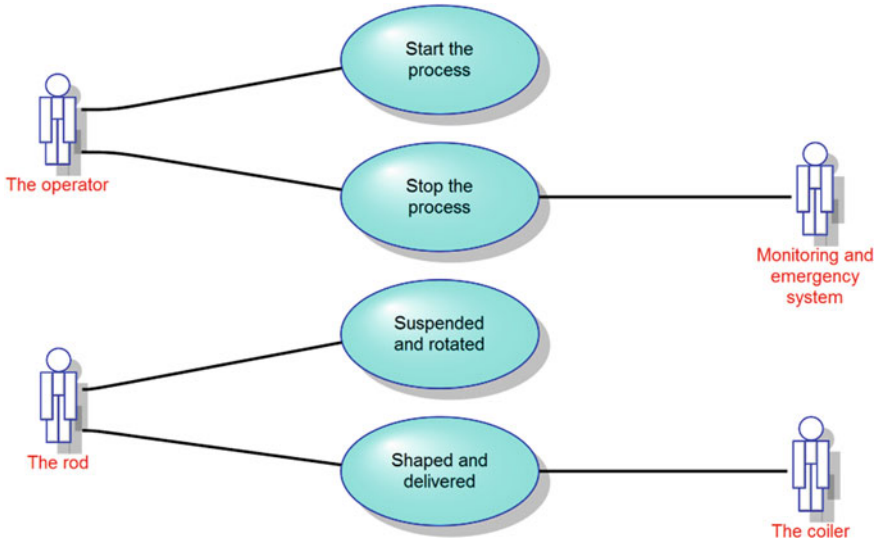


Fig. 5.5 First Use Case Diagram of the didactic use case

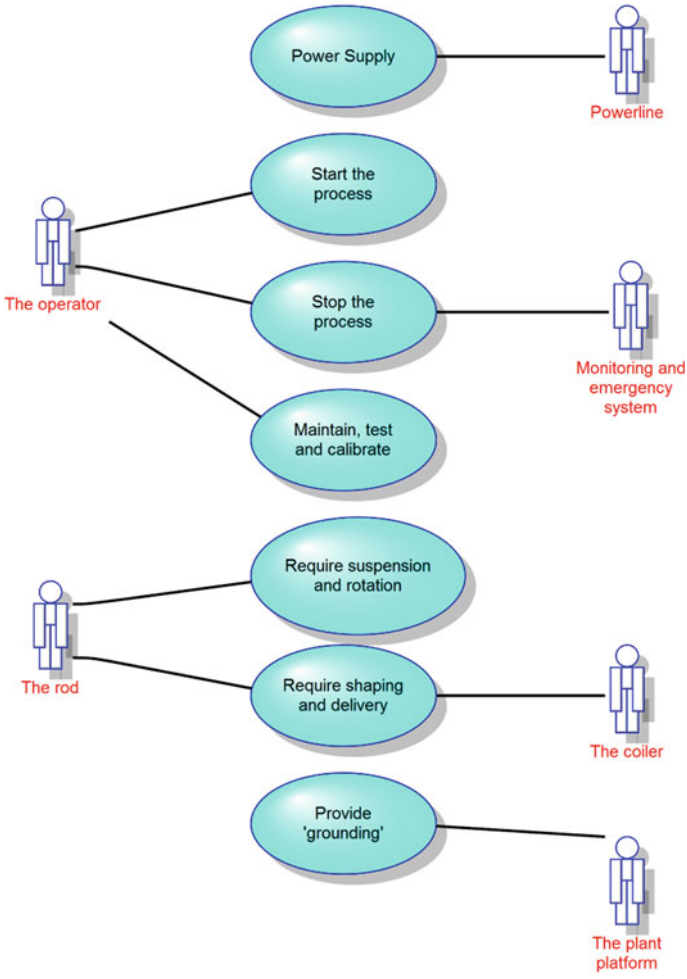


Fig. 5.6 The final Use Case diagram of the didactic test case

The diagram may be refined including all the actors and reformulating the use cases, as depicted in Fig. 5.6. The use cases of the laying head system can be summarized as follows:

- The powerline provides the power supply to the system.
- The operator wants to start the process in operation.
- The operator wants to stop the process in operation.
- The operator wants to maintain, test and calibrate the laying head system.
- The monitoring and emergency system wants to stop the process in case of emergency.
- The wire rod requires suspension and rotation.

- The wire rod requires shaping and delivery.
- The coiler is necessary for shaping and delivery.
- The plant platform provides the mechanical and electrical grounding.

This description motivates completely the role of stakeholders, although some actions are never required explicitly, but continuously and somehow implicitly. Some additional remarks need to be considered. The operator and emergency system have simultaneously access to the use case "stop the system", thus a level of priority, in case of emergency must be associated to each actor and defined during the system modeling activity, i.e. within the functional analysis. Moreover, the powerline and the plant platform provide support to the system without any query, but their activity must be considered in case of dysfunction of the system.

Differently from the industrial test case, here only the basic notation of the SysML language has been used, since this represents the first approach to SE for the manufacturer. Today the majority of SysML tools offer a wide variety of diagram elements to model a system, but this example demonstrates that few elements are really sufficient to provide a nice overview of the system to the user. However, some optional elements may improve the description and avoid any potential misunderstanding. For instance, in Fig. 5.7 the connection between use cases and requirements is better described if the related association is displayed as `<<refine>>`, in the requirement diagram.

The refine relationship, described in Table 5.1, allows specifying the use case not only to define the system mission, but also to refine further a requirement.

This step is important to assure the requirements' traceability and their consistency, during the entire modeling process. Tracing the requirements with the use cases can ensure that no one is missed and all requirements listed are fulfilled or, at least, are allocated.

The Use Case Diagrams define the environment of the system. They indicate the entities interacting with it, i.e. humans and external systems, but also describe how the actors use the system. This usage can be further detailed by the Sequence Diagrams. It represents the flow of information between the system and the actors, in a temporal sequence of events.

It is usually recommended to represent the system as a black-box on the right side, while the actors that interact with the system are kept on the left side. The Sequence Diagrams for this test case have been used to describe every use case defined in Fig. 5.6. This activity defines the information exchanged between the system and its environment to specify the system context.

Some sequence diagrams are herein depicted as an example, for the use cases "Start the process", "Stop the process", and "Require suspension and rotation".

The sequence diagram for the use case "Start the process" is shown in Fig. 5.8. This use case is connected only to the operator, being interfaced with the system to enable starting the steelmaking process. It is stated that the operator must send to the Laying Head System a startup signal to process the wire rod of steel. As soon as the Laying Head receives that information, it sends back a signal to inform the operator that the system is ready to start. Then, the operator may agree to start the

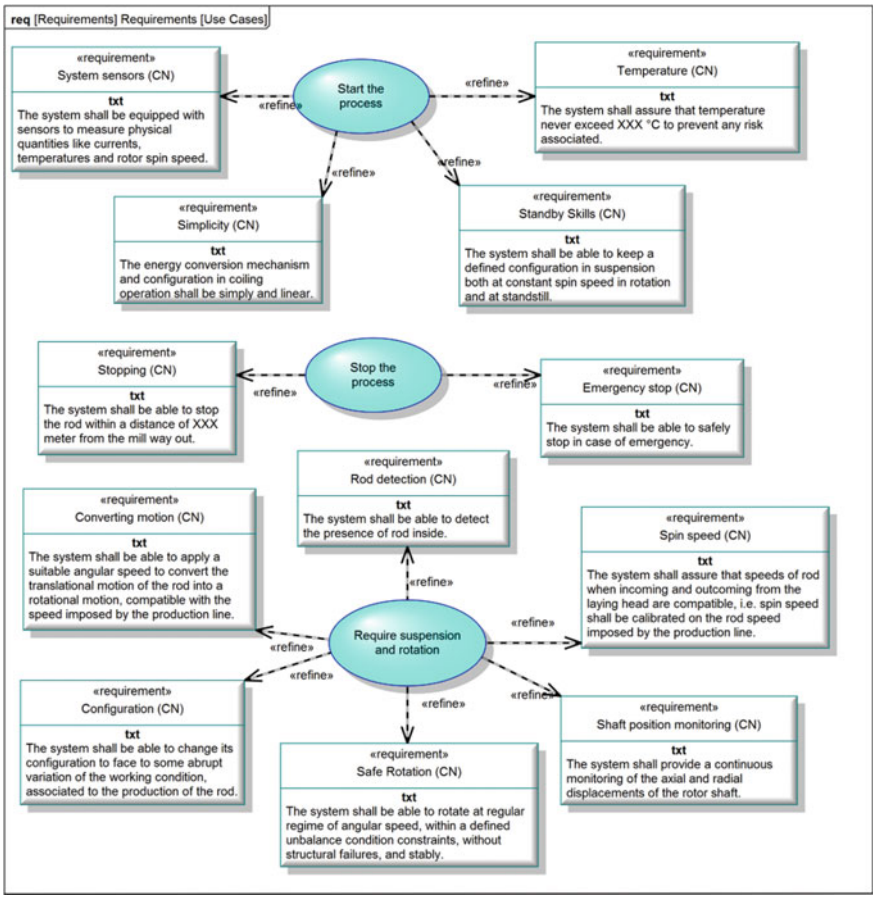


Fig. 5.7 Excerpt of the correlation diagram between use cases and customer needs

Table 5.1 Requirements Diagram, Types of dependency

Type of dependency	Meaning
Refine	The refine relationship describes how a model element (or set of elements) can be used to later refine a requirement. For example, how a Use Case can represent a Requirement in a SysML Requirements Diagram

process and sends the consent to the Laying Head. The sequence is very simple, since it represents just the interactions between system and only one external entity.

The diagram becomes more complex, when many actors are involved as in Fig. 5.9, for the use case "Stop the process". Here, a peculiarity is that either the operator or the monitoring and emergency system can stop the process. The Sequence Diagram exploits the alternate operator (Alt ()) to describe this scenario.

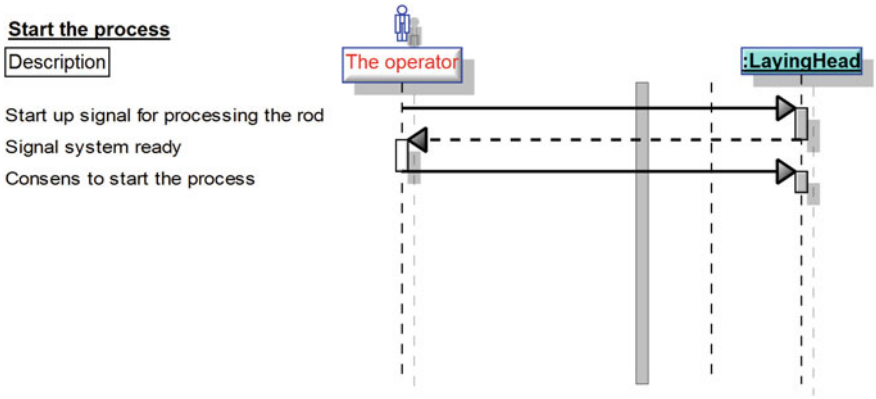


Fig. 5.8 Sequence Diagram for the use case "Start the process"

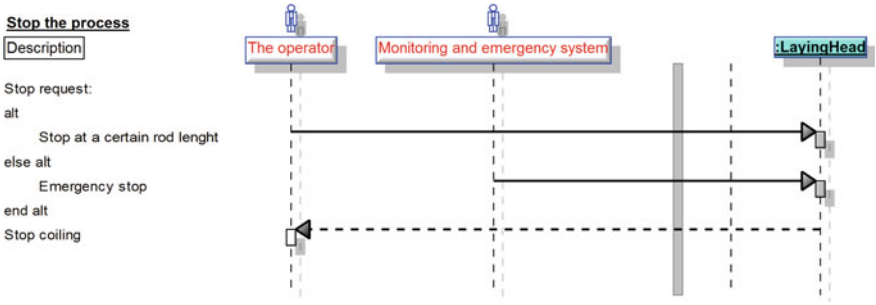


Fig. 5.9 Sequence Diagram for the use case "Stop the process"

The request to stop the process can be triggered either (`alt`) by the operator, which asks the Laying Head to stop the process, once that a certain length of the wire rod is manufactured, or (`else alt`) by the monitoring and emergency system, which requests to stop the process, in case of emergency related to a dangerous condition. When the Laying Head receives either of those signals, the process is stopped and a warning is sent to the operator.

The Sequence Diagram can include as many alternative actions as needed by the use case, using the Alt notation. The end of the set of alternatives is marked with the Type `end alt`.

In Fig. 5.10, another example is proposed. The use case "Require suspension and rotation" is described. The wire rod is suspended and rotated by the Laying Head and a safe levitation of the rotor is activated, only if it is monitored by the monitoring and emergency system. Thus, first the monitoring and emergency system requests to check the status of the rotor and the Laying Head gives back the required information. Only when the rotor is active and stable, the steel rod can be inserted. The rod requires to be detected by the Laying Head, that sends these

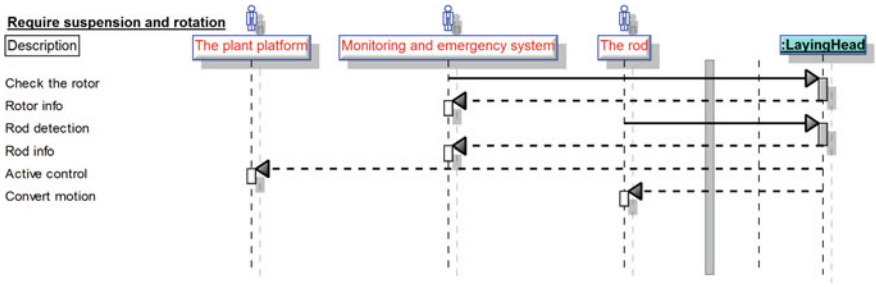


Fig. 5.10 Sequence Diagram for the use case "Require suspension and rotation"

information to the monitoring and emergency system. Once the rod is detected, the Laying Head warns the active control and the plant platform and converts the translational motion of the wire rod into a circular one, thus building the coil. In the last sequence, the system sends some messages simultaneously to the three actors involved.

5.3.2 Industrial Test Case

The aeronautical on-board equipment are a typical example of highly integrated system, where the operational context is characterized by a high level of complexity and a considerable number of interfaces with the actors, as it happens for the Ice Protection System (IPS) described in this section.

The complexity of the System of Interest sometimes induces the system developer to preliminarily define some main functions without considering the role of the operational context. This leads to forget or at least neglect some high-level requirements, defined in previous phase. This is a typical mistake, and might affect the effective integration of the Operational and Functional analyses, at the beginning of the modelling activity, although it is one of the most critical steps. Moreover, an even worst mistake consists in referring to an architecture already available on the market to develop the new system, instead of resorting to a level of abstraction, which should allow uncoupling the new concept and the technology currently used in known products.

This approach strongly affects the operational context, and often leads to identify a completely wrong scenario, in most of cases, because of some wrong use cases or actors, or some unsuitable relationships defined among them. To avoid this problem, it is extremely important proceeding by following a precise sequence of activities, i.e. starting from the requirements coming from the elicitation performed on the stakeholder needs, by paying attention to the level of specification at which each one was written. As an example, in the test case currently analyzed, among the Operational requirements, five statements can be identified:

- The IPS shall permit the aircraft operation, without restriction in icing condition specified by regulation.
- The IPS shall be operated under all flight phase, from take-off to landing, when the ice protection is required and considering the temperature envelope, to which the aircraft is exposed.
- The airframe IPS shall be operated under all flight phases, from take-off to landing, when ice protection is required and considering the aircraft performance.
- The IPS shall permit the aircraft operation, without restriction in icing condition, within the One Engine Inoperative (OEI) scenario.
- The IPS shall be conceived to support different modes of operation and to adapt to the severity of ice accretion.

Looking at those requirements, some remarks can be added. First, the system shall operate in a specific *mission*, since it is associated to the aircraft and the temperature and performance envelopes assumed as an input for the analysis. Some specific conditions are even foreseen, like the OEI emergency, defined by regulation, as the most of *environmental* details. Moreover, it can be clearly understood that *somebody* shall operate the system, using some *power source*. The system shall be also managed, in different ways, depending on the severity of icing conditions. This requires that some *air data* shall be required to perform the control action. It shall be then possible to *identify* the ice accretion and to proceed with the *ice removal*. Practically speaking, the following actors are identified:

- Pilot (or crew)—the person in charge of operating the system.
- Power source (or power system)—the device feeding the system. It is relevant appreciating that at present it is not yet defined what kind of energy or power could be selected.
- Air Data System (ADS)—is located on the aircraft, it is the subsystem which provides information about airspeed, altitude, air temperature, pressure, through some sensors and pitot-static ports.
- Environment—it reflects all the environmental conditions affecting the flight.
- Flight Management System (FMS)—is located on the aircraft, it is the system responsible for the flight management and navigation (route management, mission profile etc.…) typically collecting information from some other avionic equipment.

The identification of the actors above mentioned allows defining the Use Cases. It looks now easier, as the actors exploit some interfaces with the System of Interest:

- The pilot *wants* to predict the ice accretion on some selected surfaces of the aircraft.
- The pilot *wants* to detect the ice accretion on some selected surfaces of the aircraft.

- The pilot *wants* to remove the ice accumulated on the selected surfaces of the aircraft.
- The ADS *provides* the suitable information to predict the ice accretion on the selected surfaces of the aircraft.
- The ADS *provides* the suitable information to detect the ice accretion on the selected surfaces of the aircraft.
- The power source *is necessary* to predict the ice accretion on the selected surfaces of the aircraft.
- The power source *is necessary* to detect the ice accretion on the selected surfaces of the aircraft.
- The power source *is necessary* to remove the ice accumulated on the selected surfaces of the aircraft.
- The Environment *shall be considered (is associated)* to predict the ice accretion on the selected surfaces of the aircraft.
- The Environment *shall be considered (is associated)* to detect the ice presence on the selected surfaces of the aircraft.
- The Environment *shall be considered (is associated)* to remove the ice accumulated on the selected surfaces of the aircraft.
- The Flight Management System *shall be considered (is associated)* to predict ice accretion on the selected surfaces of the aircraft.
- The Flight Management System *shall be considered (is associated)* to detect ice presence on the selected surfaces of the aircraft.

The operational context can be summarized and implemented by means of the SysML language and specifically through the Use Case Diagram (UCD), looking in the IBM Rhapsody® like in Fig. 5.11.

As Fig. 5.11 shows, the Use Cases are connected to the actors through associations, whose meaning is specified by a label. It might be remarked that the relation between linked objects is generically described as an association, without additional details, but the link introduced will be recorded within the element properties, in the model, and it shall assure the complete traceability along the product development. The Use Cases are included within the so-called System Boundary Box, which identifies the boundaries between the system and all the external entities. It can be appreciated that the type of relation defined between actors and system affects all the following phases of the analysis. It drives even the trade-off of the system architecture, especially at interfaces.

The importance of this remark is due to the different kind of actors identified:

- Actors having a primary interaction with the system. In this case it is typically an on-demand relation, i.e. the system reaction is exclusively a consequence of some specific and unique requests of the actors. Each actor acts as a trigger for the system, whose response depends on the received input. This is the typical interaction between the aircraft pilot and the IPS.
- Actors having a secondary priority interaction with the system. They might be required for a proper operational behavior (i.e. the absence of the connection

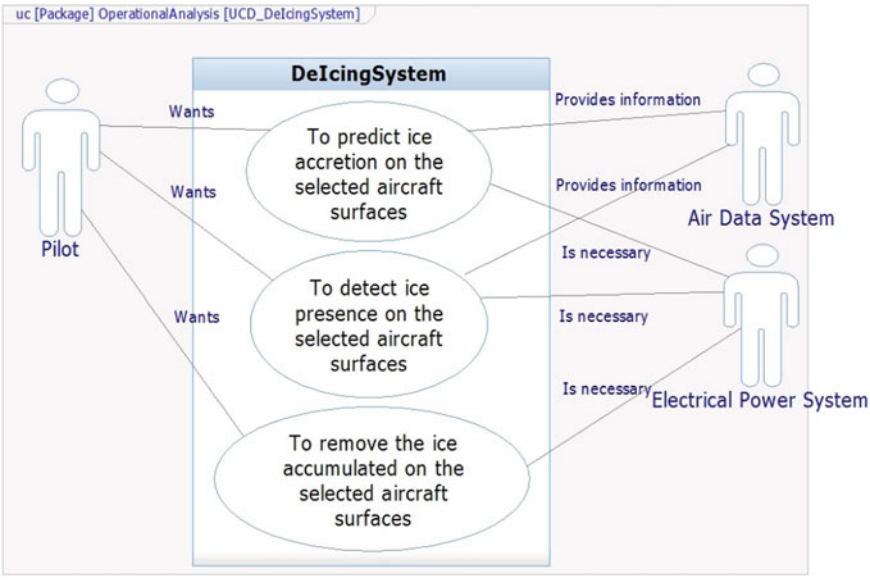


Fig. 5.11 The UCD of the industrial test case

may imply the inability to reach the objective), but they usually provide a continuous interfacing, without needing for a specific trigger. Those actors start providing some services or data to the system always under the control of primary actors, but, once the connection is established, they proceed without any further input. Sometimes, this kind of interaction is associated to the *shadow actors or stakeholders*, as they were already defined in Chap. 4. In the example of the IPS, the Electrical Power Systems as well as Air Data System typically behave like it was above described.

- Actors whose existence is never depending on the system. They are typically related to environmental conditions. They have no specific interest or objective, although their influence on the system behavior and performance might be relevant. For the IPS, the weather is the villain actor causing the problem of ice accretion, to be solved by the system. It cannot be classified as either a primary neither a secondary actor, although it provides the inputs to the system, to behave correctly.

There are many actors who can be defined in the Operational Analysis and particularly within the UCD. The analysis is mainly focused on primary actors, since they have the strongest impact on the system behavior, although even other actors are fundamental to define the whole set of system interfaces. Therefore, all the actors are included in the use case analysis, while some ones disappear in the subsequent diagrams.

In the industrial practice, an interesting detail to be considered concerns the so–called "pre" and "post" conditions described within the use cases. They identify,

respectively, the necessary conditions to start a use case and perform its activities, and the conditions reached after that the use case is completely deployed. The so–called *pre-conditions* sometimes involve some secondary or shadow actors. To be able to reach the defined target, for instance, the system must be connected to a power source, which looks like a "conditio sine qua non" to proceed. In some case resorting to pre-conditions simplifies the UCD, because the exchange of power, for instance, between actors is neglected, but it may be misleading, because pre-conditions are not linked to the properties of the use case, in some MBSE software tool and this might lead to a loss of traceability. This is due to the entity responsible to provide the power source, for example, being contained within the definition of the use case, instead of being a model object itself. Defining explicitly the whole set of actors within the Operational Analysis is usually suggested to assure the traceability of some critical models, although the use of pre-conditions and post-conditions is quite popular, when suitable supported by the software tool.

The UCD is very effective to summarize the operational context, since it is intuitive and self–explained by the SysML semantics. However, this is a simple example of UCD, just shown to describe the main steps of the implementation. The UCD, in fact, are even more complex and many elements may populate the diagram. Therefore, it is interesting to investigate the level of details to be kept into a UCD.

Let's consider the third Use Case above depicted. It might be further detailed, to derive even smaller Use Cases *included*. Figure 5.12 shows a more detailed view of the diagram, although the environment and the FMS are neglected to make easier the perception. Apart from the ice removal, which will be analyzed later on, some objectives of the pilot (or a generic operator) are related to the control and

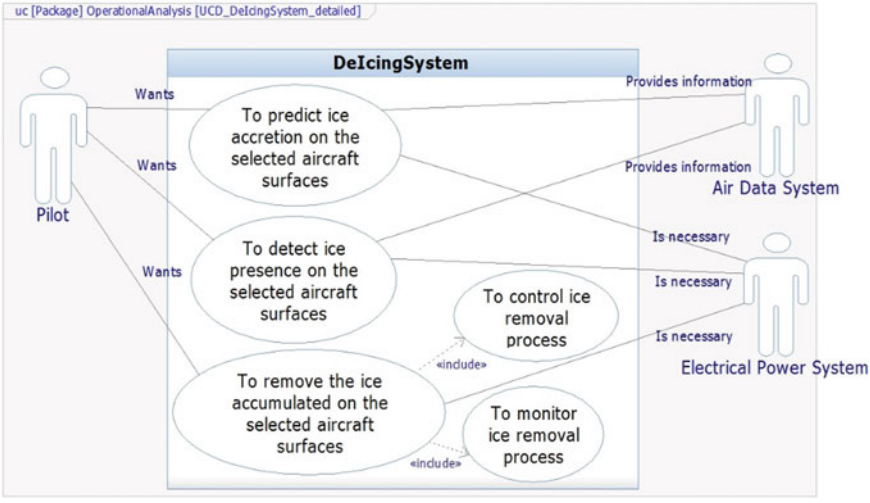


Fig. 5.12 Detailed UCD for the industrial test case

Table 5.2 Types of dependencies

Type of dependency	Meaning
Include	This dependency states that the source use case contains the objective specified by the target use case, which express a detailed item, within the boundaries of the source. It distinguishes a lower level use case, whose properties are similar to those of the father use case
Extend	The extension specifies a use case that identifies an objective being originally out of the boundaries of the target use case. Therefore the source use case is a special condition, not included within the scope of the target use case
Trace	The trace is a general-purpose dependency to instantiate a connection between use cases. The target somehow depends on the source, but this relation is not yet specifically defined

monitoring of the IPS. Two Use Cases are then created and linked to the father Use Case, through the *include dependency*. The dashed arrows suggest that the new Use Cases are part of the main one, but they can be analyzed separately, to reduce the complexity of the analysis, related to their *realization*. The main properties of the father Use Cases will be inherited by the sons. The association with the actors is neglected, in this case.

The example immediately brings to the dependencies used in the Operational analysis. These are "soft links", i.e. they specify a weak relation among elements, to be easily modified and updated during the project. Some types of dependency that can be instantiated are shown in Table 5.2.

The *Refine* links can also be used, as in the didactic test case (Table 5.1). Since the Operational analysis is located at the beginning of the project, the flexibility of links is fundamental. It is almost sure that the context will be updated several times. The Use Cases are the starting input for the following analyses, therefore they undergo a careful review process, to be well assessed and guarantee a good reference for the next steps of the product development. To perform that assessment several behavioral diagrams have to be compared each other. Particularly, the Operational Analysis here proposed uses the Sequence Diagrams (SD) to provide a more detailed description of the high-level *operations* performed in different *scenarios* and involving the system actors. This is a representation, in formal language, of the contents already depicted in the UCD, but it focuses on the specific exchanges among actors and use cases, simply described in the UCD by some *associations*. An interaction between actors and use cases, as shown in the SD, may occur only when they are connected by an association in the UCD. This correspondence allows organizing the analysis. It is an effective real-time verification of the completeness of the instantiation of each high-level relationship. This approach makes expedient understanding whether any association link is missing in the UCD, or any relation is unused.

The SD assumes different meanings along the MBSE design process. It is widely used, because of its simplicity and flexibility. Typically, the Sequence Diagrams are

defined considering the use case in the Functional Analysis, but within the Operational Analysis they are exploited to represent one or more specific *scenario*. In several examples use cases and scenarios are used as a synonym, somehow meaning that the use case may include already a considerable number of implications on the system behavior. According to this interpretation, the use case assumes a larger dimension, because it may represent a complete operational phase. In fact, this is never assumed to be the best modelling approach, because it imposes a system–centered point of view to the operational analysis. In fact, it should be used to investigate the interactions between system and actors.

The so-called *scenario* is then a sequence of *operations* and *events* describing the mutual relationships among the actors and the system, when pursuing a defined objective, which is specified by the use case. The scenario includes different use cases, especially when some dependencies are instantiated among them, as it was previously shown. In practice, the scenario is a wider instance, compared to the use case. The scenario is the formal description of a *context*, with a sequence of *operations* and *events* necessary to reach one or more actors' objectives, being specified by the use case. Some examples are proposed in following diagrams, dealing with the formal description of scenarios, for the IPS.

In Fig. 5.13 the ice forecast scenario is shown.

Some vertical instances populate the scenario depicted. The actors, represented by a thicker vertical line, are placed at the boundaries of the diagram, to point out that they are external entities. Three use cases are included as dashed vertical lines or *instance lines*. In the diagram, the time is increasing downwards and the *operations* and *events* occur along a sequence, at some specific instants, identified by

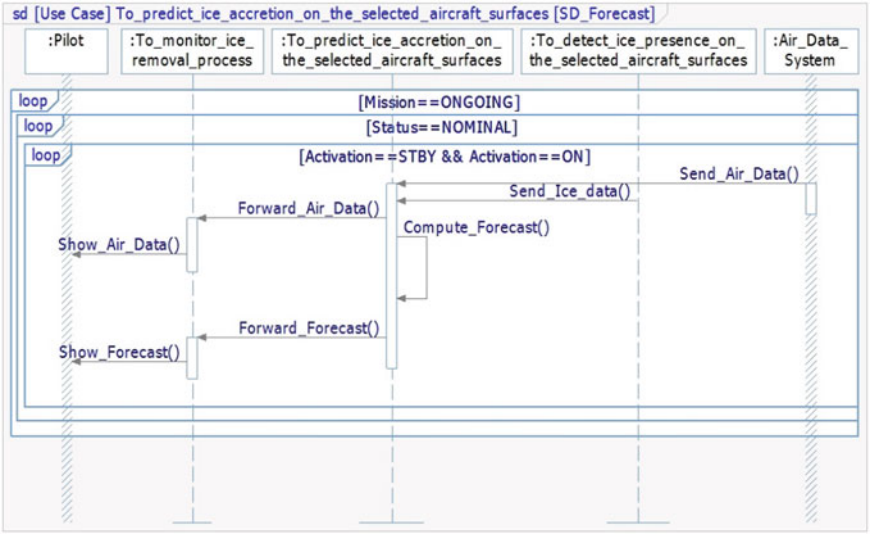


Fig. 5.13 The Sequence Diagram for the Ice Forecast scenario

horizontal arrows or lines. Some areas of the diagram can be subjected to a specific condition and the so-called *interaction operators* identify the rules applied to make the *operations* executed. The amount of time necessary to receive, elaborate and react to an incoming message is represented by a vertical rectangle, defining the extension of occurrence.

In this example, to describe a complete ice forecast scenario, the objectives related to the ice monitoring and removal process, the prediction of ice accretion and the detection of ice presence must be reached. The completion of those objectives requires that some operations are performed to actuate the whole sequence of activities. The actors must be involved in this process. Looking at the semantics, it is possible detecting a flow of interactions from the ADS to the pilot, who is able to read the air data and forecast the needed information, as soon as the ADS provides a prediction of ice accretion. The whole set of operations is active, when the mission is ongoing, the status is nominal and the supposed IPS is either in standby or in active mode. Looking at the *execution occurrences* (i.e. the white rectangles located on the instance lines to indicate the scope and the time required to perform an operation), it is possible to realize that some operations are performed in parallel. Some of those, as the computation of the forecast, require a longer time to be accomplished.

This example points out many issues of the modeling process.

- The scenario involves three use cases and two actors. They are connected consistently with the associations specified within the UCD, although not all the links are instantiated.
- The scenario is based on a mechanism of request and answer. All the elements use some information collected from others, to provide a tangible effect. Only the first operation, provided by the ADS, is a trigger to start the operations of this scenario.
- The scenario is referring to a *nominal* behavior. In aeronautical domain, this is usually referred as “sunny day” scenario, in comparison with the “rainy day” scenario, which is mainly related to failures, or non-conventional scenario. The terms “sunny” and “rainy” indicate a more probable and a rarer scenario, respectively, not only and strictly the weather conditions.
- Some preliminary hypotheses are formulated about the *system modes*. This is a typical feature of the operational analysis, since the assumptions are made by looking at the operational requirements already available. If these hypotheses are considered valuable, some new operational requirements can be added to the specifications, through a direct derivation from the diagrams.

This test case is suitable to add some remarks about the implementation. Operations and events are elements of the MBSE design and are stored within the model tree, as soon as they are defined. They are accessible and usable in following phases, since this step. For this reason, several tools use some specific strategies to save these data within the model, to assure their *traceability*. In the test case, operations are saved within the scope of the instance, which receives the data. For

instance, the operation "Send air data" is collected within the scope of the use case dedicated to prediction, instead within the actors' properties, although an evidence of the link between those two elements is saved within the properties of both instances. This structure is motivated by the fact that there is always a receiver, while some elements may not be responsible for sending data (as the pilot here). The operations internal to a specific *instance line* (looking as a closed-loop arrow) are stored within the element related to the instance line itself. These operations are foreseen, when an internal process is required to provide an output, starting from a received input. The operations are defined for the first time in the SD. A *realization* process can be even included, to allow a selection of the objects to be kept in the model and to avoid considering those to be erased. This motivates the relevance of the SD. It is used as a further verification mean for the confirmation of the consistency and correctness of the defined operations.

The *Interaction operators* are used to segregate the information and to introduce some rules to better describe the sequence. In this test case, all the conditions concerning mission, status and modes exploit a loop control to allow the execution of operations. Particularly, if the rule is valid, the sequence will be performed within a loop. Other types of interaction operators are available in the SD. Several ones will be used in following examples. Therefore, a list is proposed in Table 5.3.

Other scenarios help in understanding these topics and the difference between operation, event and message. In Fig. 5.14 the SD for the Ice Detection scenario is shown. Three use cases and one actor are included. Interaction operators are equivalent to previous examples, but some details are added. The flow of operations is deployed from the measurement of the ice thickness to displaying the ice accretion level to pilot. This sub-sequence is provided by a dedicated interaction operator, which can distinguish three levels of ice (high, low and minimal).

Table 5.3 Types of interaction operators

Type of interaction operator	Meaning
Opt	Optional interaction operator. It identifies a set of operations to be executed, under some specific condition, being unique and uncoupled with other ones. If the condition is not verified, i.e. the set of operations is not executed within the SD, no alternative solution holds and the sequence of operations skips to the main flow
Alt	Alternative operator. Meaning is close to that of the optional interaction, but an "else" condition is foreseen. An alternative solution can be activated. A dual condition including a dedicated sequence of operations is provided (as for true-false conditions)
Loop	Loop operator is used to keep the sequence of operations flowing in circle, if the condition is always verified. The flow is then stuck in loop, as the name suggests, until that the condition "false" is met
Parallel	Parallel operator is used to specify some branches of the SD, to be executed in parallel. No condition is required

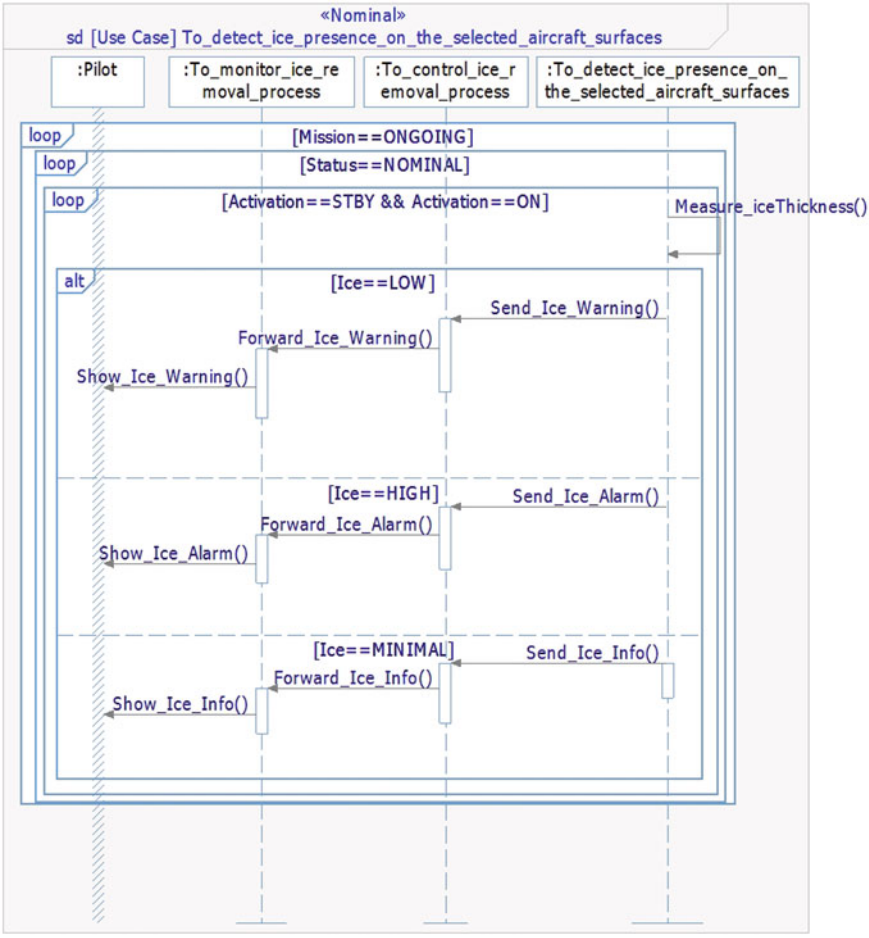


Fig. 5.14 The SD for the Ice Detection scenario

Figures 5.15 and 5.16 show the SDs for the Ice Removal scenario, being the main use case. Two SDs are drawn to facilitate the modeling process. A new model object referred to as *interaction occurrence* is included. This "black box" simply contains a reference to another SD, to be used to decompose the complexity of the representation. Instead of a unique diagram, quite complicated, this approach allows encapsulating a part of the scenario, to be also re-used in different contexts, which is analyzed separately. The part is identified by a box, including the name of the diagram to which it refers and a small *Ref* label in the upper left corner.

As Fig. 5.15 shows, in the first part of the scenario, the operations populate the sequence used to describe the effects expected in presence and in absence of ice, i.e. how the hypothesized "on" and "standby" modes are implemented. The second part deals with the sequence of system actuation. An interesting issue, at this point,

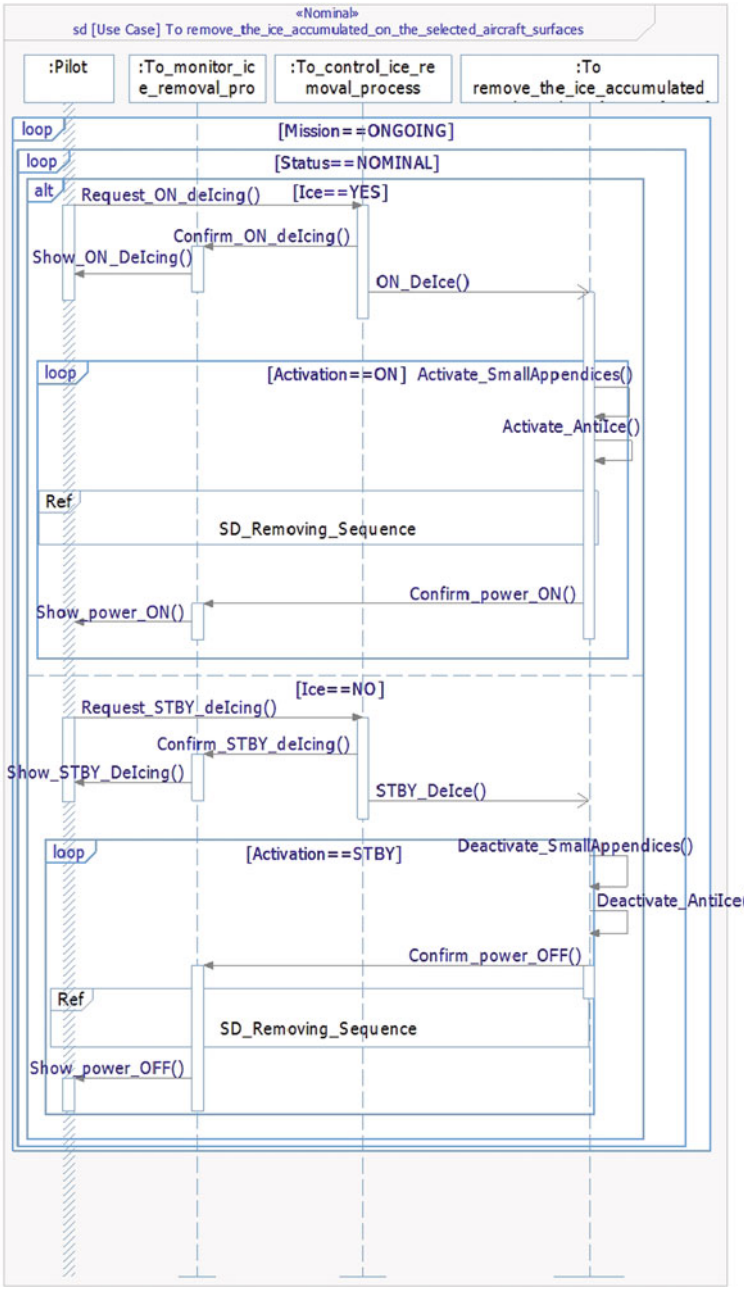


Fig. 5.15 The SD for the Ice Removal scenario (Part 1)

Fig. 5.16 The SD for the Ice Removal scenario (Part 2)

concerns the different functionalities of de-icing and anti-icing, to be clearly distinguished by means of some derived requirements.

The two SDs shown in Figs. 5.15 and 5.16 are surely fuller than the others previously sketched, because of the interactions and the number of operations. However, the minimum set of data required is there introduced. A goal, when using the SDs for the Operational Analysis, is keeping the model as simple as possible, provided that all important information are considered. The level of complexity of the scenario consequently increases with the number of involved use cases, and the amount of information they contain, but the overall set of interaction shall be kept at a level which might look as sketch of process. Nevertheless, it shall be better characterized through the following phases, just starting from this high-level concept. In Fig. 5.15, for instance, the number of operations and interactions will be surely higher than the unique one described, but the sequence represents the main core of process to be detailed later on. A specific structure for the interactions is used, when a request is sent by an actor. The Request-Confirm-Display *chain* is applied, as in most of systems where a user interface is required. Despite the simplicity, it corresponds to the core sequence actions to be described at this level of analysis. This is the typical relation between a primary actor and the *System of Interest*. Particularly, when the ice presence is confirmed and the pilot wants to activate the IPS, the sequence really operated looks as follows.

- The *pilot* wants to control the ice removal process and *requests* to switch ON deicing.
- In the nominal scenario, the *system* will *confirm* ON deicing, after a certain time.
- Since the *pilot* wants also to monitor the ice removal process, the system shall *show* ON deicing, by display.

In that sequence, the difference between the operations aimed to confirm the inputs and the real outputs is crucial, because they are both provided by the system, to reach the actor objectives. The confirmation consists in receiving and accepting the input, and requires some time and a least minimum computational effort to be executed. By converse, the operation of displaying the output is the result of computation. It identifies the execution of a command (show), and provides a real time information about the status of system operation. Similarly, in our daily experience we see that, on the elevator, the push-buttons of the consolle are illuminated to confirm the input, and display the current floor, to show the real output. This detail can be appreciated in Fig. 5.16, where the ice removal process is executed. Two confirmations are present to connect different use cases, being objectives from the point of view of each actor, because more entities are involved. This is a slight evolution of the RCD chain, in case of systems with higher level of complexity.

Extremely relevant is the correctness of the sequence of operations. The SDs must contain only sequences, which can be assumed as feasible and coherent with the domain application. It is highly probable that the sequence will be refined and the final system will have a different sequence of operations, in its final release, but the completeness and correctness of each one introduced in the development are

essential. In Fig. 5.16, for instance, when the activation command is set at "ON", the different zones are activated in sequence. They are deactivated only when the "OFF" command is provided. The activation loop may change, during the design process, to fit some requirements associated to both the functional and dysfunctional behavior, but, at this level, a general draft for the activation sequence is acceptable, to define the most important operations involved.

A crucial point concerns the use of SDs in the Operational Analysis of the concept of *segregation*. This approach allows managing and organizing the relevant information in some dedicated contexts, to involve only the entities primarily related to the process. In the Operational Analysis, this means creating some simple scenarios, to be eventually related to each other through references or interaction occurrences, being focused on some specific parts of the overall sequence, and involving a small set of actors and use cases. This motivates the separation of the ice control and removal processes, in Figs. 5.15 and 5.16. It is true that a typical feeling of users about this approach is that something important in the sequence could be missed or forgotten, but this risk is mitigated by the MBSE techniques. All the information are stored in databases accessible from any point of the model and highly traced. This approach provides a sort of map, that can be useful to check whether the set of data contained within the diagrams is coherent and complete. Moreover, the Operational Analysis, which is considered as a sort of sketch for any future improvement, is itself characterized by some uncertainty. Rather than considering big scenarios, whose completeness is realized by full-scale processes, it is better to rely on a completeness of smaller scenarios, to be reached through the capabilities of the MBSE tools applied to traceability of data.

5.4 Requirements Derivation in Operational Analysis

The requirements derivation process starts from the definition of high-level requirements, coming directly from customer needs, and goes through the different design phases. During each step, the requirements specification is updated and new relations are established between requirements and model objects. The best way to proceed, when deriving requirements directly from the model-based implementation, is to rely on the capabilities of SysML diagrams and semantics, together with the possibility of managing directly the connection between the MBSE design tools and the requirements database. In fact, although the high level requirements specification is used as a starting point for the Operational Analysis, the requirements derivation is notably a backward process, since it allows defining requirements directly within the SysML diagrams, and assures a consistent *synchronization* of information with the database. The requirements are first defined within the MBSE tool, and not within the specification, as in the beginning, although their reference remains the database, where the new requirements are recorded and classified. In the Operational Analysis is then possible to conceive three different steps for the requirements management.

- Connection between high level requirements and use cases (from the specification to the model).
- Derivation of new requirements, from the SysML diagrams.
- Updating of the requirements specification (from the model to the database).

The first step is typically faced through the instantiations of proper *trace dependencies* among requirements and use cases. This allows addressing the requirements onto specific areas of the design and directly connects the objectives represented by the use cases with the high-level specification. When instantiating the relations among requirements and other model objects, several SysML diagrams can be used. However, due to its specificity and to the available properties, the Requirements Diagram (RD) is one of the best platforms where these data are collected. As Fig. 5.17 shows, an instantiation of dependencies among requirements and use cases is performed, for instance in the industrial test case.

The direction of links indicates which element is dependent or independent. The architecture shown in Fig. 5.17 leads the requirements depending on the use cases. Therefore, they can be *refined* and *derived* in following phases of the analysis. When dealing with the operational and functional issues, it looks a good idea indicating the direction of dependencies, since this property suggests a subsequent detailed process. This approach basically agrees with the descending path described in the V-diagram. As soon as the logical analysis is concerned, a sort of verification process is required to identify which aspect is covered or *satisfied* by the logical elements of the system. In Chap. 7, when the connection between requirements and logical blocks shall be analyzed, it shall be even realized that dependencies will be used backwards, to evaluated the requirements coverage.

The SysML allows a even more schematic view over the established dependencies. Matrices and table are exploited as an alternative to the formal representation of diagrams. Within the software tool, it is never important how the link is established, because the records of the database are the same, although the representations are different and just in format. A matrix form is often used to summarize these data, as shown in Fig. 5.18 shows for the first use case of the industrial test case.

Once that the requirements are traced onto the use cases, the SDs are built to describe the operational scenarios of the system, as is shown in the previous section. In this context, the operations are *instantiated* together with the different interactions among actors and use cases. These operations specify new aspects of the system or new features, to be taken into account, especially for what concerns the interactions with users. As a second step, some new requirements are then defined, depending on the different operations, characterizing in a better way the *System of Interest*. The RDs can be again used to summarize the links and the dependencies instantiated among the model elements. An example of this process is depicted in Fig. 5.19, for the industrial tests case.

This is only a brief overview of the process, but usually a high number of requirements can be derived with this approach in the whole set of design phases.

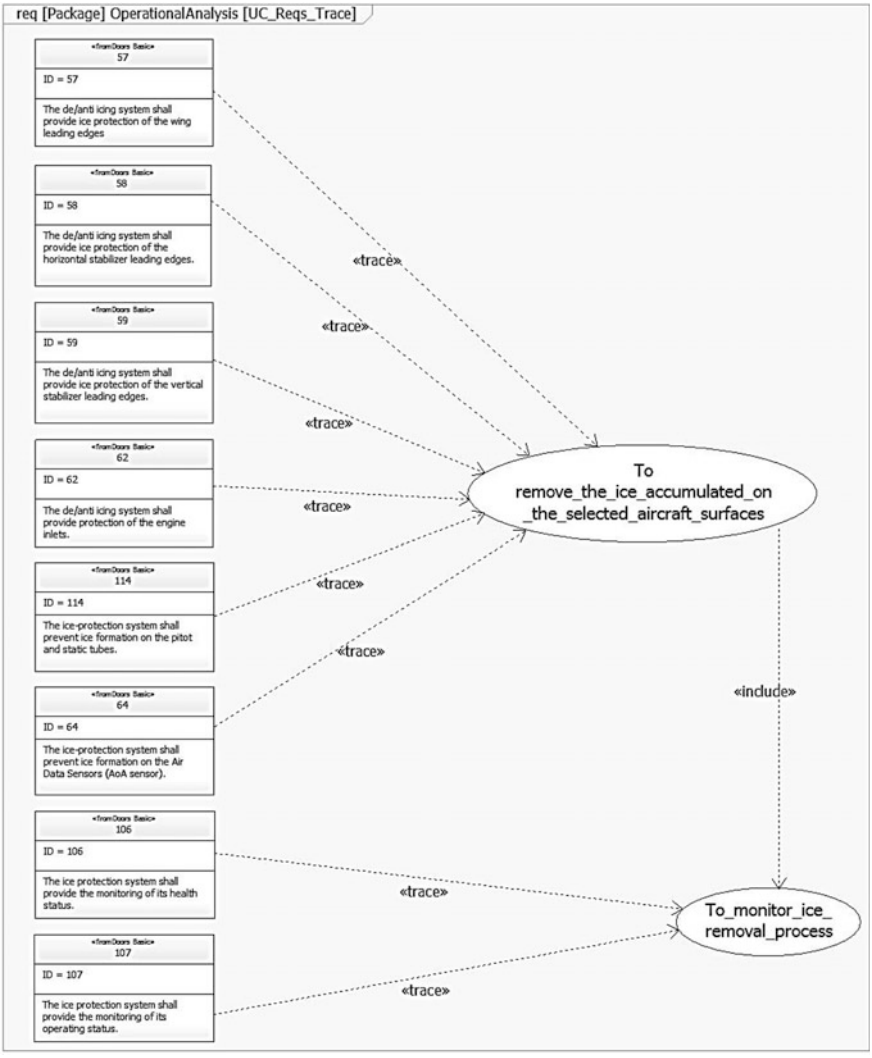


Fig. 5.17 Traceability among high level requirements and use cases in the Operational Analysis (Industrial test case)

In the industrial test case, for instance, it is possible to consider some new requirements.

- Receive mode from pilot: this operation, associated to the control use case, states that the system shall wait for the pilot decision to specify the system mode. The derived requirement is:
  - The system shall support the Flight Crew by providing the activation request message for each operating mode.

	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
57	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
58	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
59	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
62	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
114	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
64	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
60	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
61	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
63	To remove_the_ice_accumulated_on_the_selected_aircraft_surfaces
37	
38	
140	
106	
107	

Fig. 5.18 Matrix view (detail) for the first use case of the industrial test case

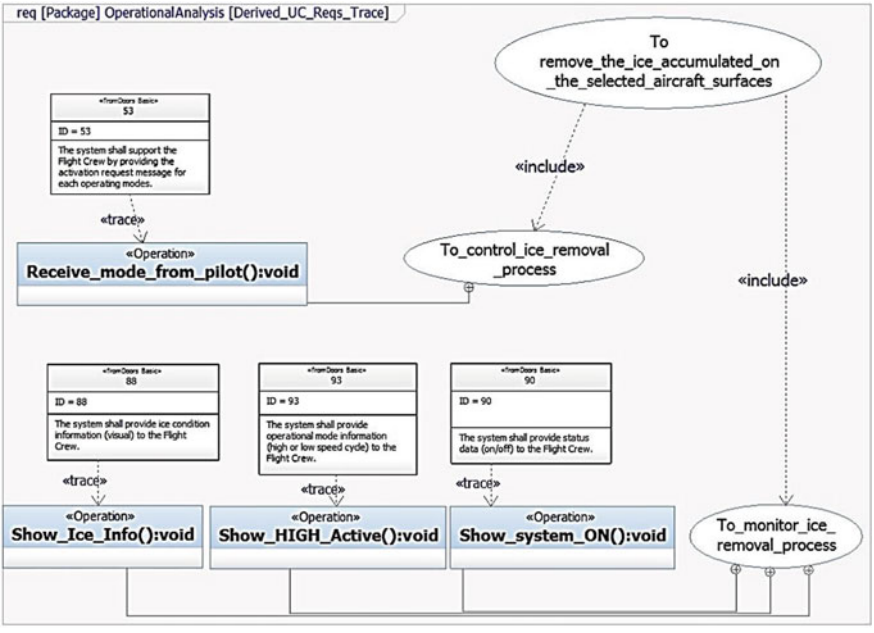


Fig. 5.19 The requirements derivation process from operations

- Show ice info: this operation introduces the display of ice information, being related to the monitor use case. The associated requirement is:
  - The system shall provide the ice condition information (visual) to the Flight Crew.
- Show HIGH Active: this operation specifies that the system shall show when "high" cycle mode is active. The requirement translates it into:
  - The system shall provide the operational mode information (high or low speed cycle) to the Flight Crew.
- Show system ON: this operation states the need to have an evidence of the activation of the system. The requirement responsible to specify this feature is:
  - The system shall provide the status data (on/off) to the Flight Crew.

This process can be repeated for all the operations, and the specification can be considerably enriched. All dependencies have always the "trace" stereotype and have direction requirement-operation (following the same concept specified for use cases).

Eventually, new requirements can be then registered within the database (the IBM DOORS® in this case), through some specific procedures, involving the MBSE software. This is usually referred to as *backward synchronization* and consists in the most powerful feature of the MBSE design, since the requirements are stored as soon as they are defined within the project. The embedded definition of requirements is quite effective, since it allows keeping the configuration control of model elements, instead of basing on documents and reports, like within the documents-based approach, where requirements defined in the Operational Analysis can easily fall in dark spots.

In all the three steps involving the requirements management and derivation, not only within the Operational Analysis, a crucial issue to be faced is the interdependency of model elements. A sort of optimization process is exploited. The coexistence of unnecessary elements is avoided. For this reason, each requirement is usually referred to a single model element, but one model element can generate more requirements, i.e. if two elements are associated to the same requirement, it is possible that one of them can be neglected or re-arranged in a different way. This is not generally true, when considering stand-by or redundancy systems, as in aeronautics. It is specifically requested to have different elements capable of performing the same functions or operations. This detail will be an issue of the Functional Analysis, in Chap. 6.

5.5 Synthesis of the Operational Analysis for Both the Test Cases

The Operational Analysis is surely one of the most important phases of the MBSE design process. It allows providing a first sketch of the operational *scenario* for the *System of Interest*, starting from the actors' objectives and high-level requirements.

This is the phase characterized by the highest level of creativity, for the designer, since it starts from the blank sheet.

Its main function is the definition of scenarios, through which the high-level behavior of the system is described. It is based on the *objectives* that the *actors* want to reach and on the instantiation of proper *operations* to allow the interaction among the different entities. The Use Case and Sequence Diagrams are typically used, in a combined process, relaying on *traceability*.

Some important issues have been highlighted.

- Actors are external entities interacting with the considered system. They want to reach some objectives, by exploiting the system. Depending on the way they are related to the objectives, they can be classified as primary, secondary or shadows.
- The objectives of actors can be modeled as use cases, which act as the main reference model objects for the entire MBSE design of the proposed test cases.
- The connection among actors and use cases is implemented through the SysML *associations*, being the basis for defining scenarios and interfaces with entities.
- The use cases can be detailed through different types of *dependencies*, to express some dedicated or low level objectives allowing the traceability.
- The Sequence Diagrams can be used to define the scenarios, which include the sequences of interactions among actors and use cases and represent their exchange of information.
- The Use Cases (in the didactic test case) and the Operations (in the industrial test case) are the main model elements of the Sequence Diagrams. They provide a first definition of the system behavior and of the architecture of input/output with actors, to be detailed in subsequent phases of the MBSE design.
- The sequences described in the Operational Analysis may be subjected to some change, during the MBSE design. Therefore, a deep detail is not required in this context, although feasible sequences are developed, to be compatible with the domain in which they are performed.
- In the industrial test case, many typical interactions among actors and use cases were explored. They concern a common scheme for the data exchange, based on the Request-Confirm-Display triplets, for active and primary actors, and continuous interfaces, for shadow actors and stakeholders. The SDs typically deal with primary actors, although the operational scenario may include also the effect of the relations with secondary ones.
- To model the interactions, different SysML elements and semantics can be used, within the sequence diagrams, to segregate the information, by creating some small scenarios, exploiting the completeness through the traceability.
- The Use Cases can be related to the Requirements, and to the Customer Needs, with a *refine* relationship, that identifies either the requirement or the use case to be refined.
- The requirements can be derived directly from the model elements of the Operational Analysis and embedded within the diagrams. As a first step, high-level requirements are traced onto the use cases, and the requirements tree

can be built, depending on the instantiated relations. New operations and modelled interactions among entities in the SDs can be used to derive new requirements, always using soft dependencies, to populate the low levels of requirements tree, being implemented through the Requirements Diagrams. The MBSE capabilities of traceability allow tracing the new requirements and their relations with the model elements. Moreover, the updating of the requirements specification is guaranteed by the connection with the database managed by the IBM Rational DOORS®.

The Operational Analysis may be implemented in different ways, and with some other diagrams, different from those used in this handbook. The semantics of diagrams themselves can be different, depending on the use, which is expected for the model, and on the way in which the designer wants to describe the model objects. The SDs, for example, appear in many variants, in literature. The definition of operations, the relations between use cases and actors, as well as the final format can be subjected to extensive modifications. There are also many differences when the model is designed for a dynamic verification or simply for a validation through the traceability and impact analysis.

As it will be shown in Chap. 6, for the Functional Analysis, if a diagram shall be executed and animated to test its correctness, a higher attention to the formal representation and coding of its elements shall be paid, to avoid debugging issues. If the diagram is conceived to provide a static view of the described scenario, more flexibility is allowed, when defining model objects, although a coherent representation of elements is always necessary to obtain a suitable level of traceability within the model.

Chapter 6
Functional Analysis

Abstract The next step of Functional Analysis is herein investigated. The existing relation between Operational and Functional Analyses is first defined. The implementation of the Functional Analysis is then performed, through the SysML. How the traceability and the requirements allocation are assured is shown. The system behavior is fully analyzed. The functional architecture is then derived, for the two test cases. Finally, the main items of the Functional Analysis are resumed.

6.1 Introduction

Notwithstanding the apparent simplicity of the terms *Functional Analysis* and *function*, these notions are often faced improperly, due to lack of a common nomenclature within the design process and to the influence of different engineering domains and practices (CRYSTAL project, 2015). In fact, depending on the context of the analysis, the process used and the tasks performed, these terms may assume several meanings. The best way to identify a general accepted definition is to rely on SE standards (listed in Chap. 2) and glossaries of concepts[1] that generally establish a strong basis for common knowledge, not only in terms of the single notion, but also for the complete and consistent set of relations with other ones. With regards to this aspect it is possible to define a *function* as the representation of what the system shall do to realize a specific *need*, usually expressed by a *stakeholder*, within a defined *scenario* of the selected *mission*, which can be formally stated as a *requirement*. Consequently, the *Functional Analysis* is the *iterative* and *recursive* technique of identifying and describing the functions of a system, aimed at defining its *functional architecture* (Friedenthal, Moore, & Steiner, 1999) As stated in the SE standards and glossaries, the function itself can be expressed as a specific sequence of lower level *operations*, which represent its practical imple-

[1]In addition to those proposed in the main literature, in handbooks, some specific glossaries were assessed within some funded research projects or by some associations as the AFIS—Glossaire de Base de l'Ingénierie de Systèmes (www.afis.fr) and the CESAR Project—Global glossary, among the project deliverables.

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_6

mentation. For this reason, a function may be considered not as an atomic element, but as a part that can be further decomposed and analyzed. Several discussions are undergoing about this open issue, since the decomposition of a function usually leads, within the typical approach of SE, to the identification of derived functions, up to the lowest possible level, not directly involving operations. As a consequence, functions and operations appear not to be the same thing. However, especially when implementing the approach through models, the straightforward use of operations as functions is quite common. This is actually very popular in MBSE methodology, because of the adoption of specific semantics and diagrams, which allow defining the system behavior, even without a tangible representation of a *functional tree*, focusing on the sequence of operations. This detail will be clarified later, in this chapter, when applying these concepts to the test cases. Apart from the actual use, a function is always aimed at reaching an outcome, i.e. a stakeholder need or expectation concerning the system behavior in a peculiar operating environment. This is also the reason why the relationship between functional and operational aspects is so tight, causing sometimes problems concerning their separation, when performing the analyses. Finally, a function is always referred to a requirement, particularly a *functional requirement*, which is a statement that identifies what a product must accomplish, to produce the required results (Chap. 7).

The Functional Analysis can be interpreted as a general "technique", which is focused on the identification of system functions and on their interactions. This is a critical point and one of the most common reasons for misunderstandings in SE. The Functional Analysis of SE, independently from the use of a model-based approach, shall be always aimed at obtaining a *functional architecture*, defined as an arrangement of functions, sub functions and their interfaces that describes the execution sequencing, conditions for control and data flow (Friedenthal, Moore, & Steiner, 1999), rather than identifying the *functional tree* (i.e. the diagram describing the hierarchical decomposition of system functions). The so-called *functional breakdown* by itself is not sufficient to consider all those kinds of relations and links (internal and/or external) among functions and the environment. Moreover, with the adoption of MBSE methodologies and tools, the functional architecture has been established as a *dynamic* representation of the *functional behavior* of the system and its interconnections, which can be updated continuously with the development of the analysis both on the same system level (iteration) and on lower systems levels (recursion), overcoming the static representation of a hierarchy model relaying on a simple tree.

Generally, there is never a single way to represent the different phases of the SE design activities (Holt and Perry, 2013) and even the Technical Standards (Chap. 2), which are a little bit more adopted cross-domain, and provide different set of combinations of phases. This is a clear proof of the size of SE approach for complex systems and of the difficulty of adopting the optimal process to face this challenge in many domains. For the purposes of this handbook the term *Functional Analysis* will be adopted since now to identify the process of defining the *functional architecture* of the system and the *functional breakdown*.

6.2 Handoff Between Operational and Functional Analyses

The Functional Analysis in SE has the objective of defining the functional architecture of the system and characterizing its functional behavior. The starting point is, of course, the result of the Operational Analysis, where stakeholders' needs have been already identified as well as the main capabilities that the system shall be able to support. The key point of the handoff to the Functional Analysis is the different view that the designer shall consider. The Operational Analysis is focused on what the stakeholders want to carry out and what the actors want to reach by using the system, so, it is notably a stakeholder-centered phase. The Functional Analysis focuses the attention on the functions that the system shall be able to provide to meet the stakeholders' requests; indeed, it is a system-centered phase.

The different approaches used to face the design between the first phases usually affect a lot the following ones, being the most important feature that allows to distinguish different methodologies. Not surprisingly, this is also another critical point that can lead to some technical errors, when the analyses are coupled in an inappropriate way. In any case, one of the most successful solutions of the MBSE approach, as was described for the Operational Analysis, is the *use-case-based* rationale, aimed to drive the Functional Analysis upon the goals and/or capabilities requested by the stakeholders. The *use case* (Chap. 5) is characterized through the definition of the system behavior, in terms of operations (i.e., functions) necessary to reach the expected outcome. Many SysML diagrams can be used for this particular task, as the *Activity Diagrams*, *Sequence Diagrams* and *State Machine Diagrams*, whose sequence, order and combination are strictly related to the method used for the analysis (OMG, 2015).

To formalize the functional architecture, some structural SysML charts like the *Internal Block Diagrams* and *Block Definition Diagrams* can be deployed, enriching and connecting the amount of information coming from the behavioral ones. One of the advantages of the MBSE approach is that the result of the analysis can be also directly assessed and verified through a logical or formal simulation, allowing a quick feedback about the expected systems characteristics. Some functional scenarios may be assessed and the relations among the system and the external actors can be analyzed at different levels. The MBSE methods using the SysML language are considerably relying on a verification through simulation, because of the number of possibilities offered to the designer to operate. In this context, the simulation is the very last step of the Functional Analysis. It is used to verify that the system functional behavior, architecture, and breakdown structure have been correctly derived, in accordance with stakeholders' expectations.

6.3 Implementation of Functional Analysis Through the SysML

As stated in Chap. 3, many formalisms and diagrams of the SysML may be used to traduce the Functional Analysis into a concrete implementation. They are applied to several design phases, therefore their meaning should be herein clarified. A general classification of the SysML diagrams was already proposed in Chap. 3. It looks to distinguish the formats used in different phases.

The behavior diagrams represent a standardized flow of data to describe the system status and transitions, under specific input, and within a given scenario. Namely, those diagrams are built around the main concepts of *action*, *operation* and *state*. The *Activity Diagram* is a flowchart of actions. Each *action* is one of the main tasks that the system shall perform within a workflow. To achieve the task goal, some steps, namely the *operations*, are required. Thus, the *action* is a sort of collector of operations, which are the basic bricks necessary to define the system behavior. Each *activity* is the collection of *actions*. It is remarkable that an *action* is not a *state*, since it does not define a change of system status. It is focused on the performed operations and never on the system modes. The interactions with actors can be implemented onto the actions, when the interfaces with elements outside the system are required, through some simple "pin" connectors. The so-called *events* can be defined within the different branches of the flow, to establish some specific occurrences (either continuous or discrete). They could be important for the system behavior, when they trigger another set of actions or they set some conditions, at the nodes of flow. The *Activity Diagram* is usually the first used in the Functional Analysis to describe the system behavior, in each use case. It exhibits a high level of abstraction, a simple structure, an easy way of representing data. It supports both the "*black box*" and "*white box*" views, i.e. it can be drawn even without including a clear reference to the elements performing the actions or, by converse, it can be expressively conceived to allocate those actions to logical and physical entities, in subsequent phases of the analysis Fig. 6.1 shows a portion of the "*black box*" version of a typical *Activity Diagram*. This is a common representation of a data flow, going from the reception of an *event* ("Start") to the selection of the branch (flow of actions). The depicted diagram shows also some elements above mentioned, as *actions*, conditional connectors, *call behavior*, *actor pins*, here sent the pilot and a next *event* ("Send").

The *State Machine Diagram*, or *Statechart*, is a representation of the evolution of system *states* and of their *transitions* within a use case or a scenario, depending on inner and outer inputs. It is a powerful tool to be used to evaluate the functional behavior of a system, since it can be verified by a simulation. The *State Machine Diagrams* contain all relevant information coming from other diagrams, and a simulation can be a reliable way to test the nominal behavior of the system, in terms of operation modes, as a synthesis of the whole functional design. A *state* represents the status of the system at a defined time, for given inputs and within an execution path. Operations are included and performed during the activation of a state. The state may

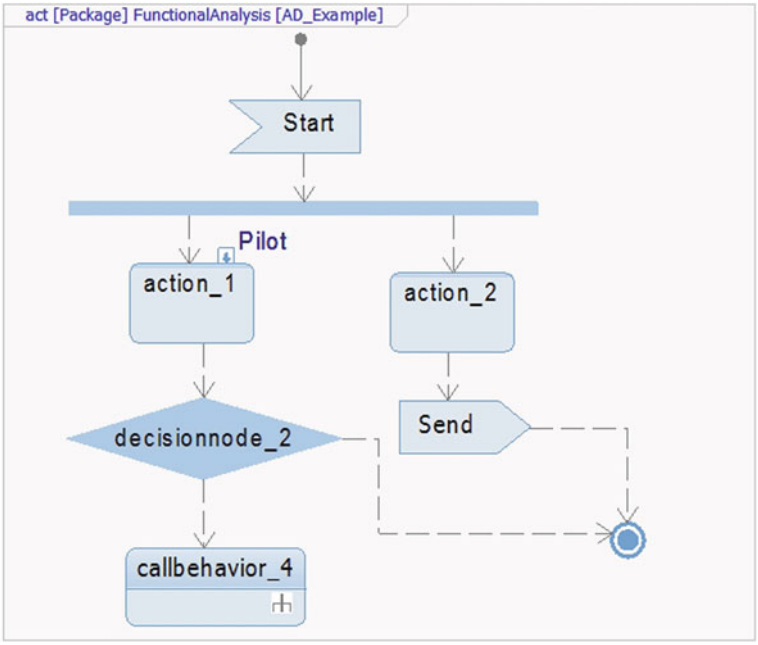


Fig. 6.1 Elements of a typical activity diagram within the functional analysis

also contain other elements, like logical and mathematical equations, to process the received inputs and produce some related outputs. The State Machine Diagram can be even chosen as a unique behavior diagram to describe the whole Functional Analysis, when the system behavior is strongly characterized by states, or as a last step of the Functional Analysis. A nested architecture is possible also for the *State Machine Diagram*, and the communication between diagrams is performed through the instantiation of some dedicated *Internal Block Diagrams*. In the Functional Analysis, this approach is aimed at connecting the entities within the use cases.

Figure 6.2 shows an example of State Machine Diagram. Its structure is similar to the *Activity Diagram*, although the main elements represented are *states*. This difference is relevant, the *state* may contain many *operations*, therefore it provides a wider view on the system behavior than the contents of the *Activity* and *Sequence Diagrams*. States are connected through *transitions*, which specify the evolution of the followed path, depending on the *guards* and *triggers* which activate and deactivate the flow. The example shows a typical structure of the SMD, with two main states. The first one ("State_0") is usually used to specify a passive behavior (such as "switch-off" or "wait" states), whilst the second concerns the active behavior of system, and is often more complex. "State_1" contains two branches, being executed in parallel, where some other sub-states are connected to each other, depending on the guard condition specified on the transition (here not shown on the right to simplify the sketch).

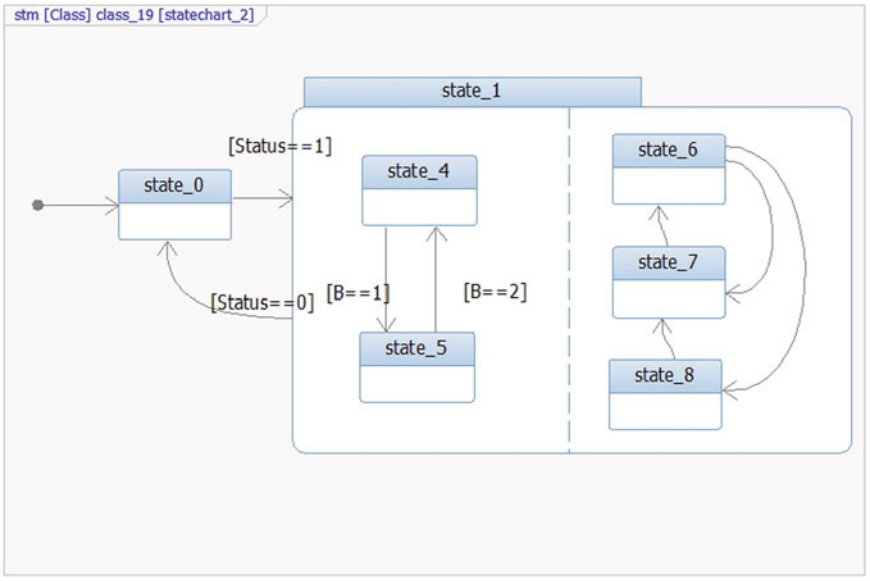


Fig. 6.2 Example of state machine diagram applied to the functional analysis

It was previously mentioned that the *State Machine Diagrams* are capable of mutually interacting, when the SysML blocks and parts to which they are referred are connected within an *Internal Block Diagram*. These connections allow the exchange of discrete *events* and continuous signals, evolving during the execution, by creating a simulation environment with clear boundaries, like in physical simulation. Therefore, the relation between the *State Machine Diagram* and the structural diagrams is stronger than it preliminary looks. Actually, the *Internal Block Diagram* and *Block Definition Diagram* are used in an extensive way within the Functional Analysis. They cover two important issues related to the functional architecture as the communication topology and the hierarchical structure of blocks. The *block* is the fundamental element of the SysML-based design. It is a modular unit and defines a collection of features to describe an element of interest (Weilkiens, 2008). The *blocks* may include a description of behavior, *attributes*, *operations* and structural features, like *ports*, *interfaces* and *connectors* indicating the hierarchy of elements. Also important is the *part*. A *part* is the contextualization of the *block*, and is used to instantiate it into a defined operational background, to implement a specific behavioral issue. The part inherits the main features of block and allows creating some specific characteristics, for the intended use of block into a scenario. The SysML structural diagrams use extensively both blocks and parts. The *Internal Block Diagram* is widely used to define the network established among some selected entities and enables the connection of several *State Machine Diagrams*. It is composed by *parts*, being eventually characterized by some dedicated behaviors. They are interconnected by means of *ports* and *interfaces*, and are

always referred to a context, a use case, or a block. The boundaries of this diagram identify the limits of the representation and state precisely the context in which the network is developed (e.g. a function, a use case, a scenario). *Ports* and *connectors* are usually different and their selection depends on the relation to be instantiated. They are used to carry *continuous* signals among the parts, within the context, and across the diagram boundaries. One important duty of the *Internal Block Diagram* within the Functional Analysis concerns the deployment of a connected functional environment, where all messages and communication needs, defined through the behavior diagrams, can be managed. Particularly, the identification of the functional *interfaces* is a critical issue, since it allows to detect the interactions and exchange of *operations*. As this exchange is usually *discrete*, the *interfaces* are used together with a list of required and provided operations, established in a *contract*, characterizing that *interface* to make it unique. The links drawn in the *Internal Block Diagram* are very important to drive the formal simulation. In Fig. 6.3 a typical Internal Block Diagram where the aforementioned elements appear is shown. In the example, the diagram contains three parts, interconnected by ports and interfaces.

The *Block Definition Diagram* is focused on the structure of blocks, within a functional scenario, like in a functional tree. The relations between blocks and parts can be understood looking at the whole diagram (Fig. 6.4). Blocks are related to other ones by some *connectors* to instantiate a sort of father-son relationship and create a multilayered structure. The block defined as a son of another one by a

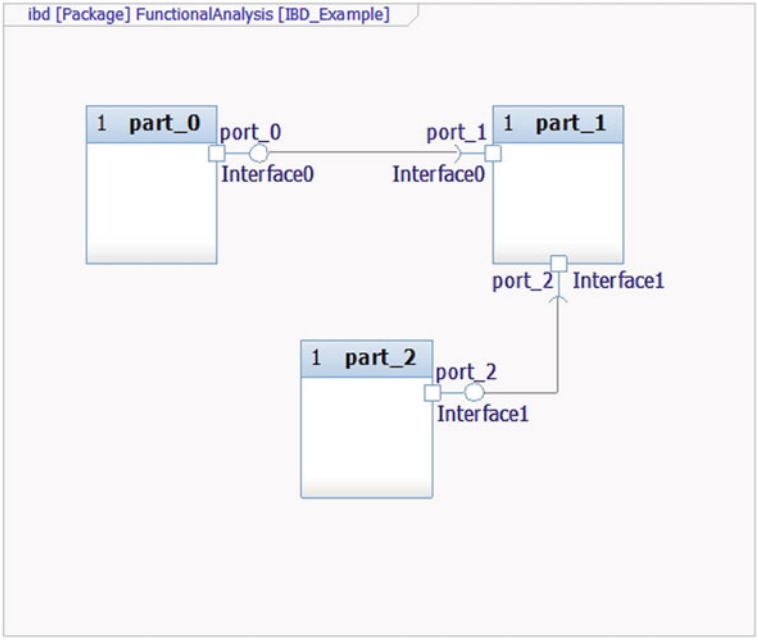


Fig. 6.3 Elements of an internal block diagram within the functional analysis

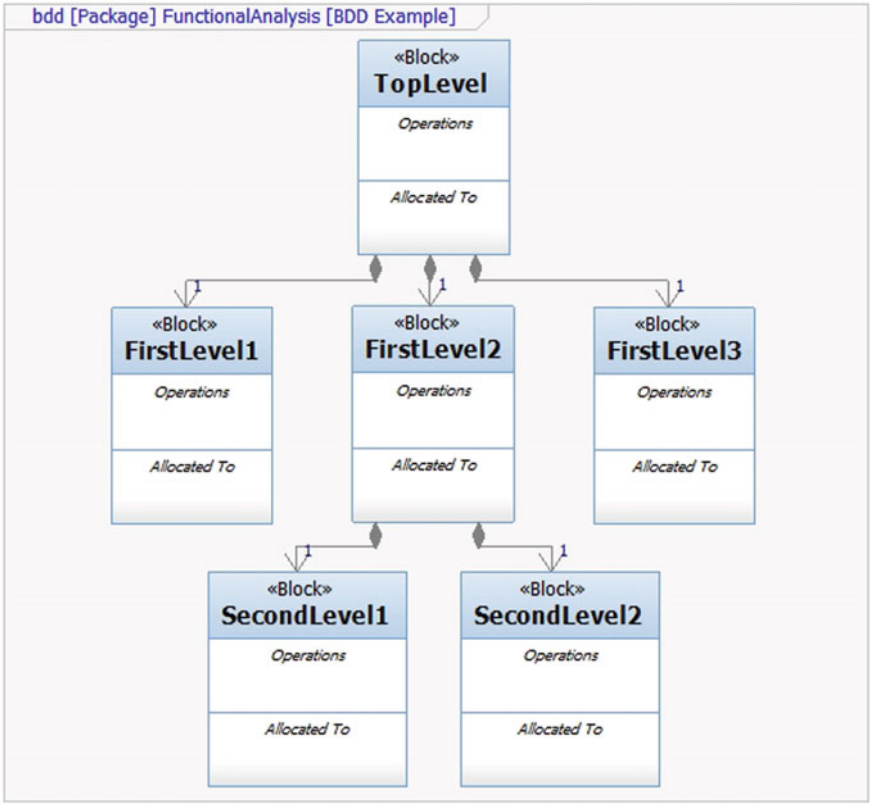


Fig. 6.4 Elements of a typical block definition diagram within the functional analysis

directed composition dependency looks like a *part* of the father block. This reflects the general definition of part. The son block is used within the specific and limited context of the father block. Therefore, the *Block Definition Diagrams* have a significant impact on the communication network, since they establish the context where the parts will be instantiated. Concerning the Functional Analysis, this aspect is mainly used to support the identification of scenarios, of their boundaries, when implementing some networks.

A typical use of the *Block Definition Diagram* in this context is the deployment of a *functional tree* to collect the *operations* and to define the *functions* which shall contain them. This is an alternative representation of the *functional breakdown* of the system, as it shall be shown in the test cases.

The set of diagrams here presented is a basis for the complete representation of the Functional Analysis phase, through the SysML. The meaning of diagrams can be different phase by phase, although their practical implementation is similar. In Chap. 7 some typical concepts of the Logical Analysis will be introduced. The differences with the Functional Analysis will be even remarked.

6.4 Requirements Derivation, Traceability and Allocation

The connection with the requirements specification is quite tight, within the Functional Analysis. The identification of system functions, directly derives from the Operational Analysis, as a consequence of stakeholders' expectations, and the functional requirements are collected, during the early steps of the design process. When the functional architecture is developed, the information about functional interfaces and functional behavior increase and new requirements are added. The requirements specification is a "living document", especially during the early phases of the process, and some precaution shall be applied to trace the relations among requirements and system operations and to track their evolution. So far, a good MBSE project should be an interconnected project, i.e. its parts are fully defined, only when a network of relations exists. This is one of the most powerful advantages of the MBSE approach. The dominant idea is that data are not only a stand-alone content of the *artifacts* involved, but also a set of relations shared with other *artifacts*, at the same or different level of analysis.

A very important aspect of the design process is the *traceability* of requirements, especially in presence of a complex system. In the context of the Functional Analysis, the *traceability* assures a complete identification of the relations connecting requirements and functional elements, in terms of *derivation*, evolution and *coverage* since their early stage definition.

The requirements definition process affects all the analysis levels and is subjected to a continuous updating. This makes the traceability very useful for the product development. It is remarkable the distinction between traceability inside and outside the requirement specification.

The traceability *internal* to the requirement specification involves their evolution and their mutual links in terms of *derivation* (relation between father and son requirement) or even a simple involvement (reference to a similar topic, influence on a related issue). This is usually managed inside the specification and it is exploited as a map to explore how it has been organized and managed. The traceability *external* to the specification deals with the *impact analysis*, i.e. it identifies the element of the functional architecture related to a selected requirement.

Besides the requirements traceability, the management of the design process is an important task. It proceeds with the instantiation of relations between functional elements. It sets up the *dependability*, being the set of relations assigned to the model elements to *trace* their interrelationships. These relations depend on the methodology applied for designing. One of the most important *dependencies* is the so-called *allocation*, which is used to assure the consistency between design phases, especially when considering the handoff between the Functional and the Logical Analysis. The *allocation* is interpreted as a formal transfer, between two subsequent steps of analysis, of one or more element of previous phase to another one of the following phase, which can carry out its duties. A typical allocation involves the transfer of system functions onto system elements.

Within a sequential approach like the MBSE supports, the Functional Analysis is essential for traceability and dependency, since it allocates the outputs of the requirements specification onto the logical elements of the Logical Analysis. Moreover, it includes even all the internal dependencies among functional elements.

This organization implements the fundamentals of the SE approach and assures a continuous stream of connected resources along the product development. The MBSE tools allow representing the dependencies in most of the SysML diagrams. Nevertheless, to visualize the requirements traceability, a good practice is resorting the *Requirements Diagram*. It may contain the whole set of relations among requirements (derivation and structure of the specification), among functional elements (hierarchy, connectors) and between the specification and the functional architecture (trace and other dependencies). Therefore, it makes possible collecting the entire set of links within the boundaries of a single diagram (Fig. 6.5).

Some traced dependencies are shown among requirements and blocks. This is a soft dependency, but is used extensively within the Operational and Functional

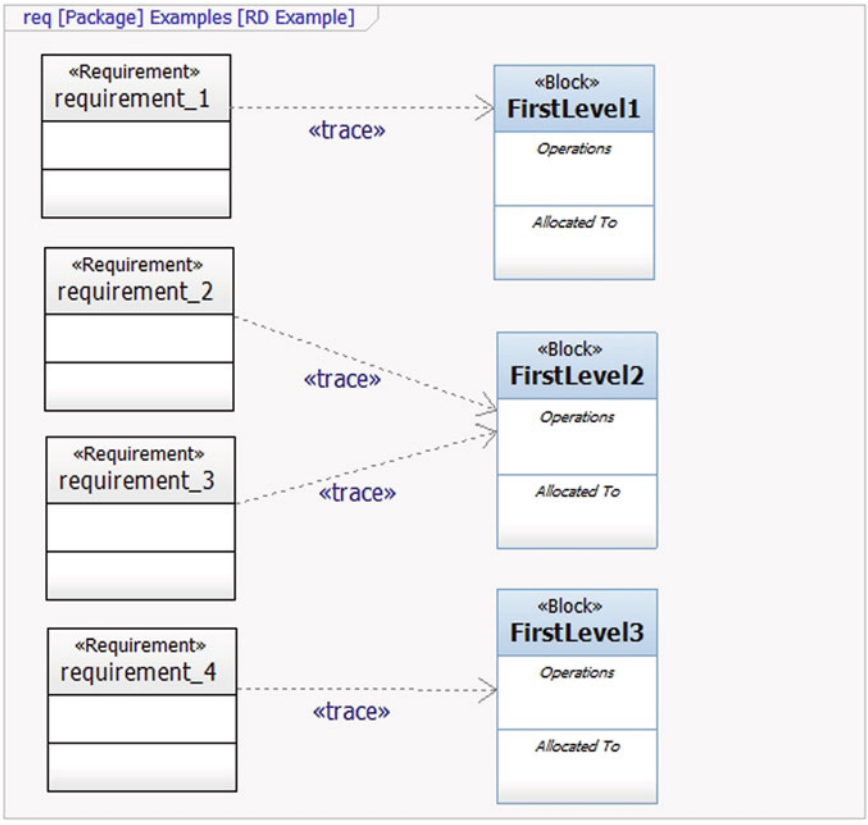


Fig. 6.5 Elements of a typical Requirements Diagram within the functional analysis

	FirstLevel1	FirstLevel2	FirstLevel3
requirement_1	FirstLevel1		
requirement_2		FirstLevel2	
requirement_3		FirstLevel2	
requirement_4			FirstLevel3

Fig. 6.6 Elements of a matrix view for requirements traceability

Analysis to express some general-purpose links between model elements, to check the requirements coverage, to perform an impact analysis, and to verify the consistency of data.

Since this set might be huge, the visual representation may be confused. To solve this problem, the MBSE tools provide even matrix and table views, having a friendlier outlook (Fig. 6.6).

It is important to point out that the information are always the same, independently on the representation preferred. The dependencies instantiated during the design process, by the user and in different moments, are always stored by the MBSE tool without resorting to a specific view. Matrices and tables allow summarizing the requirements for a coverage verification, but diagrams give a global impression to the user of relations and requirements defined. A similar exigency occurs when the Logical Analysis is performed, as is shown in next Chap. 7.

6.5 Results and Outputs for the Logical Analysis

Some typical results of the Functional Analysis concern the so-called realization of the use cases, the derivation of system functions and the definition of the functional behavior. Those activities include several tasks to be considered by the designer.

The use cases shall be characterized through a high-level description of the interactions among the system and the external entities involved. The related output is a set of behavior diagrams, like the *Activity Diagrams*, *Sequence Diagram* and *State Machine Diagrams*, together with a possible introduction of a high-level network of communication, implemented by one or more *Internal Block Diagrams*.

The derivation of system *functions* is related to the identification of *actions* and *operations* characterizing the functional behavior, and is completed by the identification of *states* representing the system modes of operation. When a classical *functional breakdown* is required to group operations, some *Block Definition Diagrams* can be exploited. The expected outcome is a complete and consistent functional architecture, made by a collection of *actions*, *operations* and *states* along with the *messages* and *interfaces* network.

A formal simulation can be performed to assess the functional behavior of the system, under specific scenarios and use cases, producing an executable version of the *State Machine Diagrams*. Furthermore, the functional architecture shall be

enriched with requirements *traceability* links and shall be organized using *dependencies*, where applicable, to keep consistency.

These activities shall be performed also considering what is necessary to perform the next phase, notably the *Logical Analysis*. The functional architecture is its main input, because it is focused on the design of a system structure able to meet the functional behavior and to perform the deduced functions.

6.6 Implementation: Deriving the Functional Architecture

The Operational Analysis provided some interesting results concerning the test cases. It is now necessary understanding how the user should move to perform the Functional Analysis. Some inputs are obtained from the previous stage, as is herein briefly resumed.

- The main actors exploiting some interfaces with the system have been identified, together with their objective (i.e. the use cases).
- Some operational scenarios have been defined, by instantiating relations among actors and use cases.
- The operations have been derived within those scenarios, and new requirements have been determined.
- The traceability among model elements is guaranteed by the instantiation of dependency links. The requirements specification has been updated, by considering the results of the analysis (backward synchronization from model to database).

It is now possible to analyze the use cases in detail, by defining the system behavior, strictly required to finalize each objective requested by the actors. It might be said that the focus shifts from an actor-oriented to a system-oriented view. This means that new actions, operations and states for the system can be defined conformingly with the data already available from the Operational Analysis.

Several SysML behavior diagrams will be defined to derive these new features and, notably, the Activity Diagrams (ADs), Sequence Diagrams (different from those of the Operational Analysis) and State Machine Diagrams (SMDs). The functional tree will be also implemented through some Block Definition Diagrams (BDDs) and the interface network shall be instantiated, through some Internal Block Diagrams (IBDs). In this way, the functional behavior and the functional tree will constitute the functional architecture of the system.

Requirements will be furthermore derived from some new model elements and their specification will be updated consistently. A final verification through the diagram animation will be also provided. Since the Functional Analysis is the core of the MBSE approach, it is performed in different ways, depending on the applied methodology and process. Some alternative solutions to organize the workflow will be discussed and the main common points and differences will be highlighted.

Particularly, considering common points, it shall be clearly stated that, apart from the workflow applied, the Functional Analysis is usually characterized by:

- A *use-case-based* analysis. This means that the use case is the main instance around which the model-based design is performed, being the focal point for properties of model elements and a node for traceability links.
- A *black box* approach, since the behavior of the different elements is not allocated onto logical or physical entities.

6.6.1 Didactic Test Case

Once the Operational Analysis is completed, a next step consists in defining the abstract functions of the system, within the Functional Analysis. The Functional Analysis models the functions just required to perform the system mission, by respecting some environmental constraints.

The *Functional Breakdown Analysis* lists the functions represented by a *Block Definition Diagram* (BDD). In the Functional Analysis, the *Block Definition Diagram* is used to describe the system sub-functions and how they are related to those of the main system, defining their hierarchy.

It looks important showing here the expected behavior of the system in terms of abstract functions, while describing how functions are implemented will be done later. The functions are modeled using blocks and the BDD shows the system decomposition in subsystems, representing either a function or a physical component, to be considered by the functional or physical analysis, respectively.

The Functional Analysis provides a first description of the main system function, within the selected context as a block, and its decomposition in subsystems, including other functions. These must be implemented to assure that the main function could be completely performed.

In the didactic test case, the Functional Analysis starts by describing the Functional Architecture depicted in Fig. 6.7. It depicts the decomposition of the main Laying Head system function into six sub-functions, as follows.

- Control Function—to control the balance, stability and dynamics of the system and to stop the system in case of emergency;
- Measuring Function—to measure the size of the rod and amount of material coiled and stored, but even the power consumption, the electric current and the spin speed;
- Shaping Function—to shape the wire rod into coils;
- Signaling Function—to display the system status and information to the operator;
- Monitoring Function—to monitor the system, e.g. the shaft position or the rotor vibration, or to detect the lack of power supplying;
- Rotating Function—to convert the translational motion of the rod into rotational motion.

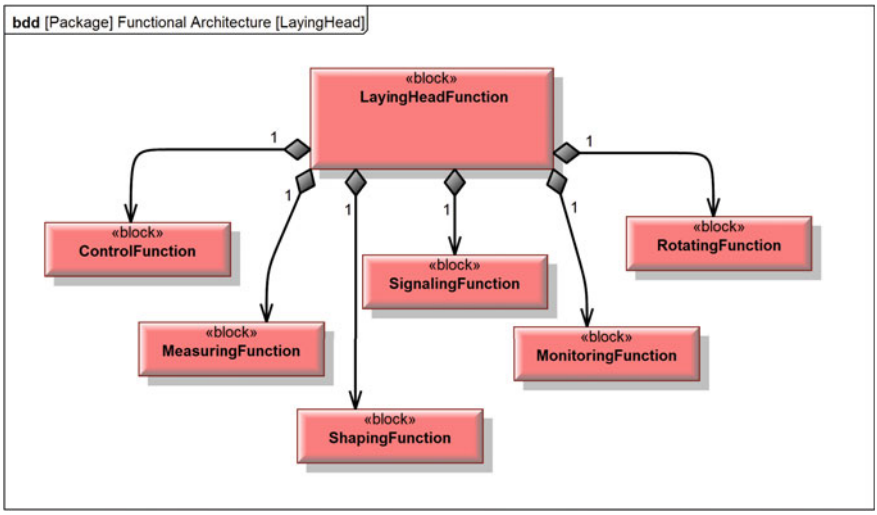


Fig. 6.7 Functional Architecture of the laying head system

The main block represents the Laying Head, while the other Function Blocks are connected to the main system, with a *composition* relationship. This association is represented by an arrow between the main block and its blocks and a single solid diamond. The number of entities per each connected block can be written just aside the arrow.

All the use cases defined within the Operational Analysis (Chap. 5) should be supported by these functions and, vice versa, every function should be used by the system. Starting from the use cases, the *Internal Block Diagrams* (IBD) are used to describe the system functions, enabling each use case. The difference between IBD and BDD is due to the possibility of linking some parts via connectors and ports, to describe how the functions are related to perform the use case.

Usually, the main system function is represented as an open box, in which some other subfunctions are inserted. The actors are represented outside the main function block and are directly connected to the subfunction blocks. Each subfunction is detailed in parts. A *part* is a specific function of the block and is described by the notation `part_name : part_block`.

In the example, the functions needed to perform the use cases are described using some Internal Block Diagrams. Only the use cases "Start the process", "Stop the process", and "Require suspension and rotation" are herein described, to simplify the description.

In Fig. 6.8 the use case "Start the process" is shown. According to the Use Case Diagram, the Operator and the Monitoring and Emergency System actors interact with the system, to start the steelmaking process and are outside the main function block "LayingHeadFunction". The system functions involved in this use case are the Measuring Function and the Signaling Function. The parts "Preliminary Check"

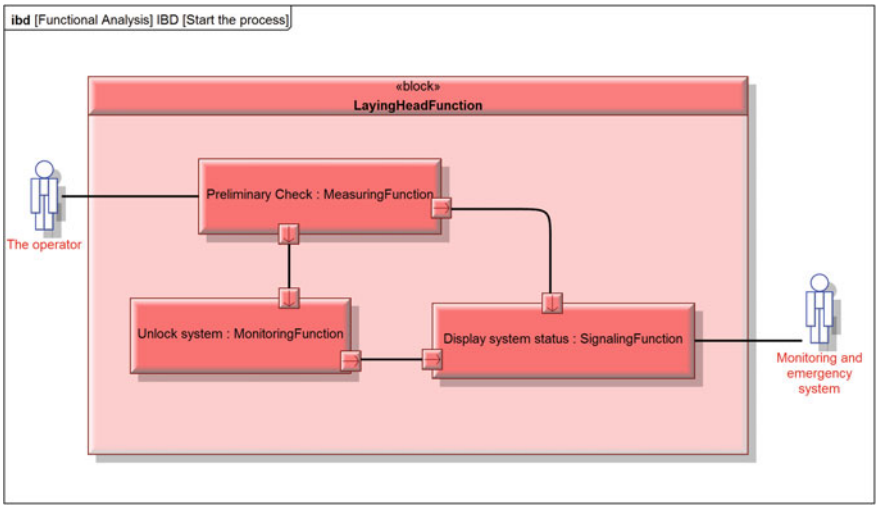


Fig. 6.8 IBD of the use case "Start the process"

and "Unlock system" of the Monitoring Function and "Display system status" of the Signaling Function are connected via connectors and Flow Ports. These Flow Ports basically show what can flow through the block, and in which direction, either in or out.

The process starts only after a preliminary check, aimed at verifying that the Laying Head can be safely unlocked and can be made rotating. Looking at the IBD, it can be easily read that to "Start the process" the system should be able to provide a preliminary check as soon as requested by The Operator. It sends some data to the Monitoring and Signaling functions, respectively, to unlock the system and display the system status. The Monitoring function sends even a message to the Signaling function. The function of displaying the system status is then used by the Monitoring and emergency system.

Similarly, Fig. 6.9 describes the use case "Stop the process". The process can be stopped either by The Operator or by the Monitoring and emergency system. In regular operation, the process is stopped when the wire rod length reaches a required value, while in case of failure, an emergency stop is launched. Therefore, the IBD shows that the Monitoring and Emergency system interacts with the Control function to request an emergency stop and with the Measuring function to monitor the length of the wire rod. The Operator interacts only with the Signaling function, through the displayed information about the system status. The Signaling function receives information on the requested stop from the Control function, in case of either emergency or normal operation.

The "Require suspension and rotation" use case is described in Fig. 6.10. To enable the rotation of system, some measurements and controls need to be done in advance. The Monitoring and Emergency system requires the measurement of

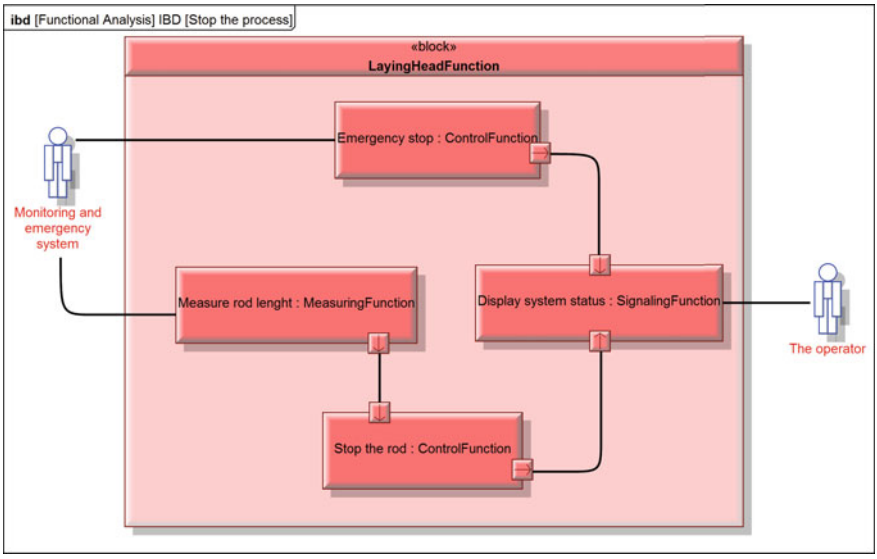


Fig. 6.9 IBD of the use case "Stop the process"

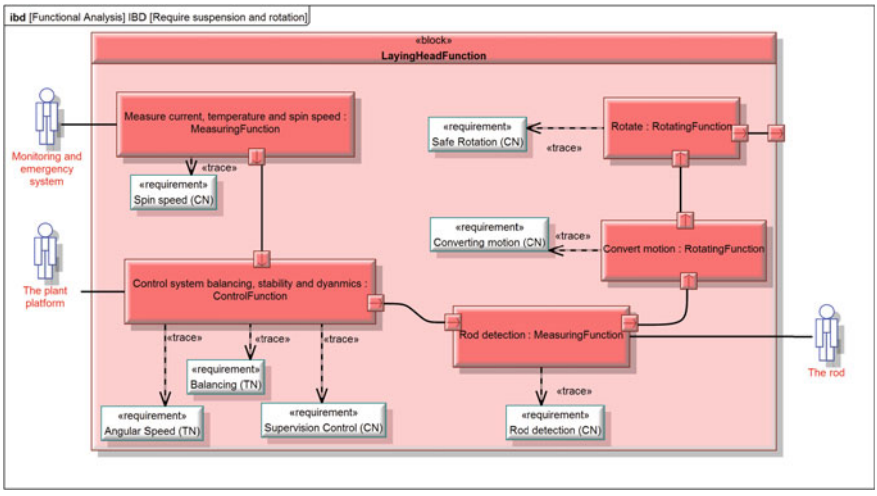


Fig. 6.10 IBD of the use case "Require suspension and rotation"

current, temperature and spin speed of the Laying Head system, being controlled by the Control function, as well as of unbalance, stability and dynamics. The Measuring function detects the position of The Rod and then the rotation is performed by the Rotating function, enabling the conversion of translational motion of wire rod into rotation.

It is significant that the IBD describes also how requirements are traced at this stage of the Functional Analysis. The dependency is here indicated with the notation ≪trace≫, which is a weak relationship, but is useful to allocate requirements to the system functions.

The BDD and IBD describe the system functions and their interconnections in each use case scenario, but the system behavior is described by the State Machine Diagram and Activity Diagram, being focused on system states and activities, respectively. A Sequence Diagram can be also used, although it also includes all the actors. Two among those three diagrams are usually enough to completely describe the system behavior, and their selection is up to user. In this test case, the State Machine and Activity Diagrams are preferred.

The State Machine Diagram (SMD) describes the dynamic transition between system states, which is activated by some event or condition, to enable its execution. In Fig. 6.11 the SMD of "Require suspension and rotation" is shown. Its main elements are herein listed.

- The *initial pseudostate* is described by a solid circle, that specifies the default state of system at the beginning.
- The *state* is a rounded rectangle, representing the system status in operation.
- The *transition* is described as an arrow, linking two states, and is defined by triggers, guards and effects.
- The *final pseudostate*, is once again a circle, in which the operation is completed.

Change of states is effected by transitions defined by trigger events, guard conditions and effects with the notation: `<trigger>[<guard condition>]/<effect>`.

A *trigger* indicates an event that can cause a transition to the target state. A *guard* is evaluated to test whether the transition is valid or not, and the *effect* is a behavior executed once the transition is triggered.

A simple state is represented as *Atomic State*, while the *Sequential State* contains a group of sub-states, which can be linked to an atomic state or another sequential state. Within a sequential state, the initial state must be even defined.

In the example, the rotor must be suspended at standstill. Before starting the rotation, the rotor needs to be centered and balanced, and a calibration of suspension is performed. It is usually operated above its critical speeds, but always below the instability threshold, to avoid a dangerous increasing of the whirls amplitude. Therefore, the system is operated within a defined range of spin speed. The wire rod will be suspended and rotated by the system, only after that the rotor reaches the condition of stable rotation in the so-called supercritical regime, assuring the best self-centering.

At the beginning, the system is at standstill and, as soon as the condition "Levitate" results valid, it enters in the sequential state "Rotor lift off". The rotor shaft is now suspended, and the state describes the calibration operated to center the rotational axis within the bearings, as soon as it is centered, the "Rotor calibrated" state is reached. This allows the system exiting the sequential state. It is then

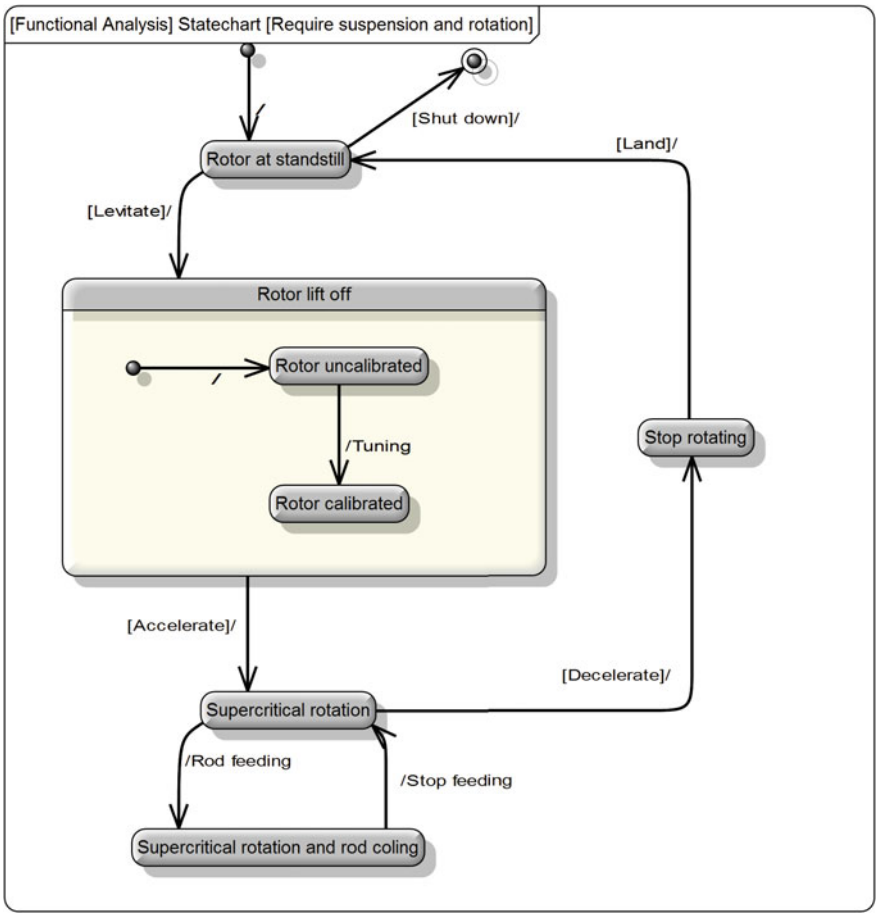


Fig. 6.11 STD of the use case "Require suspension and rotation"

accelerated, to reach a defined spin and the state "Supercritical rotation". The system can transit to the "Supercritical rotation and rod coiling", when the rod feeding starts. When a certain length of the wire rod is measured or an emergency occurs, the rod feeding is stopped and the system comes back to the "Supercritical rotation" state. As soon as the system receives the command "Decelerate", it decelerates and finally it stops rotating, entering the "Stop rotating" state. The rotor shaft is still levitating, therefore, to stop completely, it requires that condition "Land" is activated, to go back to a standstill position, which corresponds to the first atomic state.

It can be understood from this example, that the system behavior is described by one behavioral diagram for each use case, but it is even possible resorting to a single state machine or activity diagram, which includes all the use cases, simultaneously.

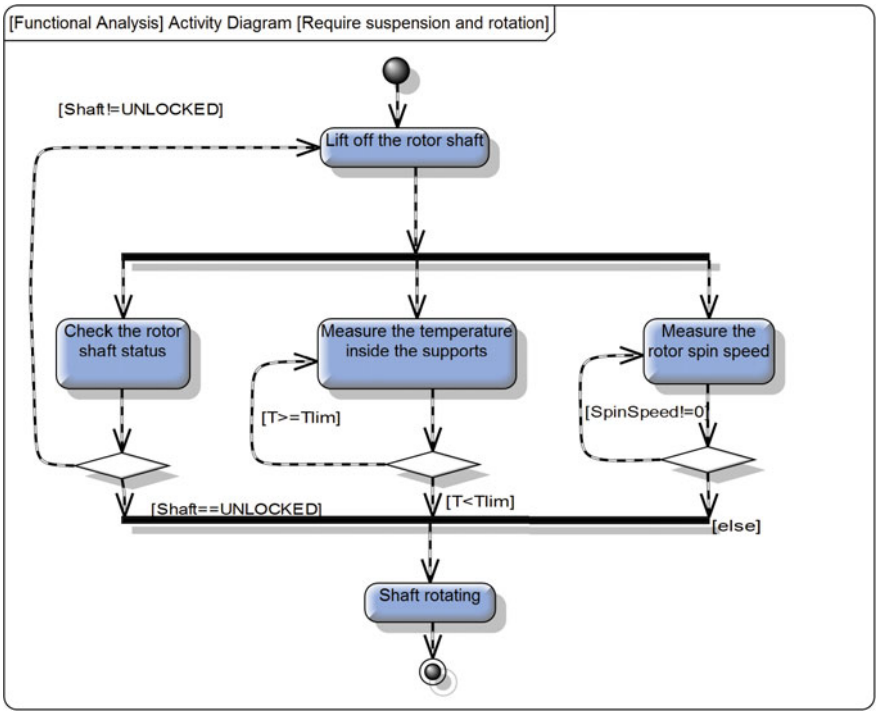


Fig. 6.12 AD of the use case "Require suspension and rotation"

This case has been modeled for every use case, using a State Machine Diagram and an Activity Diagram.

In Fig. 6.12 the AD of "Require suspension and rotation" is depicted. Its elements are herein defined.

- The *initial node* is a solid circle, where the system's activity begins.
- The *atomic action* is a rounded rectangle, showing the action performed by the system in operation.
- The *transition*, looks like an arrow, connecting two activities to specify the flow of actions, and related inputs and outputs.
- The *activity final* is a circle, in which the activity ends, being completed.

According to the figure, the system lifts off the rotor shaft, and then three actions are executed contemporarily, as the fork node indicates. The rotor shaft status is checked and the temperature inside the bearings and the rotor spin speed are measured. Each action may have two possible outputs but, as the joining node indicates, only when all those three actions give the required output, the system can execute a next action. Otherwise, actions previously described will be repeated till the output changes. To implement that strategy, a decision node with one input and two outputs is placed immediately after these actions. A loop is foreseen, since the

system can come back to the actions to be repeated. The outgoing flows are defined by a guard condition, stating the threshold output value to proceed.

In this case, the laying head can rotate the rotor shaft, only when the guards `[Shaft==UNLOCKED]`, `[T<Tlim]` are true and `[SpinSpeed!=0]` is false. In other words, the system can safely execute the rotation, only if the rotor shaft is unlocked, the temperature `T` inside the bearings is lower than a limit `Tlim`, and its spin speed is still null. Otherwise, a rotation might detect an anomalous operation, potentially dangerous, since some power is applied to the rotor shaft by another system, not by the dedicated motor. The activity can be considered completed as soon as the system is rotating the rotor shaft.

6.6.2 Industrial Test Case

In this case a use case based analysis is implemented. Both the MBSE technique and of the most complete set of SysML diagrams are exploited to characterize the system behavior. The Activity (ADs), Sequence (SDs) and State Machine Diagrams (SMDs) as behavior diagrams, and the Internal Block (IBDs) and Block Definition Diagrams (BDDs) as structural diagrams are applied. The main diagrammatic element is the operation, which is defined through the behavior diagrams, since its first definition within an *Action* to its final instantiation within a *State*. The operation is directly allocated onto some logical block, which identifies a practical system element, or a *Product*, within the next phase. This means that the system is directly allocated on a possible logical/physical implementation to define the Product Breakdown Structure (PBS), after the Functional Analysis.

This approach is characterized by a high level of granularity, i.e. the depth of details is quite good, and this is expressively because the operation is the key element and is focused on some very specific aspects of the system behavior. Nevertheless, a dedicated functional breakdown identified by the functional tree, through a BDD, is not always drawn. As a consequence, the level of analysis making possible to derive consistently the operation is depending on the designer. A special care shall be applied to avoid any mismatch between the details of different use cases.

A direct allocation of operations onto blocks, for a straight implementation, is a crucial issue. Real components often do not precisely fit the set of characteristics specified by the functional behavior, thus making difficult a very specific allocation. Two layers of implementation are often provided, exhibiting some important differences between the Logical and Physical Analyses. The logical one is an intermediate step, between the functional and the physical architecture. It is conceived to allow the identification of suitable elements and components, with some general features, at high level, of the final product, which will be described within the PBS. It is possible, for instance, introducing a logical component as an “electrical switch”, which is not yet related to the real component, as the Commercial Off The Shelf (COTS) elements, with very specific features. Only the physical component

shall be an "electrical switch", with some specific characteristics, like maximum voltage, current limits and customized production specifications, which allow relating this element to a commercial device with well-defined properties. This difference highlights the important role of the physical analysis, because it provides a real identification of the PBS, within the SysML model. This strategy is herein briefly described, for the IPS case study.

A first step is the implementation of some Activity Diagrams (ADs) for the use cases. Figure 6.13 shows the diagram related to the use case of ice prediction. The flow starts with a fork node, which introduces two parallel paths, because the prediction is performed upon air and ice data. A preliminary set of actions concerns the *acceptance event* and are placed at the beginning, since data may be missing and the related action may not be executed. A global validation of data is required and, after a consistency check, which leads to end in case of failure, the selection of prediction type is performed, depending on the available information. The forecast is then computed and some messages are sent, for monitoring purpose. The whole flow is contained within an interruptible region, which is instantly left in case of switch off signal.

A simpler representation is shown in Fig. 6.14, for the ice detection. As it can be seen, different actions are performed in parallel and ice measures are transmitted at the end of process, until that a switch-off signal is provided, and the interruptible region is disabled.

Figure 6.15 shows the AD for the ice removing. The different surfaces of the aircraft shall be equally protected from icing, as required by high level specification, but a sequence of protection is defined here, for the first time. When the actuators applied to a defined zone are activated, others are switched-off, to reduce both the power peak and the global consumption, as is demonstrated by the numerical simulation in Chap. 7. The AD for ice removing follows the same approach shown for ice prediction and detection, with a single interruptible region, which can be deactivated, when a switch-off command is received.

Let's look now at the control and monitoring use cases, responsible to manage the most of commands appearing in the ADs just shown. Figure 6.16 contains the AD for the use case related to system control.

Two operating modes are foreseen, namely the auto and manual mode. This is a relevant update to system behavior since for each scenario a separate set of actions is identified. The selection of de-icing intensity is automatically regulated, depending on the ice level, whilst, in the manual mode, an input from the pilot is necessary to regulate the system. This can be very useful in case of failures of the main control subsystem and for testing.

The Monitor AD is the most complex of the Functional Analysis. It includes many algorithms to manage the different messages coming from the other use cases. However, it is too large to be included directly, so a brief description is provided.

Three status of the system are hypothesized, notably "on", "stand-by" and "off". Each status offers a pre-determined set of functionalities for command and control, providing, at the same time, some important messages (visual/aural) to the actor (pilot). For instance, as far as the "off" branch is considered, commands are disabled

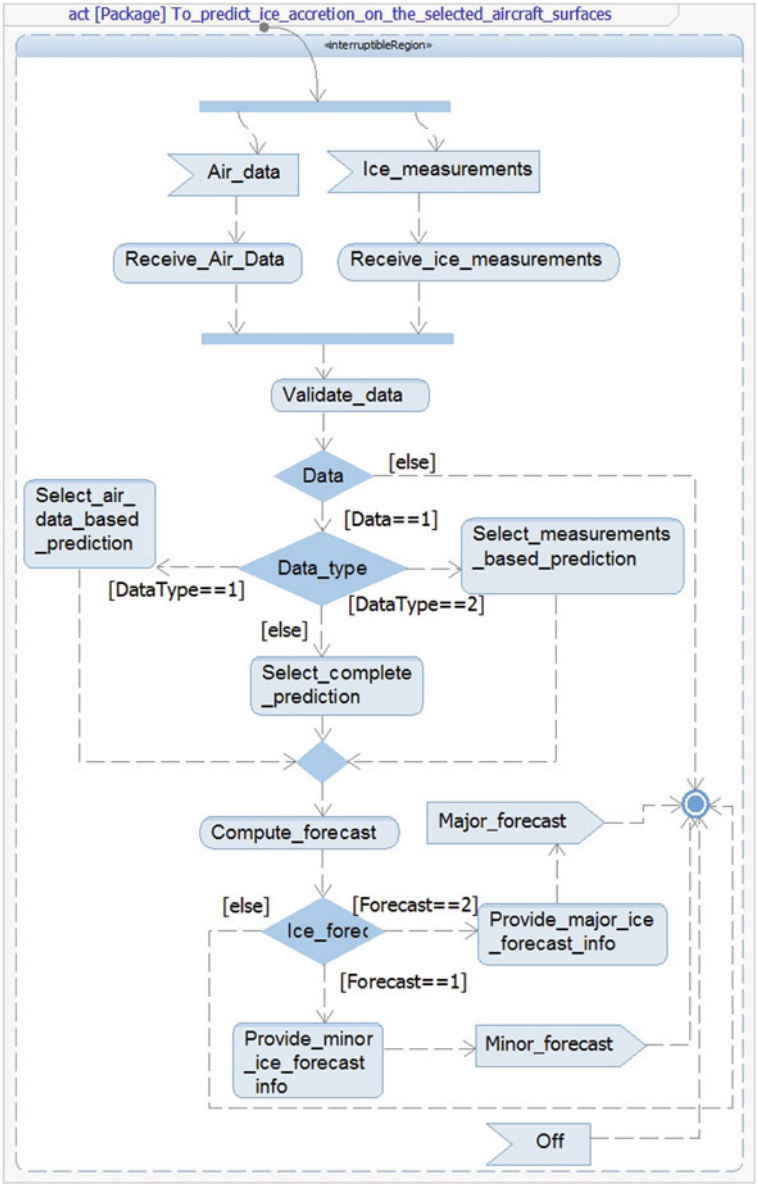


Fig. 6.13 AD for the use case related to ice prediction

and the flow reaches the activity final. By converse, the "standby" mode allows the system communicating some warnings about the ice level, to perform forecast and measurements, whilst the "on" mode identifies the full system behavior, in monitoring. This diagram is helpful to sketch the logical map of the system behavior.

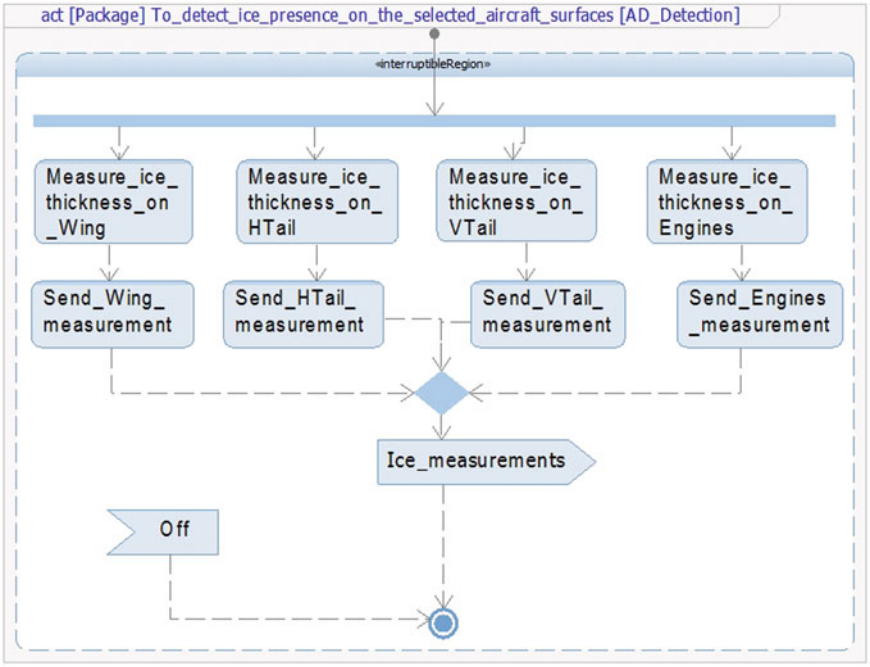


Fig. 6.14 AD for the ice detection use case

The Monitor use cases are the most difficult to be characterized, since they work under several triggers, provided by actors. They are the core of the whole model, since the behavior diagrams related to monitor are simulated, at the end of Functional Analysis. This activity validates the model and offers a preliminary overview on the working system, for given operating conditions.

Actions described by the ADs for the use cases are then instantiated as operations, in the related SDs as in Fig. 6.17 showing a partition of the SD related to ice prediction. If the SD of Fig. 6.17 is compared to the SD defined within the Operational Analysis for the same use case, it can be remarked that some operations are similar, but some details have been added here to specify, for example, how the forecast is computed and what kind of data are required. It happens the same for the other SDs. Is relevant understanding that the operations used within the Functional Analysis shall not be independent from those defined within the Operational Analysis. Traceability shall be assured between the two analyses, through an instantiation of dependency links among the operations. This allows an effective coverage of the information translated from the operational to the functional phase. This translation can be done by written matrices or drawn diagrams. As an example, a visual traceability is sketched in Fig. 6.18, for a set of operations related to ice detection. Some operations have been detailed in the Functional Analysis.

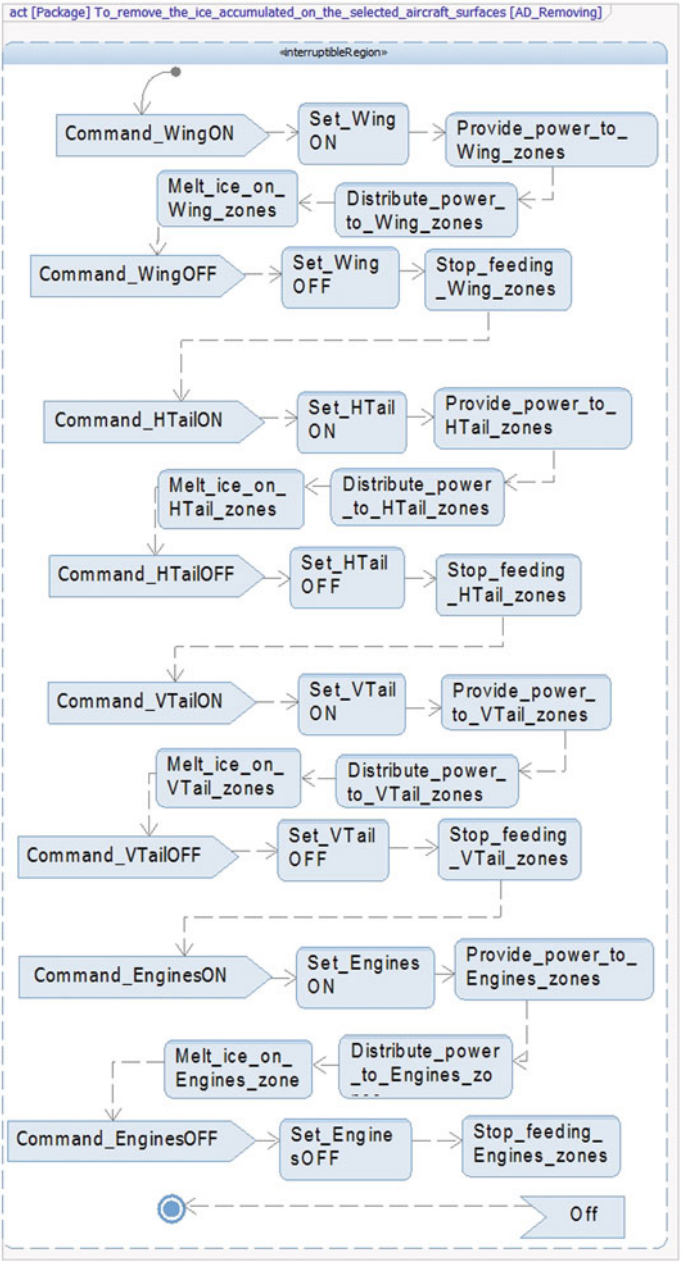


Fig. 6.15 AD for the ice removing use case

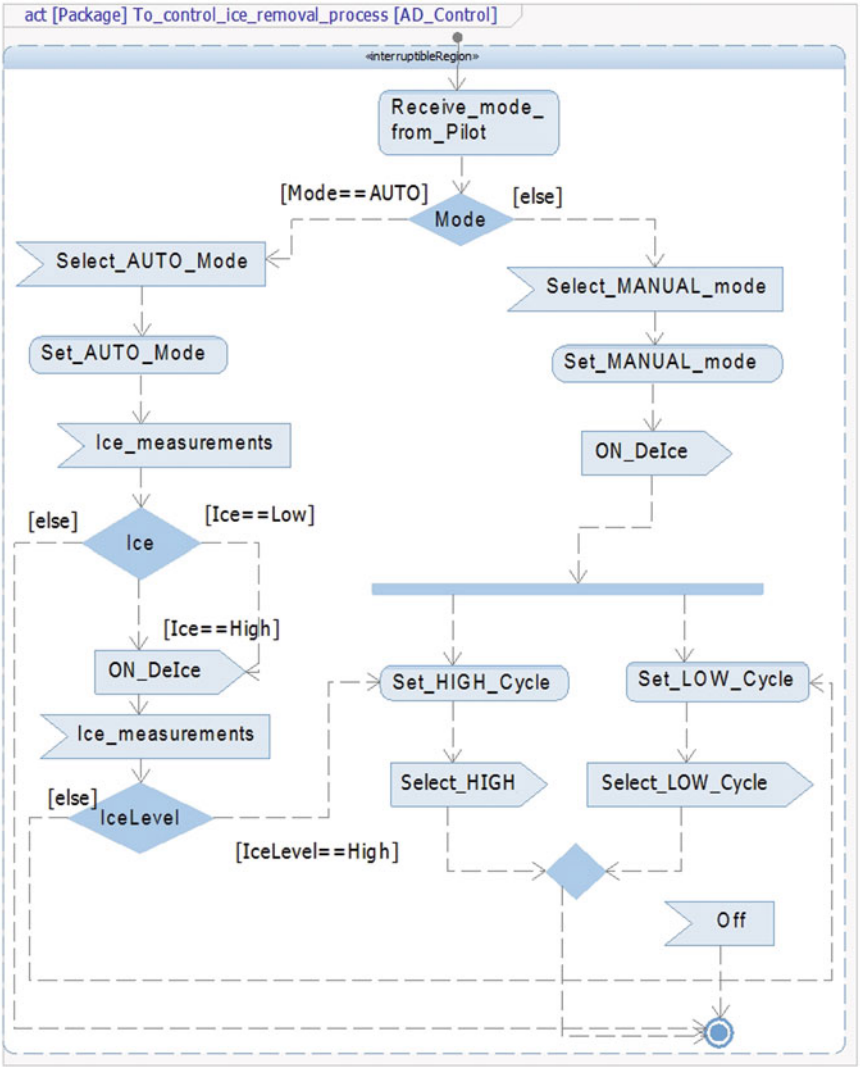


Fig. 6.16 AD for the use case associated with system control

Sometimes, it is possible re-using the operations previously defined, although it might be helpful re-defining them, within the right package, in the model tree.

The whole set of data coming from the first steps of the Functional Analysis is summarized by some behavior diagram, as the State Machine Diagram (SMD). It allows working with states, bigger elements than actions and operations. They represent some complex operational status of the system, while working, and include multiple tasks. Moreover, SMDs are usually adopted to simulate the system behavior, because they can be animated through some dedicated panels, to emulate the

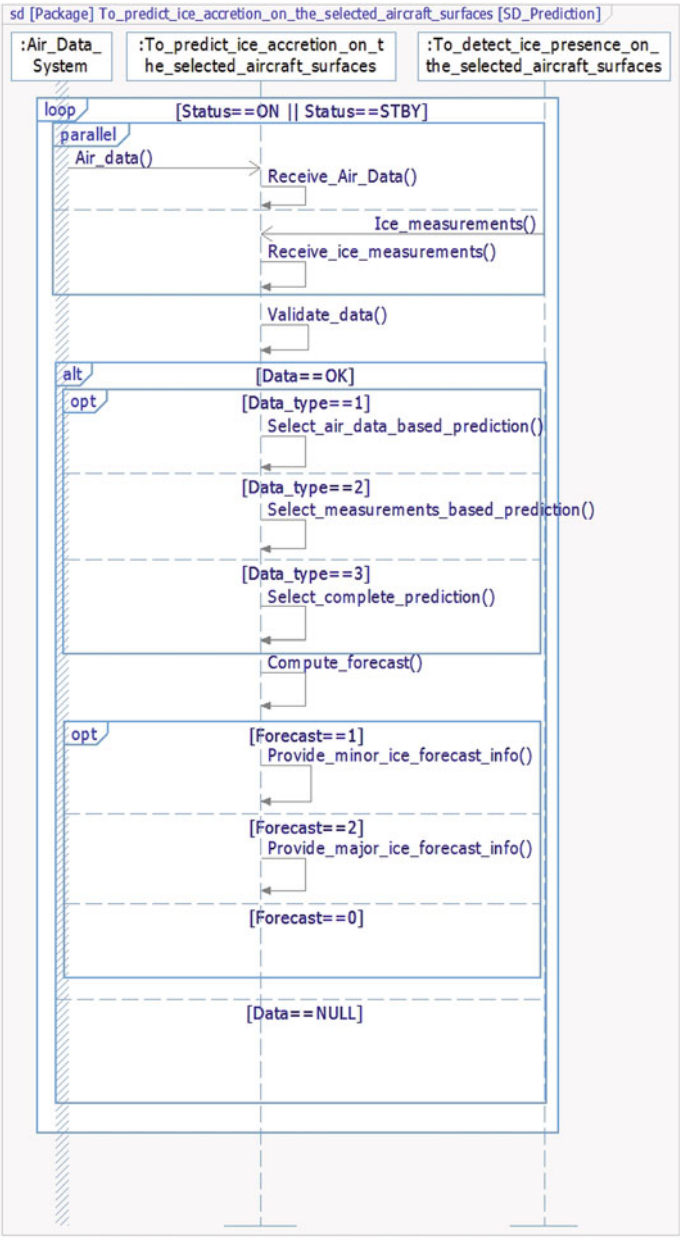


Fig. 6.17 SD for the ice prediction use case

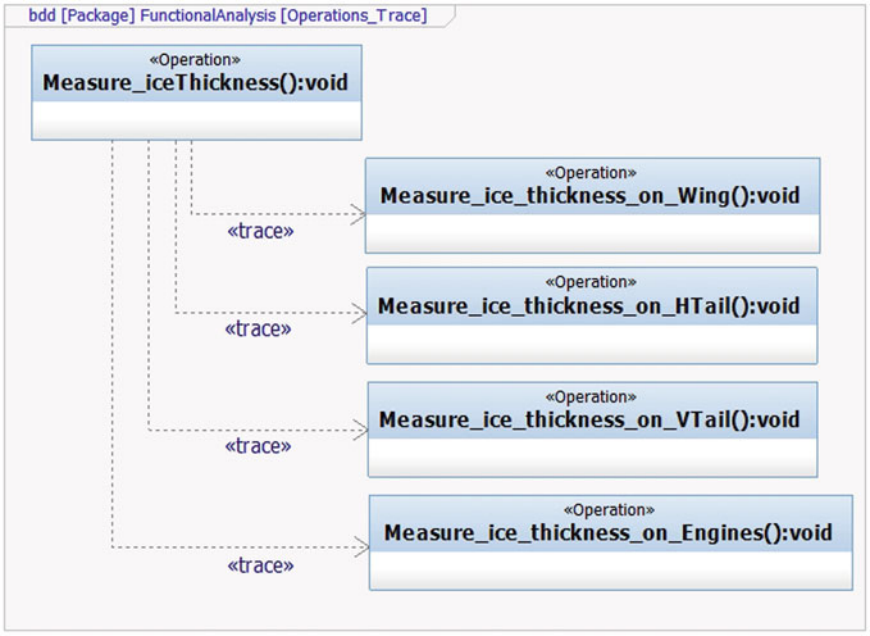


Fig. 6.18 SD for the ice prediction use case within the functional analysis

interaction with actors. SMDs of different use cases can be then integrated by the Internal Block Diagrams (IBDs), to perform a complete simulation of the behavior of the whole system. The SMD concerning the ice prediction is shown in Fig. 6.19.

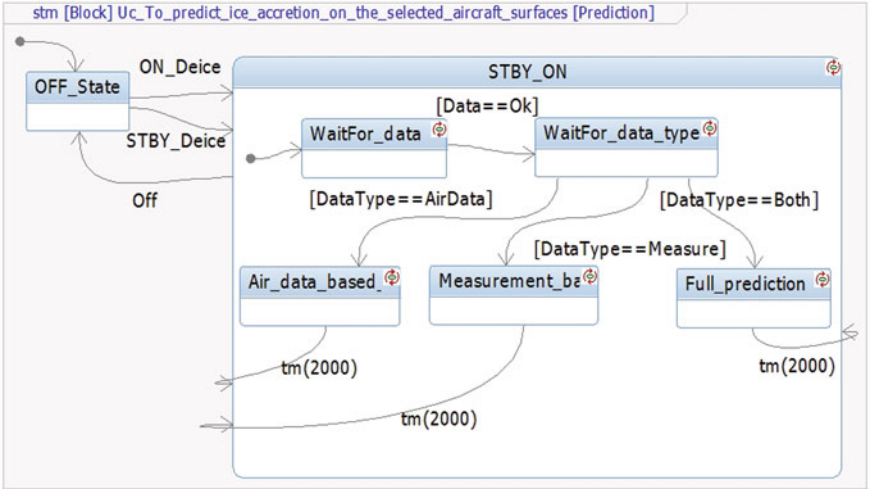


Fig. 6.19 SMD for the ice prediction use case

This SMD contains two main states. The first one identifies the "switch off" situation, when the system is not working and no tasks are performed. The "switch on" state is, instead, more complex, since it hosts other sub-states. Each one contains one or more operations to be executed. Transitions between states can be activated through guard conditions or triggers, which can either be related to some specific values that signals may assume in different situations, or to timing, as shown by the trigger "tm" (which stands for "time millisecond"). In this case, the "off" state is left, when the "on signal" is provided. The system enters the "standby-on" state, being the same for both operating modes, since some features shall be executed in both conditions. After some waiting states, the proper type of forecast is selected. The forecast is updated every two seconds, when the main state is executed again starting from the wait states. This is a typical structure of a SMD for this kind of application.

Some typical features of this type of diagram are herein listed:

- The identification of a main state, containing the different algorithms, and a "switch-off" starting state, from which it starts, when required, and where the system returns, after reaching a typical "end";
- the definition of dedicated "wait" states, very useful to initialize some variables or to enter the main state;
- the updating strategy, concerning the time required to update the whole diagram, depending on the part which is computed, to avoid the freezing of data, as it happens when the signal is unable to exit a state, during the execution.

Those three main issues can be recognized in the several SMDs here proposed. Figure 6.20, for instance, shows the SMD for the ice detection.

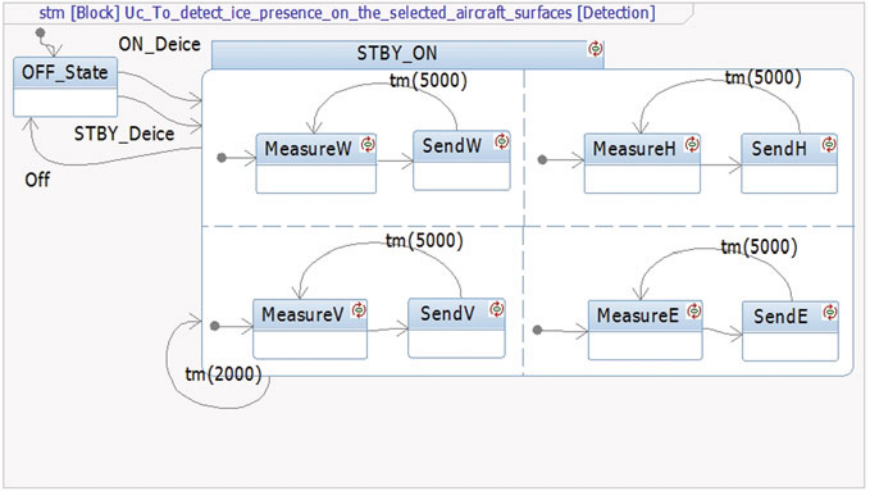


Fig. 6.20 SMD for the ice detection use case

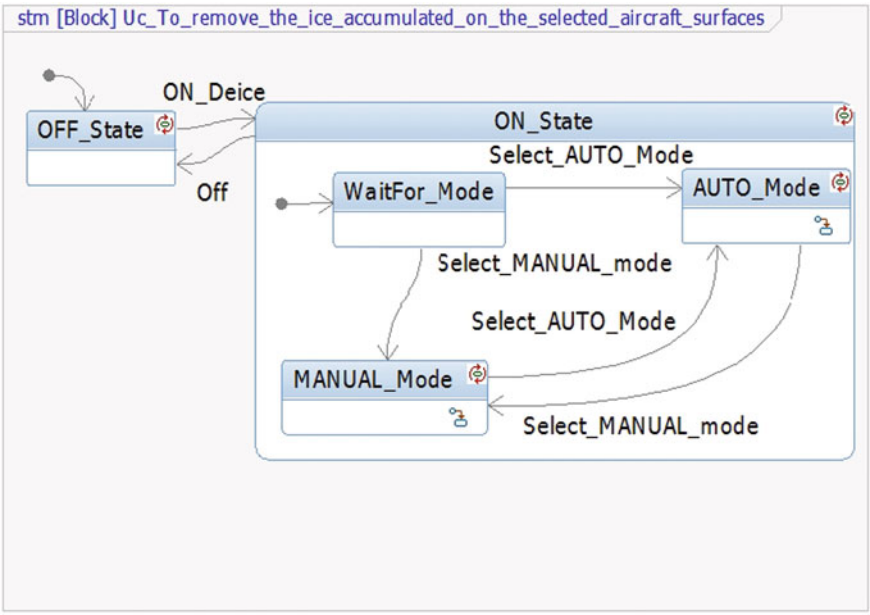


Fig. 6.21 Main SMD for the ice removing use case

As it is shown, a similar structure is maintained. Four main groups of states are here identified, within the main one. They are executed in parallel, since the detection of ice level is performed simultaneously on four zones of the aircraft.

In Fig. 6.21 the SMD for the ice removing use case is shown.

In this SMD is relevant that some states may be characterized by some other lower level SMDs, because the representation concerns no longer a static behavior, but a dynamic response of the system. The higher-level diagram presents a typical structure, as it was previously mentioned. However, since manual and auto modes are quite different, some low level SMDs have been built, to define them.

They are shown in Figs. 6.22 and 6.23 for the Manual and Auto mode, respectively.

The first diagram shows four main groups of states. They can be executed in parallel, depending on the input sent by the pilot. The second SMD performs a sort of loop, which is based on a suitable timing, i.e. a specific time interval between subsequent activations is automatically chosen, as a reaction to the current ice severity (CT, cycle time).

The control SMD is even similar to the previous one, as Fig. 6.24 shows.

If one observes the low level SMD, some differences between manual and auto modes controls are detected. The manual control is actuated by an input of the pilot, while the auto control is based on the measured data of ice level and icing severity, being compared to some guard values, to select the proper state to be reached by the

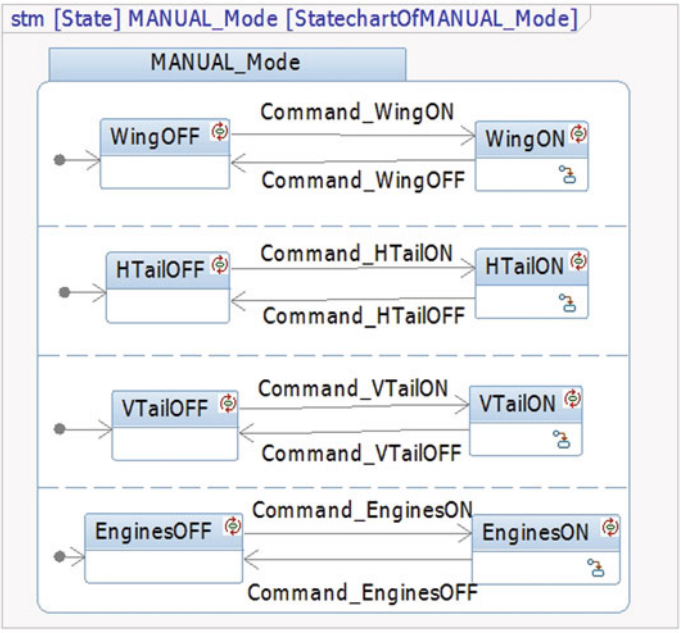


Fig. 6.22 SMD representing the Manual Mode state of the ice removing high level diagram

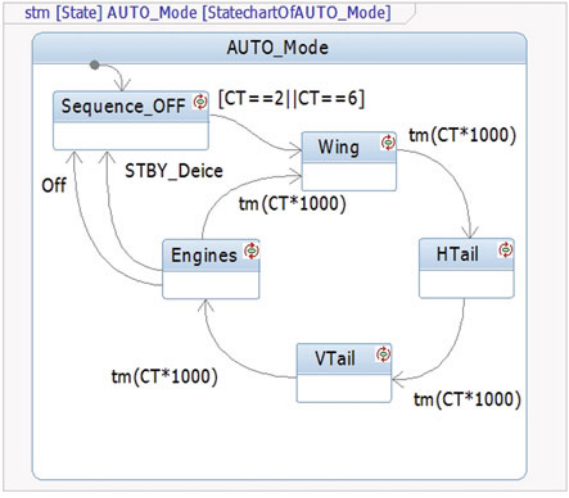


Fig. 6.23 SMD representing the Auto Mode state of the ice removing high level diagram

system. The target state is responsible of the selection of the cycle time, to be applied to regulate the strength of ice removing (Figs. 6.25, 6.26).

As for the AD, the monitor SMD is the most complex one, since it shall include the whole set of messages and inputs, provided by the pilot. The output related to

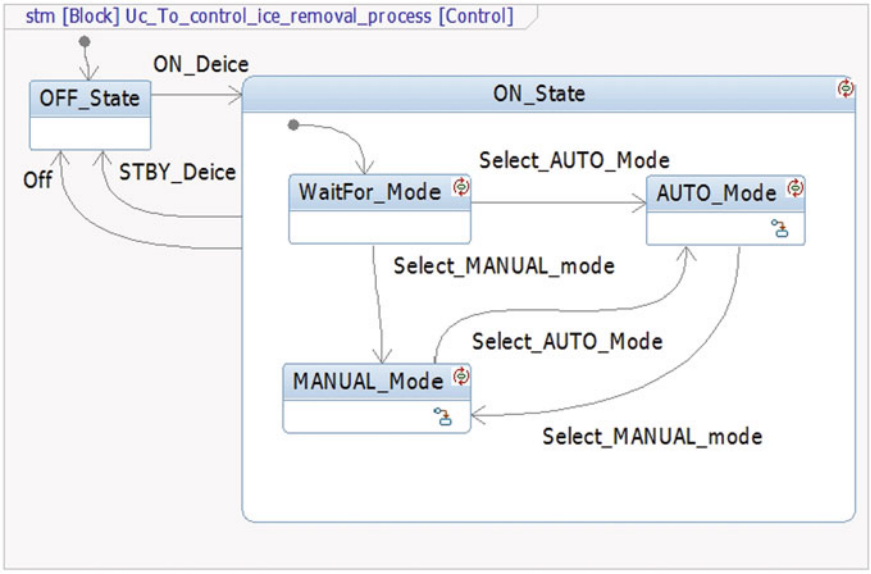


Fig. 6.24 Main SMD for the control use case

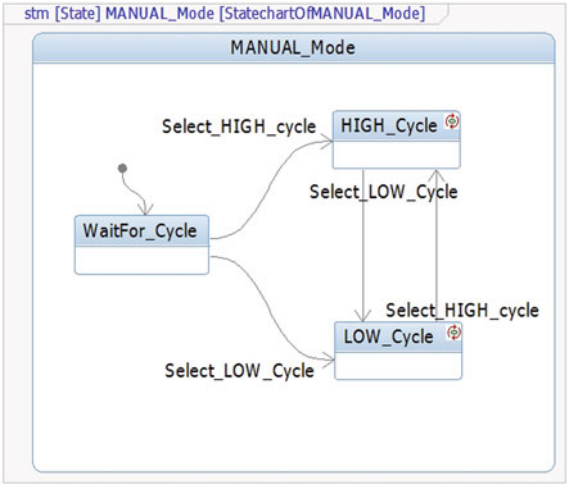


Fig. 6.25 SMD representing the Manual Mode state of the control high level diagram

ice severity, forecast and system modes and status shall be provided as well. The SMD is therefore full of elements. Some details are described in Figs. 6.27 and 6.28, separately.

Three main states can be recognized. The "Off" state simply shows the deactivation of the system. The "Standby" state is responsible to show the ice level and the forecast, as previously described by a dedicated AD. The "On" state contains

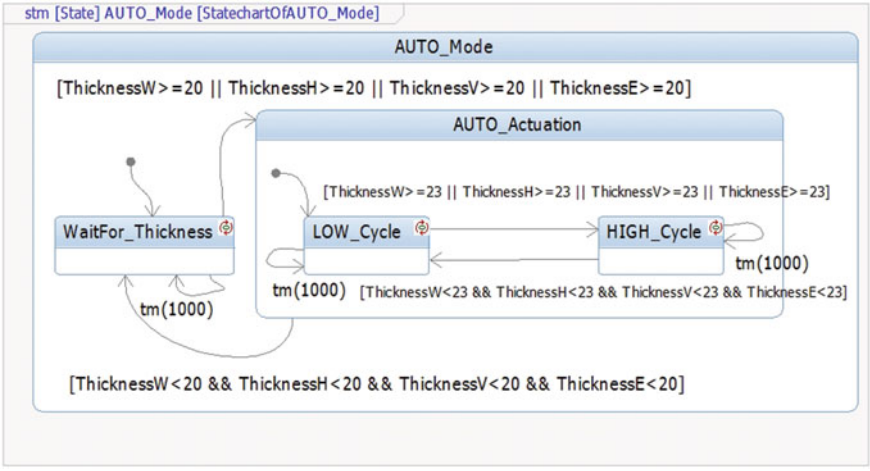


Fig. 6.26 SMD representing the Auto Mode state of the control high level diagram

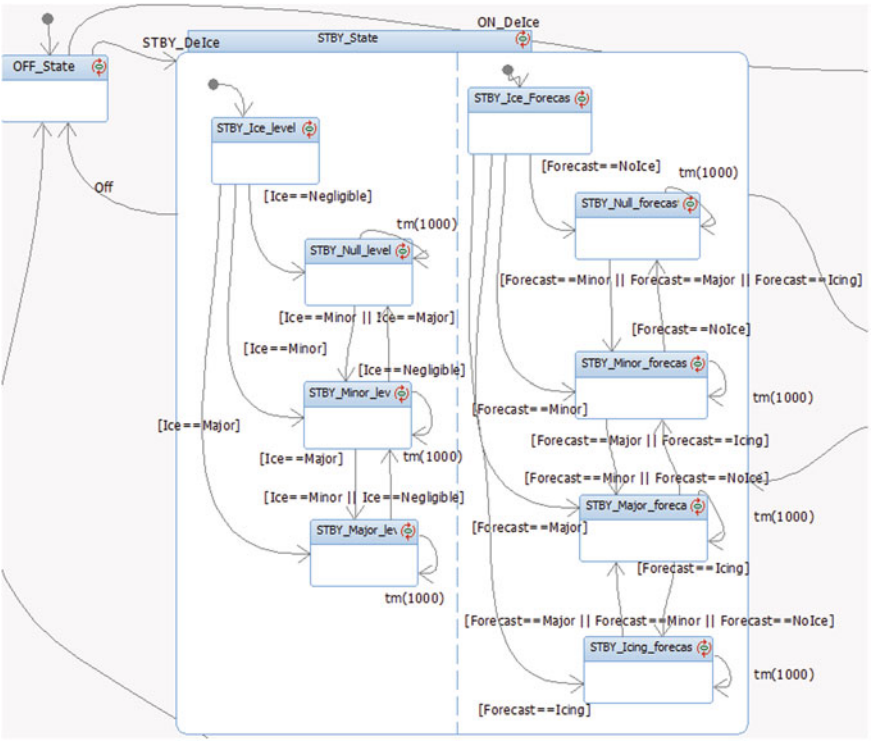


Fig. 6.27 Detail concerning the "Off" and "Standby" states of the SMD related to system monitoring

the different transitions to show the ice level, forecast and all information related to system control (system status, type of cycle, mode etc.…). Those states are involved as the guard conditions associated to transitions define. Some timing triggers are used for updating.

Some elements coming from other SMDs are even replicated, if it is necessary, but, generally, the data coming from other diagrams are used to compute the output requested, as it is appreciated for the main states of Fig. 6.28.

As it appears now clearer, different SMDs shall be associated to each other, through an effective representation of data architecture, to be simulated together. This is a preliminary integration of the system and is done by defining an Internal Block Diagram, shown in Fig. 6.29. The IBD is composed by the *Parts* which are expression of the use cases in this specific context. Each part is an instantiation of a class, and is typical of the SysML language, as previously described. The parts show interfaces and ports required to expose data to other parts or instances. These send or receive some variables (continuous data) and messages (discrete data) to feed the SMDs. As the small icon upon the blocks indicates, the SMDs include the dynamic behavior of the parts here defined. Therefore, the IBD provides an enriched information, and its representation becomes powerful and now complete, in terms of the Functional Architecture of the IPS.

Once that this kind of instantiation is complete, it is possible to perform the simulation. For this application, a dedicated panel diagram was built, to provide a user interface to the SMDs. The panel is shown in Fig. 6.30.

On the left side, the main inputs are included and, notably, the main switch which regulates the "on—off—standby" modes, the selector of auto and manual modes, and the indicator of de-icing intensity. In the middle, an overview of the aircraft with the indication of active zones is depicted, and, at the bottom, the control panel for manual activation of zones is provided. On the right side, the main environmental parameters can be set up (ice level and temperature). The panel is completed by a small selector of the prediction type (on the right) and the visualization of the actual thickness of ice (bottom right). Figure 6.31 shows an example of simulation.

It can be appreciated that the system is running in auto mode. The ice level on the wing (displayed through the indicators on the right) is initially considered to be low, if compared to levels defined by requirements, and the low de-icing cycle is used. However, since the temperature belongs the range suitable for icing and the ice is accumulating on some aircraft zones, the forecast at the right corner of the panel warns that a major icing condition occurs. This message is displayed according to the forecast strategy implemented, being based on both environmental conditions and ice measurements (push button "Both"). This motivates why the wing surfaces are active in the picture, as it is highlighted by a yellow led.

In addition, the user is interested to appreciate inside the SMD where the simulation is currently working, and the states active, step by step during the simulation. This is possible, since the animation visualizes by different colors the blocks involved, and allows validating the model and verifying whether a right transition

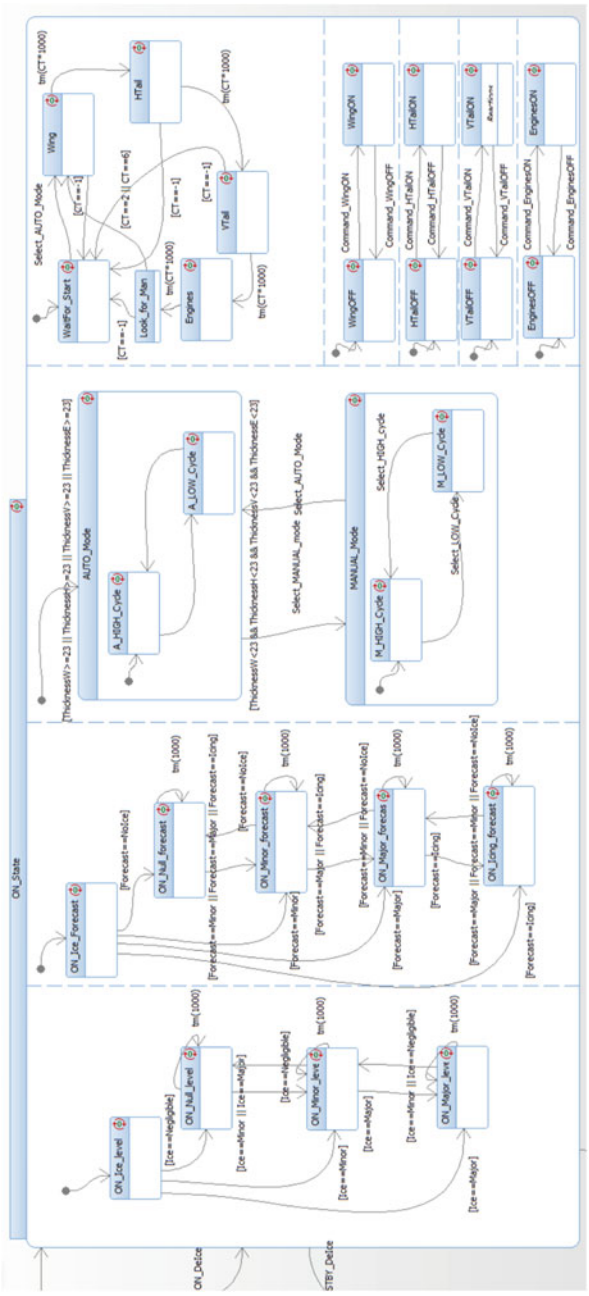


Fig. 6.28 Detail concerning the "On" states of the SMD related to system monitoring

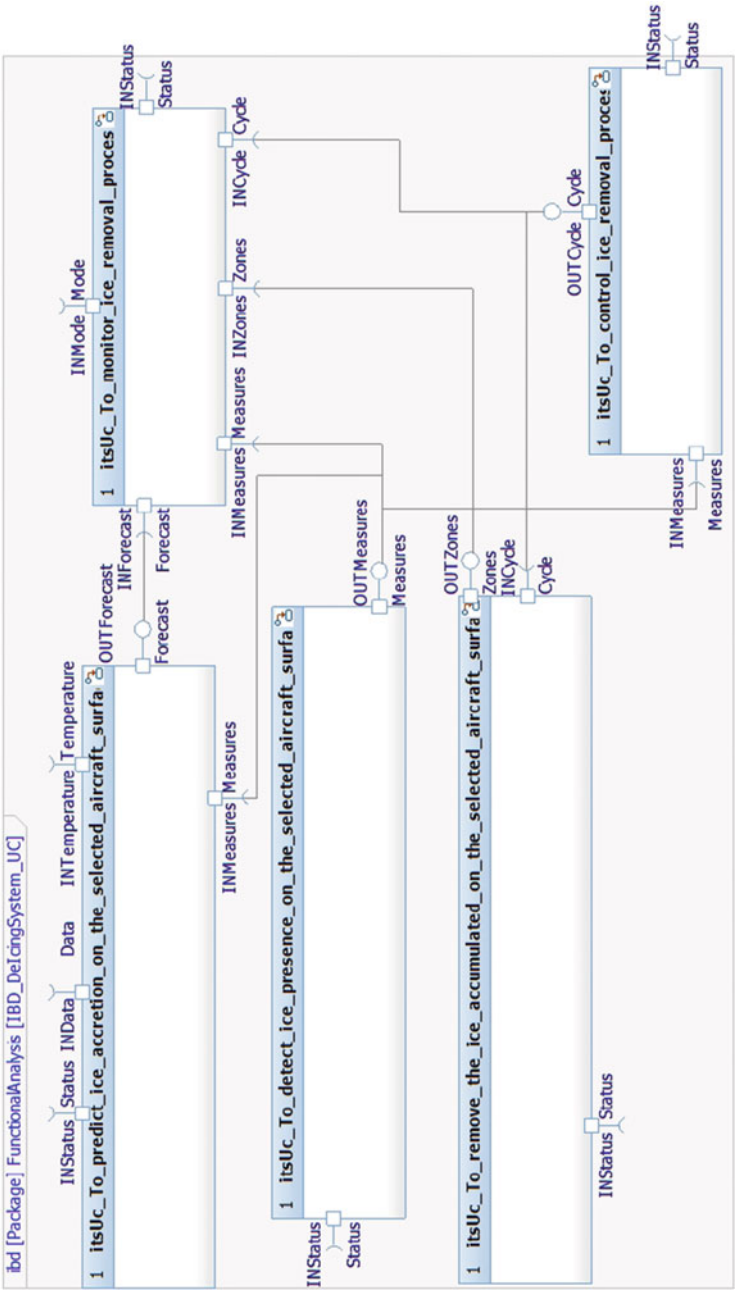


Fig. 6.29 IBD defined to connect the different SMDs describing the system use cases

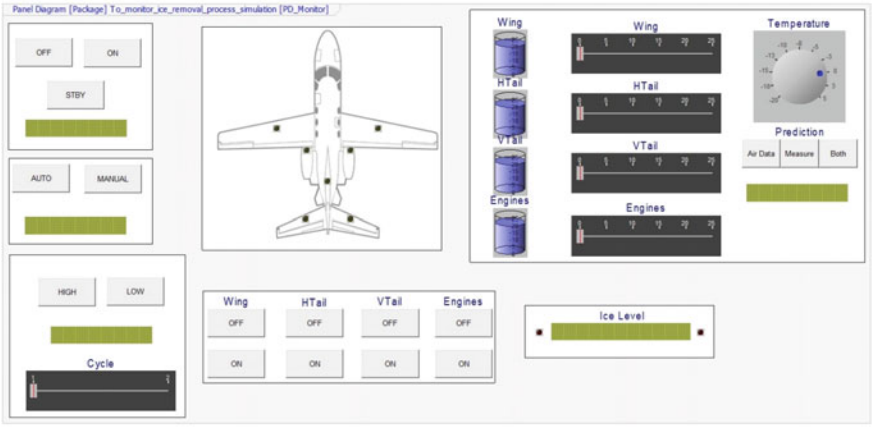


Fig. 6.30 Panel diagram used within the functional analysis for the IPS case study

between states is predicted. Figure 6.32 shows the diagram animation for the situation depicted in Fig. 6.31.

Active states are shown in magenta. The focus is upon the "On" state, being the system working. For what concerns the ice measurement, a "minor" state is highlighted, whilst data are flowing, as well as a "major" state is highlighted for the forecast. The system is running in auto mode, set at low de-icing cycle. The wing de-icing is presently active, as highlighted in the upper right state of the "auto" sequence, whilst the commands on the bottom right, dedicated to manual control are disabled.

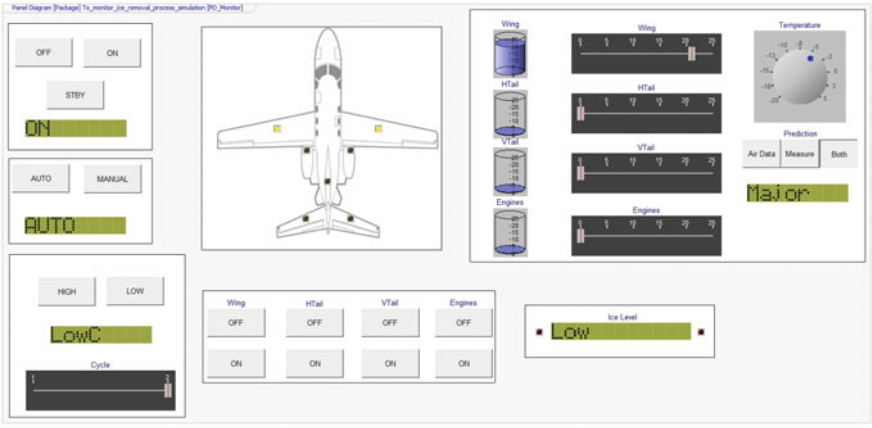


Fig. 6.31 Example of Panel diagram animation (functional simulation)

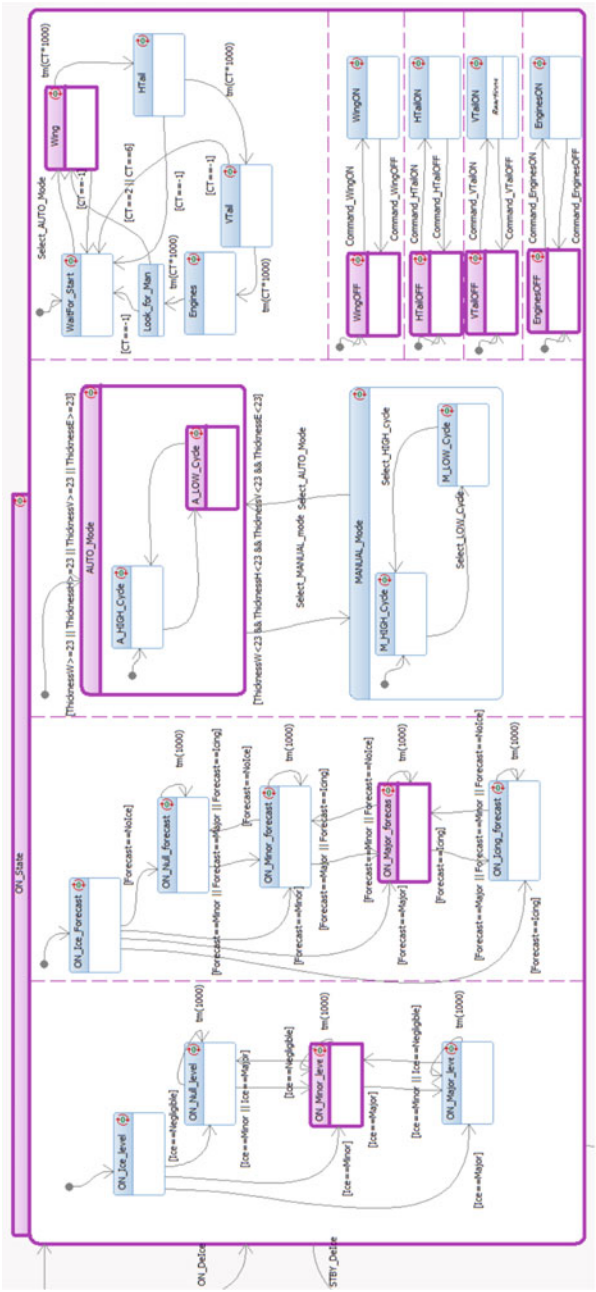


Fig. 6.32 Animation of the SMD for the system monitoring use case (detail of "On" state)

6.6.3 Comparison Between Use Case and Black-Box Based Approaches

Several scenarios can be simulated through that visualization. It is also a good assessment to verify that the model has been defined correctly. This activity completes the modeling of the Functional Analysis, when the whole set of behavior diagrams is used for the instantiation of operations. This approach does not allow a clear representation of the functional breakdown structure, being more focused on the system behavior and on its formal representation, through the simulation. It is effective in terms of results, but it may be difficult applying this approach to fields where functional trees and best practices are commonly used, or some regulation or connection with other domains drives the design activity.

The aerospace engineering is a typical field, where design practices are today fairly assessed, from the point of view of regulations and workflows, inside the companies. The last are expressively adapted to fulfil the requirements of technical standards, about the quality control for safety critical systems. This situation is a consequence of the document based approaches, used in the past to connect different engineering domains and several departments, involved into a product development process based on a waterfall methodology. In that case, the functional tree, or functional breakdown of the system, is used for many purposes. It might be remarked that together with the SE a very important branch of engineering focused on system functions is the safety engineering. The safety assessment for safety critical systems usually starts from the definition of failure conditions. They affect the nominal behavior of the system and cause the loss of some functionalities. This investigation is performed through the Functional Hazard Assessment or FHA, which is based on the system functions to explore its dysfunctional behavior (see Chap. 10). This motivates the relevance of the functional tree within the Functional Analysis. Therefore, the two approaches are herein compared, in the industrial test case.

The black-box based approach to the Functional Analysis starts from the definition of a proper functional tree, implemented through a Block Definition Diagram (BDD). It should be capable of identifying the functions as main blocks, to be then characterized by a proper behavior. In this way, a clearer understanding of the functional levels is provided, without losing the benefits of the MBSE design. The BDD is shown in Fig. 6.33.

As it can be appreciated, this representation is quite intuitive, each block has a precise hierarchical relationship with the father/son blocks. They can summarize many types of data, which are related to the analysis performed. For instance, are evident the information about operations associated to different functions, the allocation to logical/physical components and the links to requirements. In this case, the upper part of the functional tree is just shown. Main functions are here described by some SysML blocks, where properties are highlighted, such as operations and allocations from logical elements. Each block can be further characterized through some SMDs, to specify better the system behavior and to include some operations,

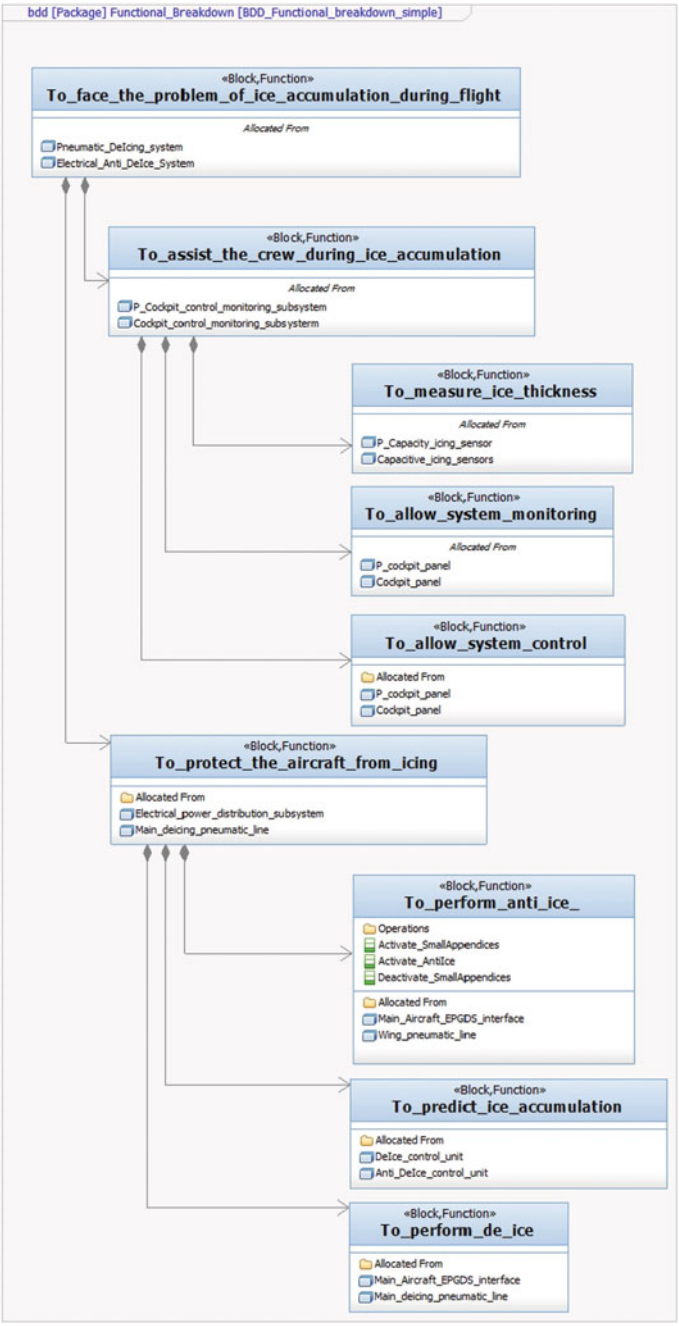


Fig. 6.33 Detail of the BDD containing the functional tree of the IPS

	57	58	59	62	37	38	106
To_constantly_protect_part_of_Wing	57						
To_constantly_protect_part_of_HTail		58					
To_constantly_protect_part_of_VTail			59				
To_constantly_protect_part_of_Engines_nacelles				62			
To_cyclically_protect_part_of_Wing	57						
To_cyclically_protect_part_of_HTail		58					
To_cyclically_protect_part_of_VTail			59				
To_cyclically_protect_part_of_Engines_nacelles				62			
To_allow_system_monitoring							106
To_allow_system_control							

Fig. 6.34 Consistency check for requirements in the functional analysis through a matrix view

associated to the block within a dedicated flow of states. Other behavior diagrams can be used like in the approach above described.

The allocation process is here crucial as well as the design data flows are for the implemented solutions. The black-box approach considerably relies on the SysML features, for traceability, to perform the *consistency* and *coverage* analyses. The first analysis allows checking that every processed data is traced upward to requirements and downward to logical components, without blank spots, whilst the coverage assures that to each logical component a design element is associated. The coverage analysis highlights the instances impacted by a specific function or operation, contained within the functional tree.

The consistency analysis usually refers to some specific representation, as matrices or tables. They can be easily used to check whether any data is missing or has been forgotten, during the design activity. A detail of a matrix view relating functions and requirements is shown in Fig. 6.34.

Function "To allow system control" has no requirement associated and a corrective measure shall be applied to avoid a loss of data, for the next phase. In the same way, some requirements are not assigned to functions.

This is the so-called upward consistency check, from functions to requirements. However, if the downward direction is considered, from functions to implementation, it is possible to perform the same check to continue this sort of chain of data, as it will be shown in next Chapters, as the Logical and Physical Analyses will be described.

Concerning the coverage analysis, a similar implementation is performed. Nevertheless, the coverage analysis is wider, in terms of elements checked, and involves several instances having different characteristics and heterogeneous granularity (functions, requirements, operations, attributes, interfaces etc.…). This is the motivation leading the MBSE tools saving all the relationships occurring among the SysML elements, to make available different views as the one shown in Fig. 6.35.

Many links are shown in the Requirements Diagram. This representation includes the SysML elements associated to the function "To allow system control", from high level requirements up to the level described in this section (Functional Analysis), together with some details of the Operational Analysis.

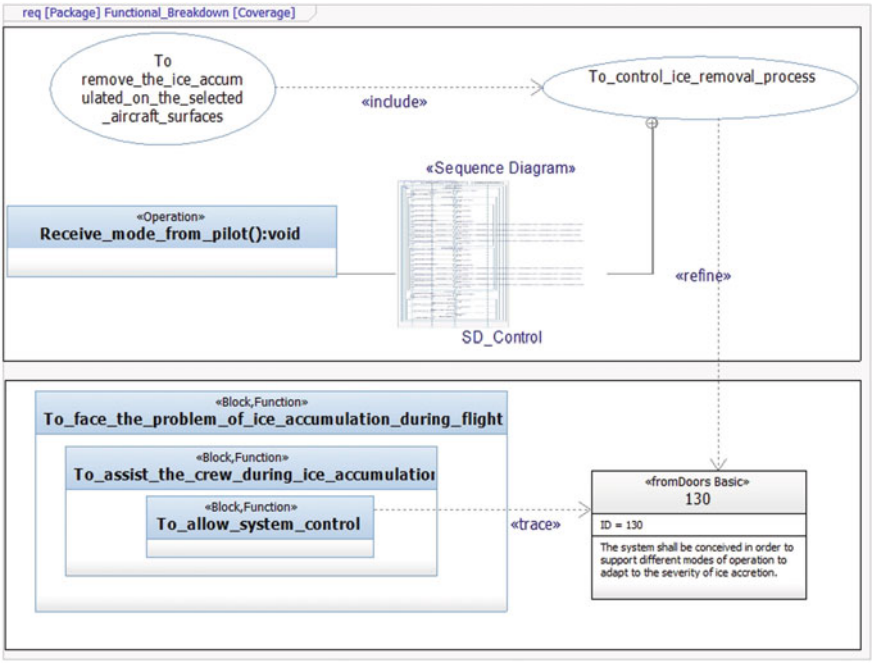


Fig. 6.35 Visualization of an example of coverage analysis through SysML diagram

It looks interesting to think about the flow of tasks through which the system is designed. It starts from a high-level requirement, that specifies the need for a modulation of de-icing intensity, depending on ice level and ends with the definition of the function, and the derived requirement at lower level, passing through some modeling activities, describing the system behavior.

Some MBSE tools already available on the market include some specific toolboxes, which enable those representations. Figure 6.36 shows an impact analysis for the requirement related to system control capabilities, which identify several elements to be associated.

When the deployment of an enlarged toolchain, including tools for different kind of analyses, is considered, the consistency and coverage analyses can be enlarged to verify the correctness of data. The consistency of data between functional tree and FHA is an example. This task could be more difficult when a use case based approach is applied, as long as defining a unique functional tree is hard. By converse, some functionalities of the system can be represented in a federated way, in different diagrams, to follow the use-case-based approach.

The use case based approach allows characterizing better the system behavior and exploits all the SysML diagrams, through a specific sequence of steps. Therefore, it is effective in identifying a lack of consistency between phases, while

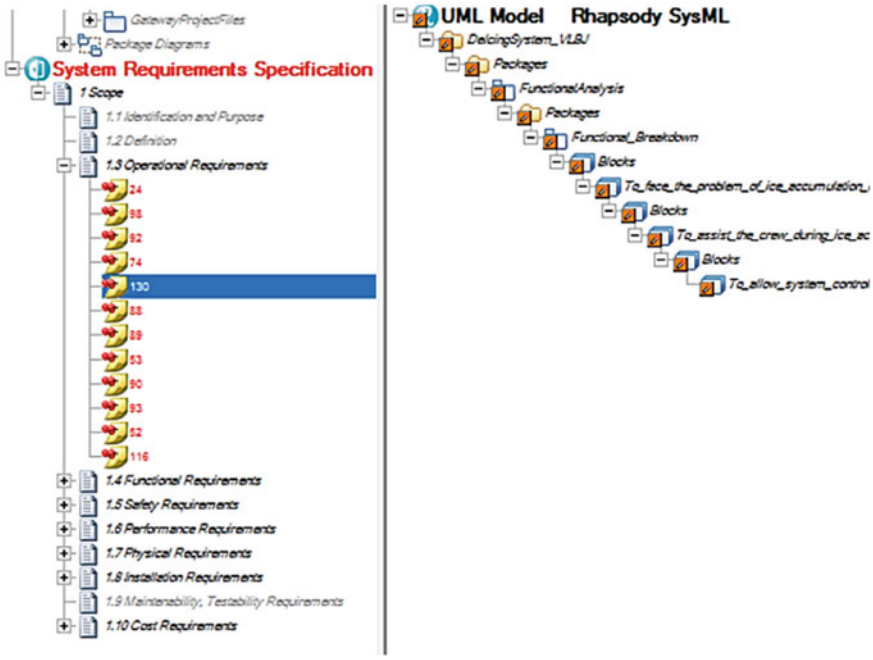


Fig. 6.36 Visualization of the coverage analysis through the Rhapsody® Gateway®

the black-box approach focuses the user attention on the correctness of data, as long as each design phase is deployed.

Particularly, requirements are derived in different ways, depending on the approach used to define the functional architecture of system. They are derived at high level, as in the Operational Analysis, through both visual and summary methods in the use case based method. The granularity of operations is higher than that of functions.

The second approach is based on breakdowns. Therefore, diagrams can be easily structured for requirements derivation. Typically, the BDD or requirements diagrams are used to connect the model elements to requirements, which are then updated within the specification.

In both cases, dependencies are established between requirements and model elements. Considering that the system characterization is still at functional level, some soft links should be preferred. The "Trace" dependencies are chosen, for this example, to connect functions and operations to requirements, to start a preliminary consistency analysis together with an initial impact analysis (coverage) without stressing the relationships (for example with "satisfaction" or "verification" dependencies). This might appear as a just formal difference, but content and meaning of dependencies have an impact on the model-based design and on practical features exploited by tools to enable some consistency checks.

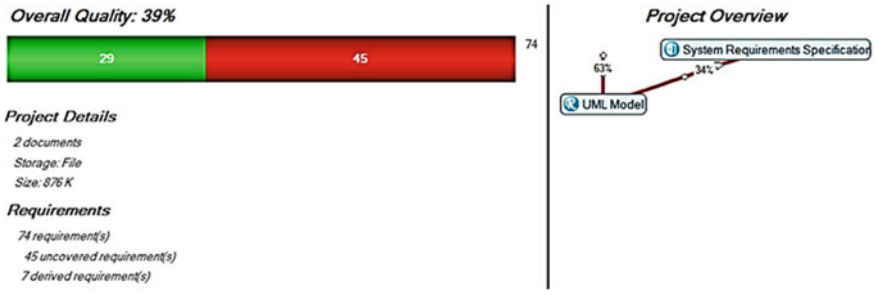


Fig. 6.37 Analysis of coverage quality through the IBM Rhapsody® Gateway® in the functional analysis

Once the functional architecture of the system has been established, is crucial investigating how the specification is modified. In addition, the actual coverage implemented between the IBM Rhapsody® and the IBM DOORS® should be checked. An overview of the IBM Rhapsody® Gateway® is shown in Fig. 6.37.

The level of requirements coverage is expressed by a bar on the left side. In this case, the coverage refers to trace links, between requirements and other objects, since only soft dependencies have been used. The percentage of coverage should be 100% at the end of each phase, and all requirements defined shall be covered by a design element. No isolated design element should be present, otherwise they might be not required. The coverage analysis can be summarized by some views provided by the Gateway®, as shown in Fig. 6.38.

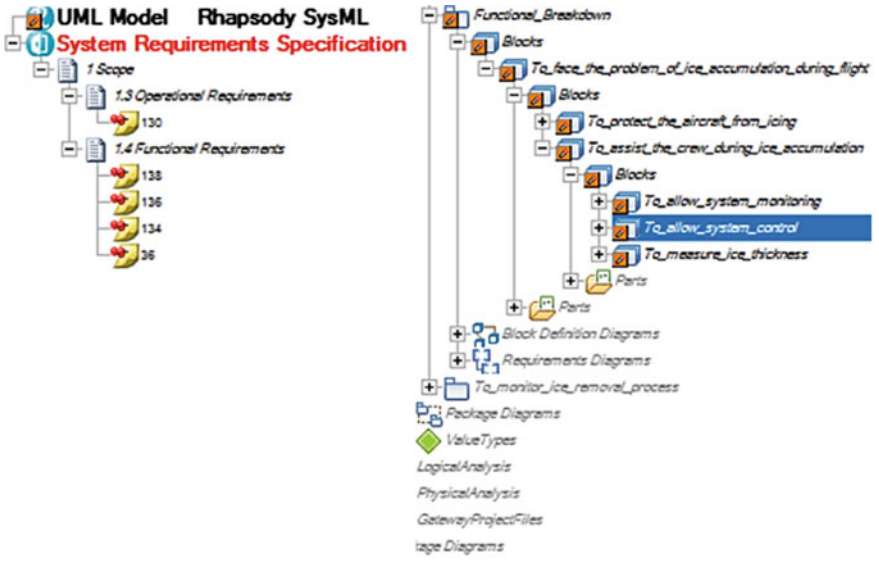


Fig. 6.38 Impact of requirements on system functions as shown within the IBM Rhapsody® Gateway®

This action is generally applicable to both the approaches above described, as far as the use of dependencies is concerned. The specification in DOORS® is then updated, through the connection with the Gateway®. New requirements are added to the formal module containing the specification. The link module can be updated as well within DOORS®, if the connection among requirements is a critical issue.

6.7 Results and Final Remarks About the Functional Analysis

The Functional Analysis is a crucial phase of the MBSE approach. It is used to define the functional architecture of the system, being the synthesis of functional behavior and breakdown. Depending on the case study and engineering domain or practice, this phase can be customized and tailored to focus on some specific aspects of the functional architecture. Different levels of priority may be assigned to the definition of behavior and breakdown, and some approaches can be applied. Some common issues are identified for a large variety of application of the functional analysis, such as its use-case-based characteristics and its deployment as black-box process.

Test cases, here proposed, showed the most important features of the modelling activity, dealing with behavior and structure diagram of the SysML. They proved the effectiveness of these tools, in different fields. Several levels of complexity have been shown. A high flexibility of the proposed approaches could be even appreciated, in terms of depth of analysis.

Some main results can be here summarized.

- Operational scenarios and interfaces with actors have been translated into a realistic behavior of the system, in terms of functionalities. It was possible to define several diagrams to specify the system behavior, in each scenario.
- Interfaces with actors have been validated through the diagram animation, by means of some dedicated diagrams, which allows testing the system behavior.
- Traceability assures the consistency and coverage checks, both upward and downward, between phases of the design process.
- Requirements have been updated and the specification reached a good level of maturity.

The Functional Analysis is break point within the design process. It is the last phase in which the approach is performed through some black-box views. Therefore, in following phases, the allocation process will assume the most important relevance, while the description of behavior issues will be progressively less important. Constructional and physical aspects will have a growing importance. So far, it shall be relevant stating the strategy used to manage the allocation process, and the degree of customization allowed.

Another crucial remark concerns the comparison between functional and non-functional behaviors of the system. Actually, in the test cases, functional requirements will not significantly change. In some complex applications, when some iterations of the design process are performed and the dysfunctional behavior is considered, the requirements specification might be significantly modified.

In next chapters, the Logical and Physical Analyses will be presented and the practical derivation of the PBS will be proposed. For both examples, some main features of the modeling activities concerning allocation and traceability will be described. The results of the Functional Analysis will be used to propose a logical breakdown. It shall be then analyzed in terms of non-functional issues, through a numerical simulation and even other kinds of assessment. Some examples of PBS formulation will be provided as well as some implementation alternatives, currently proposed within the frame of the MBSE.

References

CRYSTAL Project. (2015). *CRYSTAL global glossary.* Graz, Austria [online]. Available at http://www.crystal-artemis-eu/fileadmin/user_upload/Deliverables/CRYSTAL_D_102_050_v0.6.pdf. Accessed 28 September 2017.

Friedenthal, S., Moore, A., & Steiner, R. (1999). *A practical guide to SysML, the system modeling language.* The MK/OMG Press.

Holt, J., & Perry, S. (2013). *SysML for systems engineering. 2nd Edition: A model-based approach.* IET.

The Object Management Group. (2015). *OMG systems modeling language.* Needham (MA) [online]. Available at http://www.omg.org/spec/SysML/1.4/PDF. Accessed 28 September 2017.

Weilkiens, T. (2008). *Systems engineering with SysML/UML—modeling, analysis, design.* The MK/OMG Press.

Chapter 7
Logical Analysis

Abstract Before that a real architecture of system could be defined, by selecting some suitable components, available on the market, it is suggested to describe the Logical architecture of the system, to investigate the most promising solutions available in terms of exploitable technology. This leads to perform a Logical Analysis, as is herein described. Relations between Functional and Logical Analyses are investigated. Examples are exploited to show some properties of this analysis. At the end of chapter, the main items of the Logical Analysis are resumed.

7.1 Meaning of the Logical Analysis

After the Functional Analysis, the *functional architecture* of the system is defined. A real implementation of this architecture is not yet specified, at this level, since the analysis is just performed through a solution-independent approach, i.e. the system elements are described by the functions they must provide, but the real components are not yet included. The purpose of the *Logical Analysis* is proposing a preliminary system architecture to support the *functional behavior*. It is a specific design phase, and is focused on the *allocation* of the functional architecture on one or more system candidates, which should be compliant with the functional requirements and comparable among each other, through a selection of design parameters. This is an intermediate step between the pure functional analysis and the physical one, when the non-functional requirements are assessed and the real performance of the system is measured.

The Logical Analysis connects the functional modeling with the modeling and simulation activity, which usually implements some numerical models, to predict the dynamic behavior of the system. A candidate solution for building the system is found and is organized to be compliant with the functional architecture. It is used to propose a set of possible alternatives, which are identified through the allocation process of functions on some available technical solutions, including subsystems and components, being just identified at high level. In this way, all the candidates

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_7

will satisfy the functional requirements and a technical assessment will be used to evaluate the non-functional features, concerning the items constituting the system.

As it happens for the Functional Analysis, there is never a unique interpretation for the Logical Analysis. This phase is performed after the functional modeling activity, to propose some system candidates, for the *trade-off* analysis, which usually follows. Therefore, in this step the logical definition of the system is assessed, while the numerical simulation and the technical assessment are included in following steps of the product development.

It should be remarked that the logical architecture of the system is always a design issue, but the approaches proposed in the literature to assess it, are quite different. One of the most popular is organizing three phases, as is proposed in this handbook, including the operational, functional and logical analyses. However, the definition of logical and physical architecture, respectively, is not unique.

Sometimes, the logical architecture is described as the whole set of activities used to perform the operational and functional analysis, while the physical architecture is related to the implementation and allocation. In other technical domains, the logical analysis is an intermediate layer, between the functional and physical analysis. It is considered always as the final phase of the functional modeling. Furthermore, the term logical analysis may be used to describe the allocation process from the functional analysis to system candidates, while the physical analysis is simply interpreted as the numerical simulation.

To be straight, the definition of system candidates includes system, subsystems and components not yet fully characterized. At logical level, for instance, a specific brand for a component is never cited or some detail like the part number is never included. The term "logical" itself is used to specify that the proposed architecture does not comply with a product breakdown structure. Therefore, a logical breakdown structure precedes the product breakdown, along the development.

The SysML language offers different approaches to perform the Logical Analysis. Despite the number of slightly different interpretations, the most important issue is the goal of allocating the functional architecture, of identifying a set of candidate solutions, for the trade–off, and to prepare a structure to implement a numerical simulation of the system performance. In this handbook, the *Logical Analysis* is interpreted simply as the last phase of functional modeling, aimed at allocating the functions to the system logical blocks, composing its architecture. Logical blocks will be transformed into physical components, through the *Physical Analysis*, intended as a numerical simulation campaign.

7.2 Handoff Between Functional and Logical Analysis

All the relevant information used to map the functional architecture of the system shall be imported into a suitable engineering solution, when some candidates will be generated. The functional elements, like actions and *operations*, shall be allocated on the system elements of the physical architecture. This target is relevant to understand

how the product development switches from the functional to the logical analysis. Two details must be considered. A first one consists of the solution independent nature of those two phases. The functional analysis is a "*black-box*" where the system behavior is defined only through the functions, but without referring to any element able to perform any action or operation. The logical analysis is expressively focused on the assignment, namely *allocation*, of the functions to some entities able to perform both. The Logical Analysis looks like a "*white box*", since the system is explored in its parts, at least up to a defined level of depth.

A second important aspect concerns the management of the use cases during the synthesis of the logical architecture. They shall be able to support all the operational scenarios. Independently on the method applied and the tool used, the allocation process collects the data for the defined use cases and is performed for each one separately, since the traceability is assured by the model based approach. At the end, the system elements will be allocated to functional elements defined in each use case and operational scenario. This allows a complete coverage of the handoff process.

The characteristic independency of the solution and the use case oriented process have a direct impact on the implementation strategy of the Logical Analysis. Most of the diagrams presented in Chap. 6 are used also for the Logical Analysis. A use case based approach is applied, but a strong "white box" connotation is introduced. Both the behavioral and structural diagrams are organized to actively include the system elements within the definition of actions, operations, messages and data network, thus promoting a first logical implementation. This usually involves the diagrams already defined within the Functional Analysis, with some suitable modifications. A global perspective helps to define the system architecture, in terms of hierarchy and data network. It allows comparing different solutions and to pass the related information to other domains, to perform further analyses.

The definition of solution candidates is an iterative process, mainly related to the allocation. When a set of alternatives is considered, the allocation might be different. Some system elements may meet the functional architecture identified in many ways, this property leads to several possible layouts. To select the best candidate and eventually modify the system architecture, a set of parameters is used, mainly characterizing the non-functional issues of the system design. They are collected within a global representation of the system, to enrich as much as possible its description and to identify some performance indicators, to be measured during the simulation activity.

7.3 Implementation of the Logical Analysis Through the SysML

The Logical Analysis involves many diagrams used within the Functional Analysis, but it is specifically focused on the integration of an architecture within the system boundaries. Each action, operation or message is referred to a specific logical

element, and the whole architecture represents an alternative solution, to be considered during the trade–off activity. Some behavioral diagrams, as the activity, sequence and state machine diagrams, include some partitions, which represent the system components and pats. The structure diagrams, for instance, include some data network and the hierarchy of system elements, with reference to the allocation provided by the behavior diagrams. The diagrams used in the Logical Analysis are basically the same of the Functional Analysis. Often the view exploited to represent the system is different.

The Activity Diagram is the first SysML formalism applied to develop the Logical Analysis. It describes a first partition of actions, by allocating them on some physical elements. Actions and related operations, together with the control flow, are reorganized to be assigned to some specific partitions, in this representation which is a 'white box', but remaining consistent with the 'black box' of the Functional Analysis (Fig. 6.1). In case of the Activity Diagram, these partitions are typically shown as *swimlanes* (Fig. 7.1).

The Sequence Diagrams, are then used. They are typically populated by many *lifelines*, related to the system elements. The messages among them and the environment (i.e. the external actors) are reorganized and, eventually, updated. Since the analysis is use case based, those diagrams are always related to some specific operational scenarios. The use of a "white box" version of the Sequence Diagrams leads inevitably to the definition of a more detailed sequence of messages, with a new input/output structure for the lifelines (Fig. 7.2).

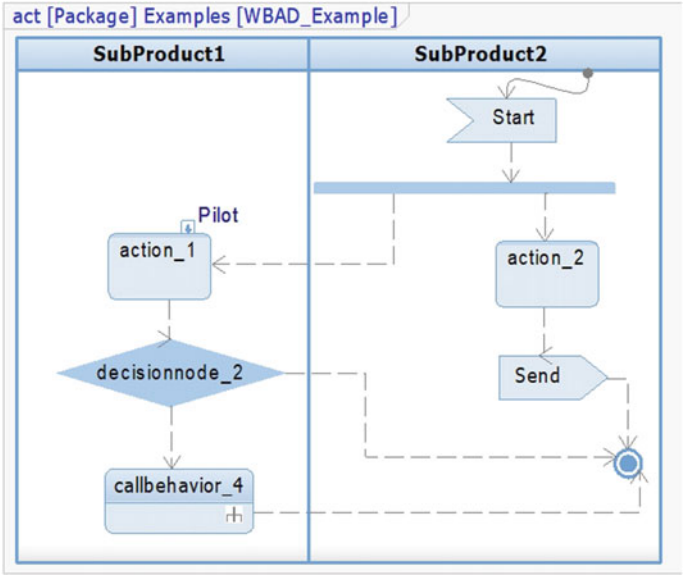


Fig. 7.1 Example of activity diagram with swimlanes

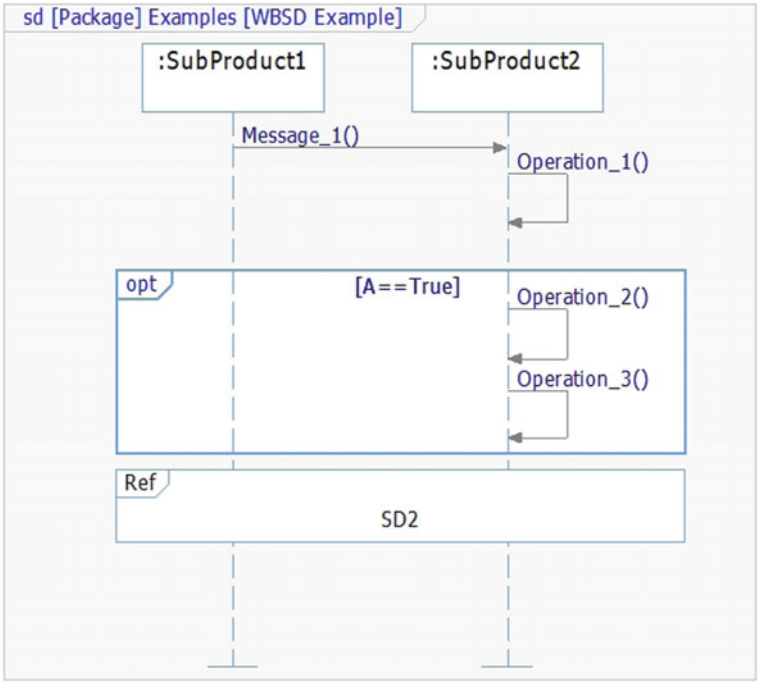


Fig. 7.2 Example of sequence diagram used within the Logical Analysis

This detail has an important impact on the State Machine Diagrams and even on the Internal Block Diagrams. Those two diagrams are now considerably connected to each other. Since the structure of the State Machine becomes federated, to characterize the system elements (Fig. 7.3) and a more detailed data network, based on the messages circulating among the parts, is required to completely understand the system behavior. Each new element is characterized by a specific behavior, which can be implemented through a dedicated state machine diagram. All these diagrams are integrated with the main one, representing the use case.

The network is implemented by a reviewed version of the Internal Block Diagram. Ports, connectors and interfaces among the parts are reorganized to meet the information provided by the behavior diagram, and new Internal Block Diagrams may be included. Several IBD (Fig. 7.4) are drawn at this phase, to formalize the data exchanged among the blocks and to hypothesize a possible interface architecture. This step is also important to prepare the tasks performed within Physical Analysis, through the numerical simulations.

Finally, the logical architecture can be defined within the Block Definition Diagram (Fig. 7.5), which contains the breakdown of system elements. This representation includes a list of blocks and can be used to formalize the allocations and

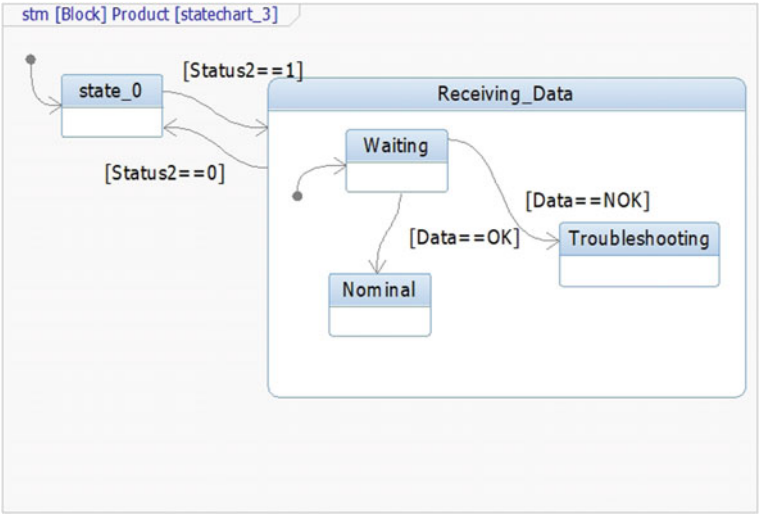


Fig. 7.3 Example of statechart for a logical component

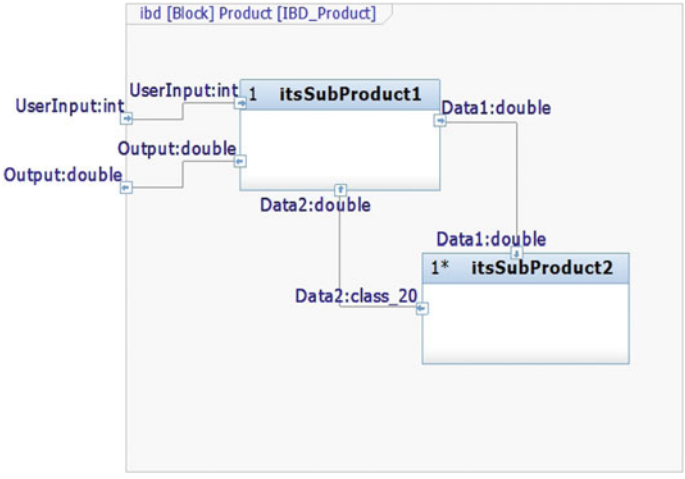


Fig. 7.4 Example of Internal Block Diagram for the Logical Analysis

the different *dependencies*, useful to tailor the Functional Analysis onto the Logical Analysis. Matrices and tables behave as a summary of the handoff and as coverage report.

Moreover, different BDDs can represent the proposed logical architectures, to evaluate the identified alternative solutions.

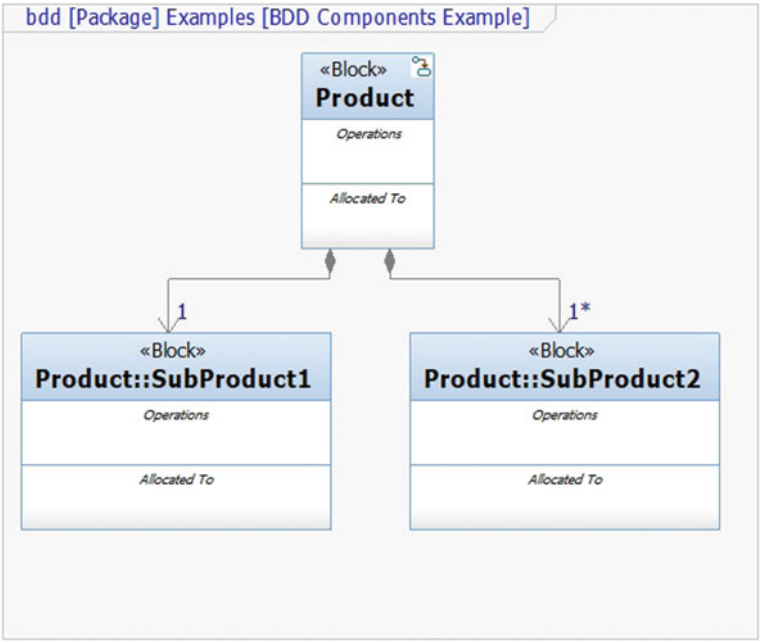


Fig. 7.5 Example of Block Definition Diagram for a logical decomposition

7.4 Requirements Satisfaction and Architecture Allocation

The requirements and the Logical Analysis are connected to couple the preliminary solutions for building the system to the SRS. This process is performed as it happens in the Functional Analysis, but a stronger connection is instantiated in terms of traceability. The so–called *satisfaction* is used to link the requirements to the system logical blocks. A complete map among requirements, functions and system elements is built up, together with a full verification of the compliancy of the proposed alternative solutions to the functional architecture. The requirements coverage can be verified through the Model Based relations defined during the functional modeling. They create an effective collection of links among the requirements, functional and logical analyses. An expected outcome consists in a complete prediction of the mutual impact of the system elements on their design. A crucial task is the verification of the coherence of the system architecture with the functional one. The compliance with the Functional Analysis must be guaranteed, to start a trade–off between the proposed solutions. The *allocation* is used to map the functional architecture to the logical one, been directly derived. Even other types of links are established among functional and logical entities. The whole set of properties regarding the *traceability* is here part of the model. This is the main

benefit of the Model Based approach, if it is compared to the Document Based one. The information can be actively navigated and the impact analysis is effectively performed. The presence of two types of architectures, being focused on the system functions and elements, respectively, is helpful in detecting the flow of information leading the requirements from an early elicitation to the final allocation.

7.5 Towards the Next Phase

Independently on the approach applied through the Functional Analysis, being either behavior–based or breakdown–based, the Logical Analysis is conceived to propose one or more system architecture as a tentative solution to the problem of designing the new system. The main result expected concerns the structure of the system, with its hierarchical levels, a set of new *allocation* links and a stronger connection with the SRS. Some new versions of the SysML behavior diagrams, looking as a "white box", are therefore expected, together with some IBD and BDD, which might offer a logical view of the system. To set up the Physical Analysis and related numerical models, some other outputs are required.

The logical system layout shall be characterized by some parameters, namely *attributes*, to be used to define some performance indicators. Those are computed to perform the selection of the best solution, among those proposed. The attributes are referred to some high-level assumptions about the system behavior, such as the power consumption or other Key Performance Indicators (KPI), to be verified by means of the Physical Analysis. A proper allocation of non-functional requirements is therefore formulated at the end of the Logical Analysis, to be further developed in the following Physical one.

For the transition to the Physical Analysis, the *model compatibility* is very important, i.e. to make easier the modelling process and reduce the workload associated, the data contained in the SysML Structure Diagrams must be compatible with those defined within the numerical simulation model. This turns out into a seamless translation of data between the functional and numerical models. The data network implemented through an Internal Block Diagram, in terms of flows, variables and connections, for instance, should be as close as possible to the structure of the dynamic models simulated in the Physical Analysis. Moreover, data should be automatically transferred from the IBD to the dynamic model, to avoid replication. So far, the Logical Analysis is often seen as a bridge between the functional and physical worlds. Nevertheless, to be effective, it needs to be robustly consistent, in terms of contents (allocation process, link between functional and logical analysis), and formalisms. This specific item shall be described in following sections, aimed at providing an overview of the application of this methodology to the test cases.

7.6 Implementation and System Logical Architecture

7.6.1 Didactic Test Case

The operational and functional analyses defined the system mission, including the interaction with the environment, at least at higher level, and the functional behavior of the system, described as an ideal behavior. Requirements have been derived and updated, considering the results of the analysis performed. A complete overview about how the system shall behave and what it shall do is now available.

To propose a feasible system layout, the Logical Analysis is run, to identify some system components which may match the modeled functional architecture and are able to perform the required system functions.

As a first step of the Logical Analysis, the subsystems and components to be integrated into the real system are looked for. The Product Breakdown Structure (PBS) of the system is modeled, through a BDD defining the system hierarchical structure and including all its components.

As Fig. 7.6 shows, the PBS of the laying head system is composed by three main subsystems, according to the Functional Breakdown Structure, depicted in Fig. 6.7, which shall enable the system to rotate, to measure some parameters, and to control the rotor unbalance and its dynamic stability.

The laying head system architecture should include the following components:

- the rotor subsystem—made by motor, rotor, stator, nozzle, and bearings;
- the sensors subsystem—including position sensors, package sensors, encoder, rod detector, and thermocouples;
- the ECU subsystem—is the Electronic Control Unit, with a power amplifier.

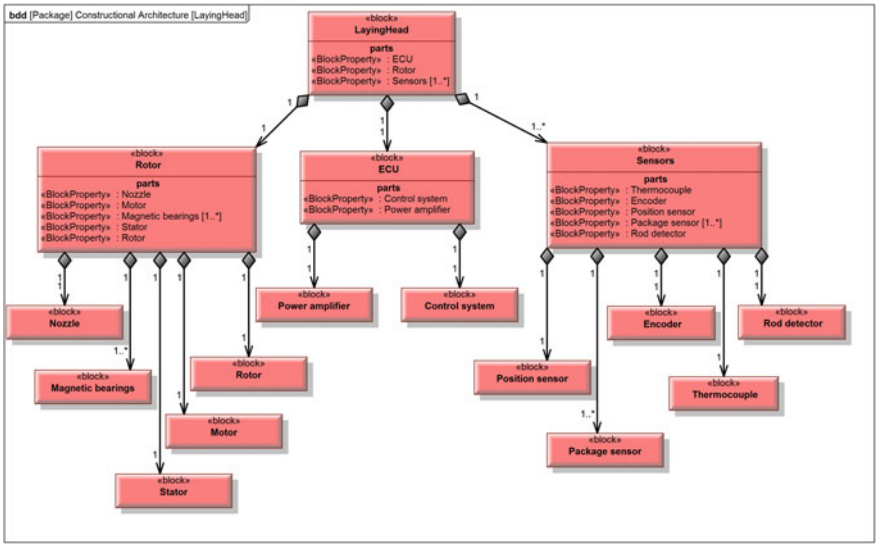


Fig. 7.6 Logical architecture of the Laying Head System

Like in the FBS, the Product Breakdown Structure is sketched by a BDD where the system is the main block and its blocks, here the "Rotor", "Sensors", and "ECU", are subsystems, made by the following components:

- Rotor: nozzle, bearings, stator, motor, and rotor.
- Sensors: position sensor, package sensor, encoder, thermocouple, and rod detector.
- ECU: controller, power amplifier.

Since the Logical Breakdown represents the *allocation* of the FBS, each component defined in the PBS must accomplish at least one of the system functions described within the FBS. The system functions need to be allocated to the system design components, using the allocation link "Allocate to" and "Allocate from".

Figure 7.7 shows that all the function blocks are allocated to a logical subsystem. In particular it can be seen that:

- The Rotor is required to perform the rotating and shaping functions.
- The Sensors are necessary for the measuring functions.
- The ECU is required to perform the control, monitoring and signaling functions.

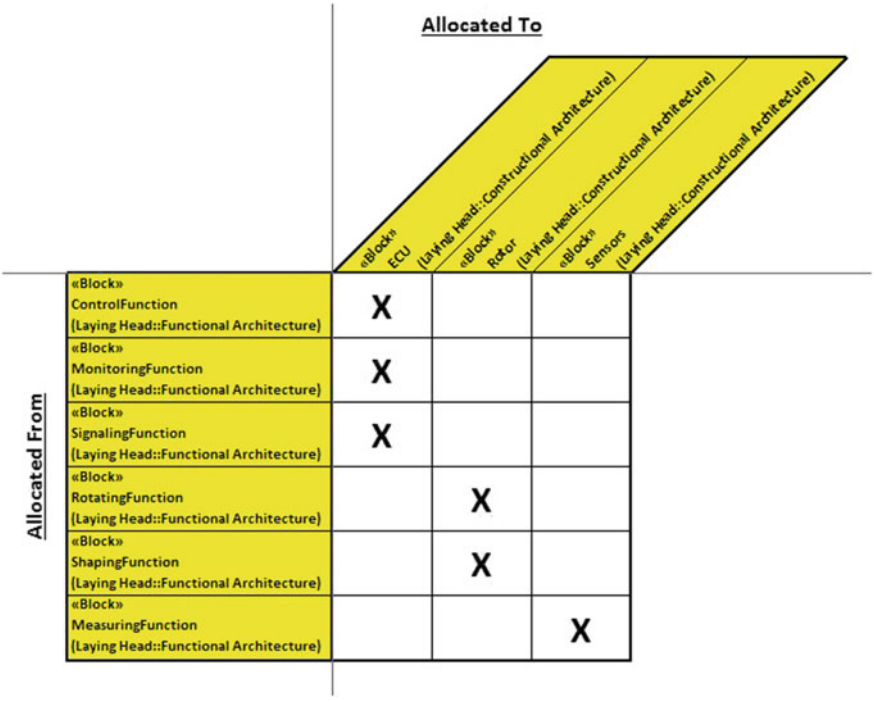


Fig. 7.7 Allocation of functional blocks to logical blocks

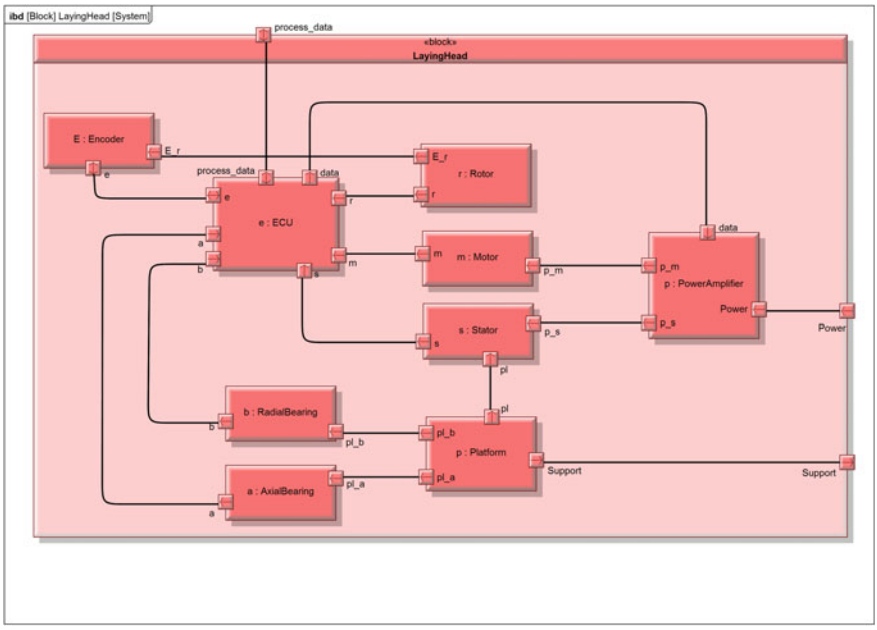


Fig. 7.8 Internal Block Diagram of the Laying Head system

Another important issue of the Logical Analysis is the definition of relations between the components, in terms of data exchanged. Thus, like it has been done in the Functional Analysis, some Internal Block Diagrams are used to model the interactions between the identified components. In Fig. 7.8 the internal structure of the laying head system is modeled through ports and connectors, which even express which data they exchange each other.

The requirements traceability is relevant target of the Logical Analysis. It consists in verifying that the whole set of functional and operational requirements is allocated on some suitable component, being capable of completely satisfying them.

In Fig. 7.9 an excerpt of a requirement diagram represents the dependency between requirements and logical components, defined by the logical analysis. The dependency is here designated with the notation ≪satisfy≫, which indicates that each element satisfies a specific requirement.

As it can be appreciated, in this example the FBS is simply allocated to the PBS. Basically, functions are depicted at higher level through a BDD, then components which provide those functions are described through a IBD. Nevertheless, the rationale is followed as it was above introduced. Functions are preliminary defined and depicted, components are then described, in terms of logical blocks performing a defined activity. The PBS shall be refined and updated to include some selected

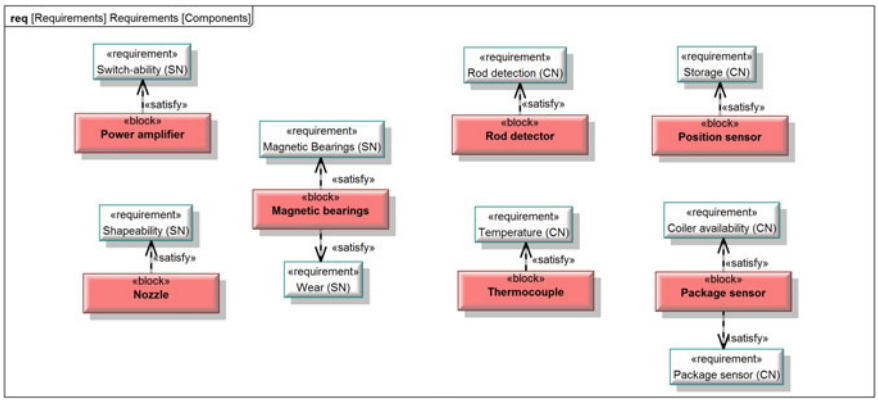


Fig. 7.9 Formal requirements satisfaction for the Laying Head system within a Requirements Diagram

physical components, as the Physical Analysis shall be completed. The diagram is the same, but contents will be better specified. From this point of view, it might be assumed to deserve as a Logical Breakdown Structure right now, and then really as a PBS, when completed.

7.6.2 Industrial Test Case

The Functional Analysis allowed describing in detail the functional architecture of the IPS system, in terms of breakdown of capabilities and behavior. The focus shall move now on which component is responsible to accomplish the defined functions. A preliminary decomposition of logical components, to be subjected to an assignment of system capabilities through allocation, is made. Looking at the process followed for the IPS, the results obtained from the Functional Analysis provided some important inputs.

- The exploitation of the main behavior diagrams allowed defining the operations of the system (behavior-based approach). Alternatively, a functional tree has been implemented to consider the whole set of system functionalities (breakdown-based approach).
- Requirements have been traced on functional elements, so that the traceability links are now available to drive the formal satisfaction in logical analysis.

The Logical Analysis is based on two main processes, the *allocation* of functions on logical components and the *satisfaction* (traceability) of requirements. To reach these goals, four main steps are implemented.

Allocation of operations/functions on logical components. This is the most important task of the analysis, which consists in formulating some possible system

alternatives, to match the required functionalities. Depending on the way the Functional Analysis has been conducted, the allocation may use a "white box" version of the behavior diagrams previously developed. In this case, the allocation of operations is performed directly on the diagrams, using the features of the SysML formalism. Alternatively, it may rely directly on a logical breakdown of components. The first approach is required when a behavior-based functional analysis is considered, whilst the second one is applicable to the breakdown-based approach.

Definition of the system logical architecture. At the end of the Logical Analysis, a tangible architecture of the system shall be defined. This means that, for systems designed through a behavior-based Functional Analysis, a further step shall be performed to synthetize a breakdown of the system. For a breakdown-based Functional Analysis, it is already available. The logical architecture shall include both a hierarchical decomposition of elements, being usually sketched through a BDD, and a preliminary communication network among them, which can be close to a possible implementation and is, generally, described by one or more IBD.

Requirements satisfaction. Logical elements shall be linked to the traceability process, through the satisfaction dependencies. This activity has an impact on the functional requirements of the SRS and allows closing the loop of requirements coverage as well. Non-functional requirements can be defined to enrich the SRS, but their satisfaction will be assessed during the Physical Analysis. At the end of the Logical Analysis, it shall be possible to navigate the traceability links, from the use cases to the logical blocks, without blank spots.

Definition of system alternative solutions and trade-off study. Several alternative solutions can be proposed within the Logical Analysis to meet the functional architecture. In principle, different kinds of system may be compliant with the SRS. A trade–off is therefore required to choose the best option. This is usually done through the formalisms of the SysML, that allows defining the candidates and the scoring process, in a standard way.

Several system architectures are usally proposed. For each one, many Key Performance Indicators (KPI) are introduced, using some attributes. The KPI are expression of non-functional aspects related to the system or to its operation. These attributes will assume some hypothesized values, to be then verified, during the physical analysis. However, their introduction in the Logical Analysis suggests that some of candidates can be neglected.

It is worth noticing that, even if the breakdown-based functional analysis allows avoiding an additional step in the Logical Analysis, the behavior-based approach is easier, since the consistency check on the completeness of information is faster and usually embedded within the MBSE software. For the breakdown-based approach, this shall be performed manually, with some additional time and efforts.

For the IPS, the behavior-based approach is chosen as a starting point for the Logical Analysis, since it allows covering all the four steps just described. Moreover, two system alternatives have been proposed to show the scoring process within a trade study.

An interesting way to show the allocation through the "white box" behavior diagrams is relying on the activity diagrams. They define some specific formalisms, namely *swimlanes*, which represent a sort of partition of the diagram, that can be used to indicate the components responsible for each *action*. However, since the activity diagram is the first behavior diagram developed within the approach herein described, the *operations* were not yet present, when the graph was defined at the beginning. Therefore, a preliminary synchronization and updating is required to assure that the actions described within the diagram are representative of the operations depicted by the related sequence diagrams. Apart from this aspect, the implementation is quite straightforward. It is simple to be understood, although the diagram is usually affected by the introduction of the partitions. It looks like cluttered, although the contents are the same. Moreover, this approach allows maintaining the use case based approach up to the allocation process. This is very useful to look at small parts of system behavior, reducing the risk of data loss and misunderstandings. The whole picture can be then provided, as soon as the system architecture is defined and requirements are linked to the logical blocks.

The "white box" activity diagrams are herein shown for the use cases considered for the IPS case study. The first one is referred to the ice prediction use case (Fig. 7.10). It is a modified version of the diagram used within the Functional Analysis, and differences concern the introduction of the swimlanes. Particularly, the main flow is allocated to the de–icing control unit, which is responsible to receive data related to icing condition, to elaborate them and to compute the forecast. Information are provided to the crew, by means of a dedicated visual indicator, that shall be able to distinguish between major and minor icing conditions. Eventually, the main switch is responsible to turn on and off the system.

For the diagram in Fig. 7.10, the type of IPS is not yet specified. The system components where the operations are allocated simply represent a generic equipment, being associable to many technologies and categories of products. The same happens for the ice detection use case, shown in Fig. 7.11. In this case, the measurement of ice accretion is made by several sensors, applied to different surfaces of the aircraft. The control unit receives the measures performed. The components appearing more than once, within different diagrams, refer always to the same item. This is a consequence of the use case based approach, which leads to have that different issues of the system behavior are treated separately, during the Functional Analysis. During the allocation to logical components the system structure is disclosed. Therefore, it might happen that the same component performs similar or even different operations, in several working conditions.

The system nature is dealt with the allocation of the ice removal use case operations. Two candidates of IPS have been selected. An electro-thermal system and a pneumatic one, based on pressurized boots. Figure 7.12 shows a portion of the diagram, to describe the allocation of the electro–thermal solution. The selection of the surfaces to be heated is allocated on a proper selector, either automatic or manual. The power distribution is performed by some feeding lines. The first one is the main feeding line, while the second one is specific, for each zone heated. Some

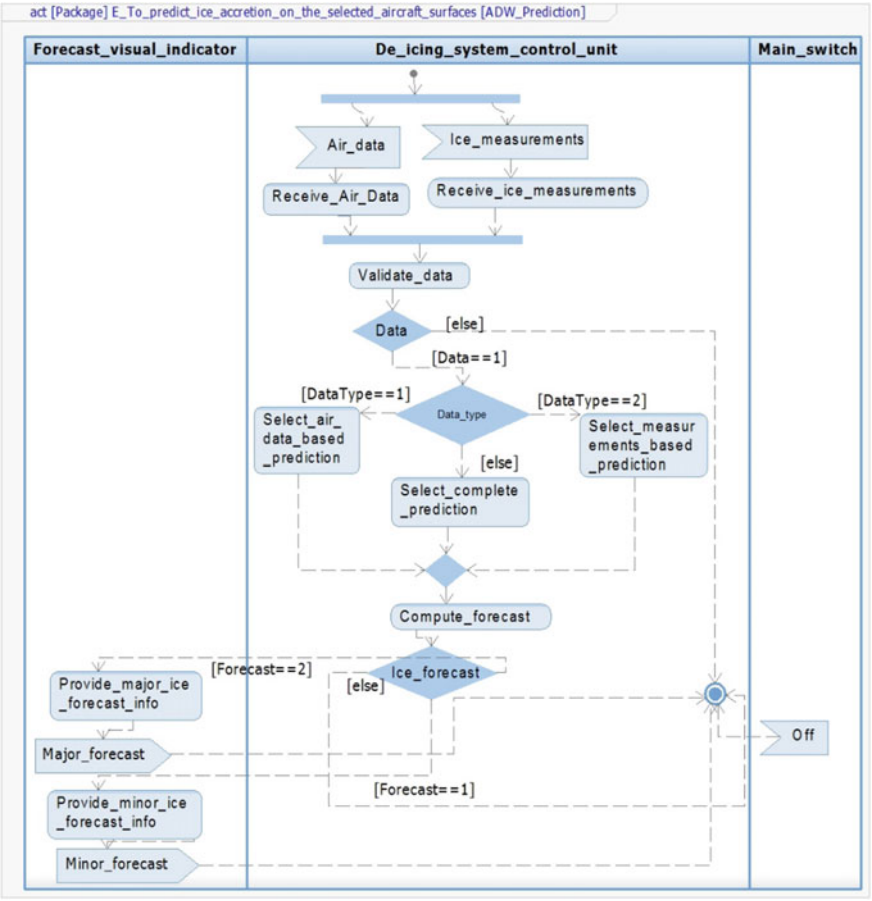


Fig. 7.10 Activity Diagram for the Ice prediction use case with swimlanes

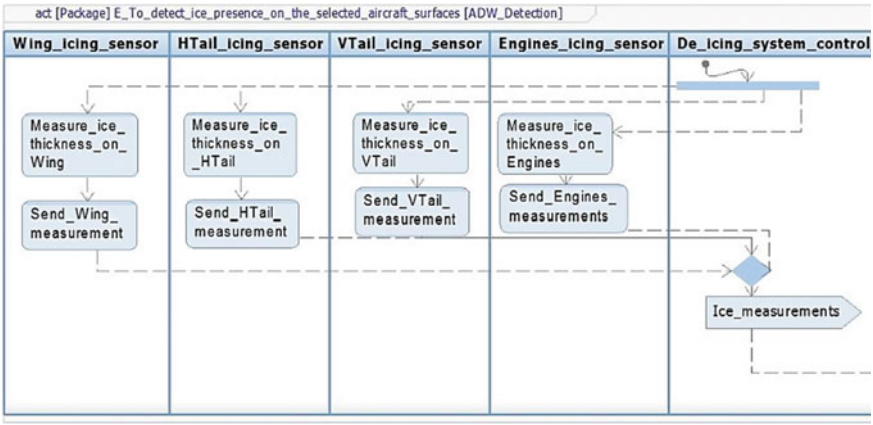


Fig. 7.11 AD for the ice detection use case with swimlanes

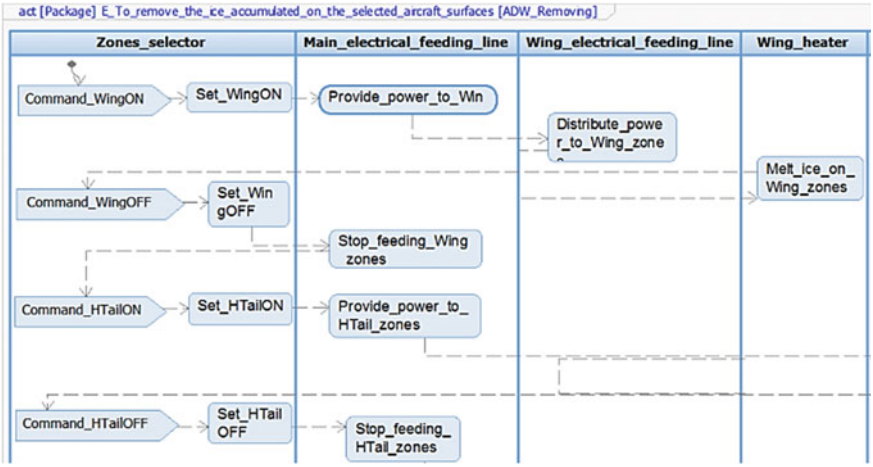


Fig. 7.12 AD for the ice removal use case with swimlanes (electro-thermal system)

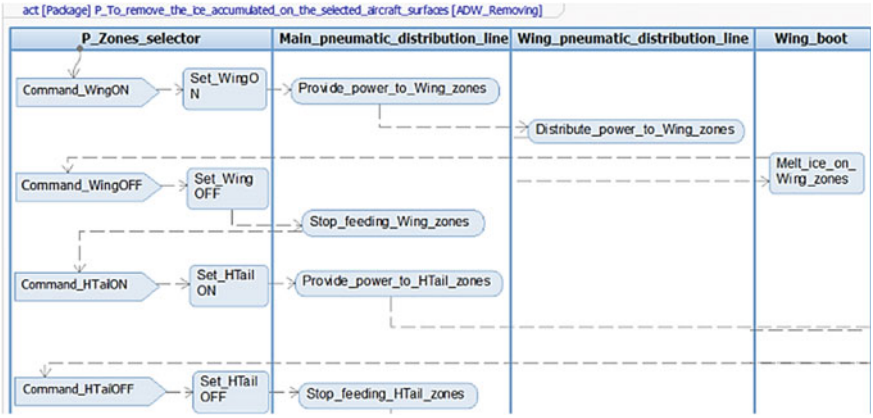


Fig. 7.13 AD for the ice removal use case with swimlanes (pneumatic system)

heaters provide the thermal energy to the external surface of the selected zone to melt the ice.

Figure 7.13 shows a similar allocation for the second technological solution, based on the pneumatic system. The structure is the same, while the distribution lines, consists in a pneumatic feeding. The actuators are inflatable boots, bonded on the protected surfaces, which break the ice through a cyclic inflation and deflation.

This sequence of actions is repeated for each protected surface of the aircraft. Therefore, a feeding line and an actuator system (either electric resistor or boot) are foreseen for the different zones (here only wing and horizontal stabilzers are shown).

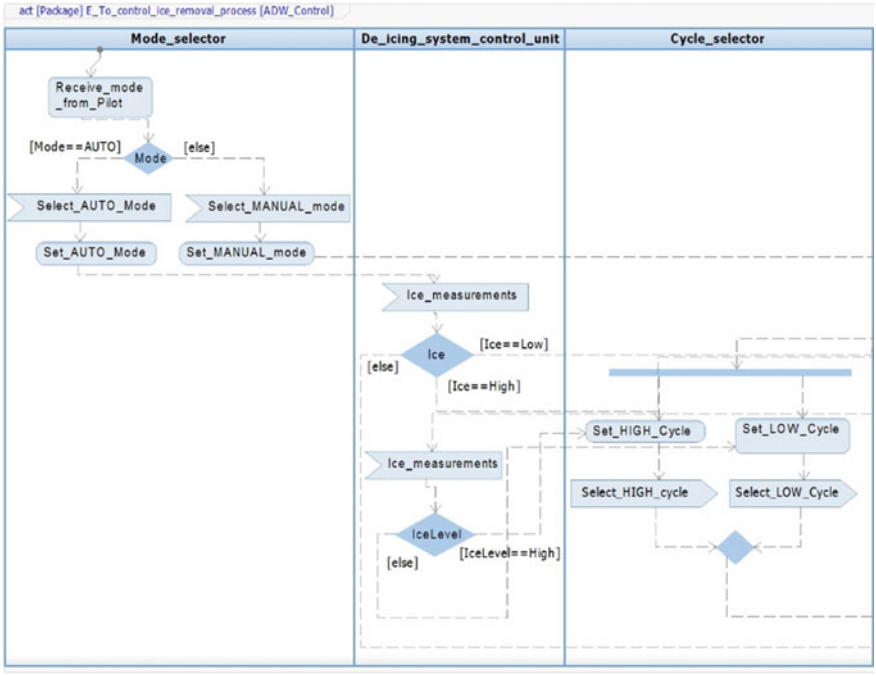


Fig. 7.14 AD for the control use case with swimlanes

The system control in both the solutions is performed through some similar operations. This situation makes the related use case poorly sensitive to the technology applied. Figure 7.14 shows a portion of the AD for the proposed use case. The control action is implemented through a dedicated mode selector, which determines the type of operation (either manual or automatic), and a cycle selector, which sets the intensity of the de-icing operation. The control unit is responsible to evaluate the ice accretion severity. In case of automatic mode, it tunes the de-icing process accordingly.

The monitor use case has a more complex AD. Multiple information must be provided to the flight deck. It is difficult to represent the diagram with the swimlanes, without incurring in some reading problems. Nevertheless, the monitoring operation looks like the actuation of both the electro-thermal and pneumatic systems. The main switch controls the operation of panels and indicators. A new feature is allocated to this component, notably, the *stand–by mode*. In this mode, the de-icing process is switched off, but the ice measurements and forecast are still performed. The mode selector can be used to choose either the manual or automatic mode, but is disabled in case of system off signal. Aural and visual indicators are provided for the ice level and forecast warnings, as far as forecast is concerned, only a visual warning is implemented. Considering the manual mode, a zone selector is also introduced with

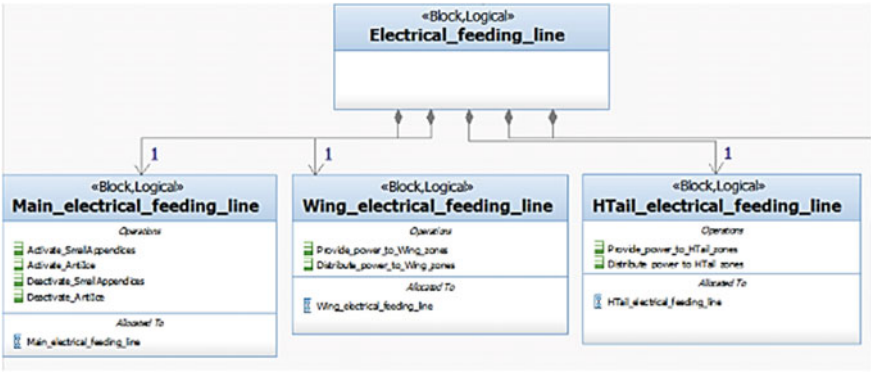


Fig. 7.15 Detail of the BDD for the electrical IPS with feeding lines

the dedicated panel. Cycle selector is even present and the control unit is always computing the data received by the different components.

Some items typical of the control use case are here included, since the panel with its selectors and switches is usually also a typical input/output user interface. Therefore, the control function is related to the input provided by the user, while the monitoring function is embedded in the output shown by the panel through a light or simply the position of a knob indicator.

This allocation process, based on some AD enriched with the swimlanes, leads to describe clearly the logical structure of the two system solutions. The BDD can be then used to summarize the breakdown of components, as is depicted in following figures. Particularly, Fig. 7.15 shows the details of some of the feeding lines. Each block shows in its compartments the data related to the operations allocated and the reference to the swimlane, from which the allocation derives. The

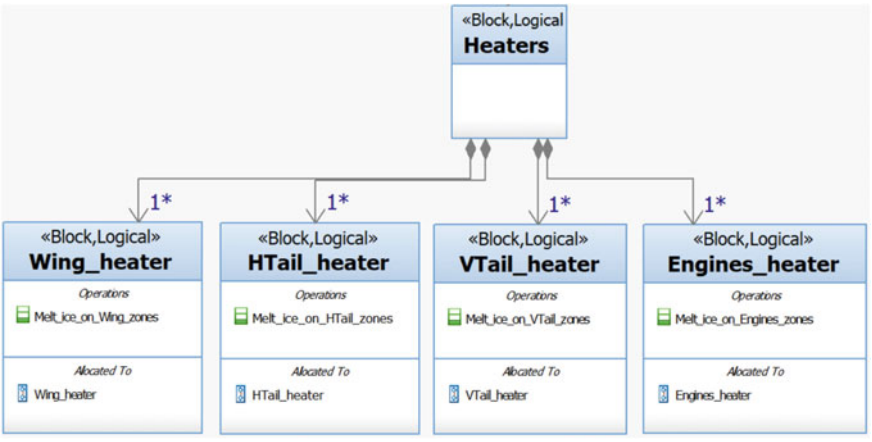


Fig. 7.16 Detail of the BDD for the electrical IPS describing the heaters

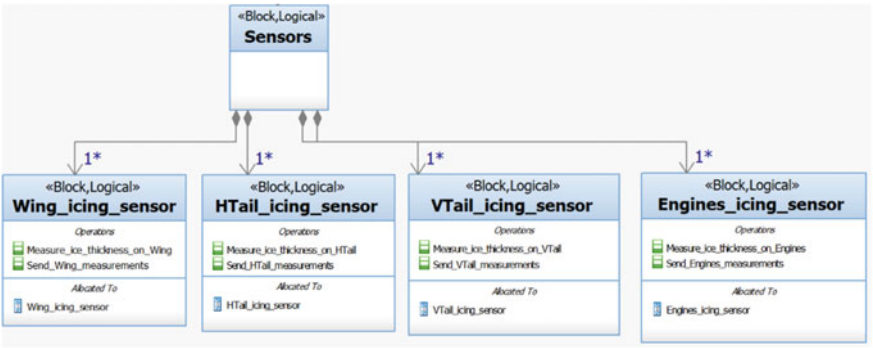


Fig. 7.17 Detail of the BDD for the electrical IPS describing the sensors

same happens for heaters, sensors, controls and indicators group of blocks (Figs. 7.16, 7.17 and 7.18). For heaters and sensors, the *multiplicity* property of *directed compositions* is used to specify that it is possible to consider a multiple selection for the final system, i.e. a certain number of sensors and heaters could be necessary to cover the whole set of zones, although the number of components is not yet defined.

For the indicators, a unique display is used, since the information regarding the operation of the IPS can be embedded within the centralized monitoring system of the aircraft, despite the number of controls doubled in the flight deck. The control panels of on-board systems are even unique in the cabin.

A different system breakdown is shown in following figures, in case of the pneumatic IPS. The allocation performed is similar and the blocks derived from the decomposition of the system are comparable, but the feeding lines and the actuators are different. To realize just these differences, only the distribution and boots subsystems are shown in Figs. 7.19 and 7.20.

As far as the breakdown-based Functional Analysis is concerned, a logical breakdown can be defined directly without performing the allocation of operations. The functions previously defined are already a representation of some sets of operations, defined within the Functional Analysis. Therefore, it is possible to connect functional and logical blocks, without any intermediate step.

The electro-thermal IPS shows a different breakdown to avoid misunderstandings with the previous method. Moreover, the Logical Analysis following a breakdown-based Functional Analysis is usually oriented to define more specifically the system components, almost close to the physical ones. It follows the Functional Analysis where some features of the Logical Analysis have been already performed, such as the definition of the groups of operations. This kind of Logical Analysis is a little bit closer to the Physical one and somehow appears redundant for this approach. A direct correlation with the physical components may be more convenient to define faster the real system. By converse, a behavior-based Functional Analysis leads to a Logical Analysis requiring a preliminary allocation

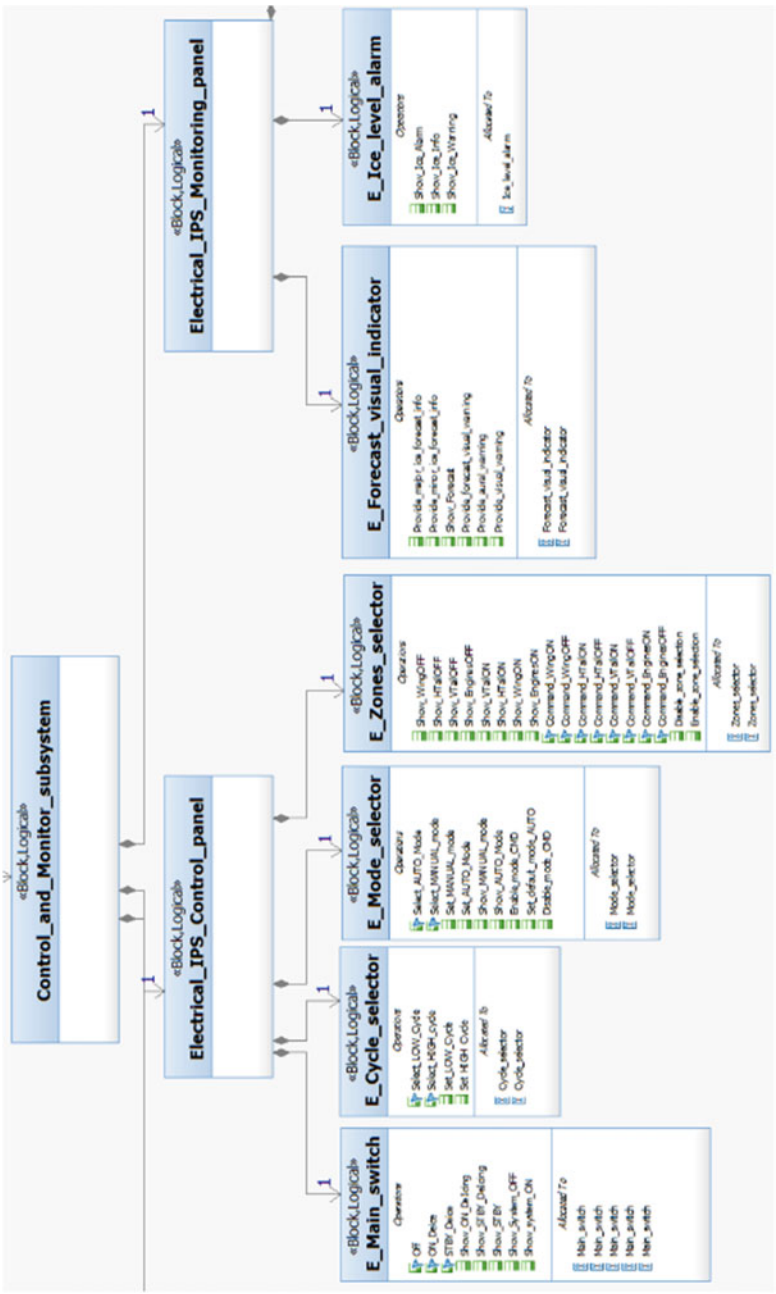


Fig. 7.18 Detail of the BDD for the electrical IPS describing the control and monitoring

Fig. 7.19 Detail of the BDD for the pneumatic IPS describing the distribution line

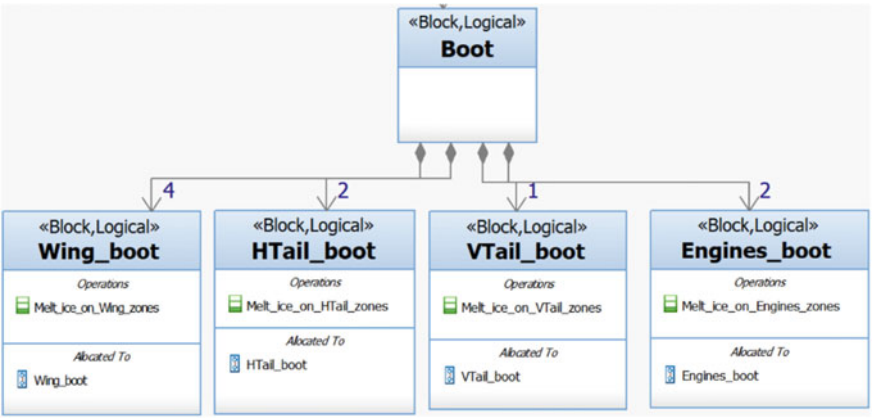


Fig. 7.20 Detail of the BDD for the pneumatic IPS describing the actuators

of operations, before that a physical system layout is defined. An example of this breakdown, applied to the electro-thermal IPS, is shown in Fig. 7.21, for a portion of the control subsystem, where the allocations can be appreciated inside the blocks compartments.

In both cases, the allocation process can be also summarized using table and matrix views, as it was done for the Functional Analysis. The IBDs can be built to show the internal structure of the system, being necessary to implement the communication network and to describe the way in which the different parts are interfacing with each other. Moving from the final components to the system blocks, through a bottom-up approach, it is possible to derive the inputs required and the outputs provided by the system and how they are managed internally. The result is the definition of these data for each block, to enhance a further detailed definition of the system behavior, through the Physical Analysis. The high level IBD for the electro-thermal solution is therefore shown in Fig. 7.22, starting from the BDD developed within the Logical Analysis of a behavior-based approach.

The two subsystems are connected through some signals, related to commands and control (from controls to actuation) and through some feedbacks and system status (i.e. power consumption, ice levels etc.…, from actuation to control and indicators). Some of those signals move out of the boundary of the system, since they are interfaced with the whole aircraft. Figure 7.23 shows the detail of the control subsystem.

A similar set of IBDs can be drawn for the pneumatic IPS. In Fig. 7.24 the high level IBD is shown.

A structure comparable to previous one is depicted, although the physical parameters of the actuation subsystem are different, because of the applied technology. It is interesting to observe the lower level blocks, describing for instance the boots, to realize the parameters which are provided (Fig. 7.25). Each boot is characterized by a volume, which varies during the actuation, determined by the airflow coming from the pneumatic system, at a certain delivery pressure.

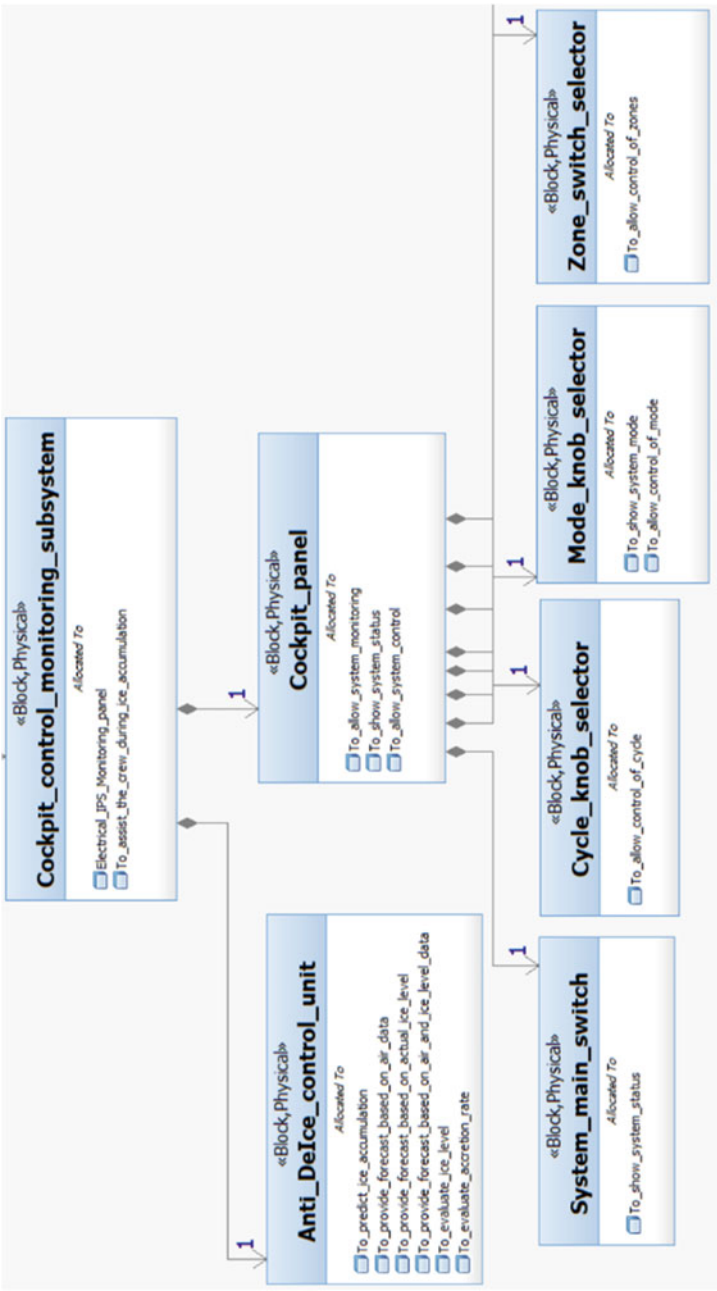


Fig. 7.21 Detail of the BDD for the electrical IPS derived through breakdown-based approach (control subsystem)

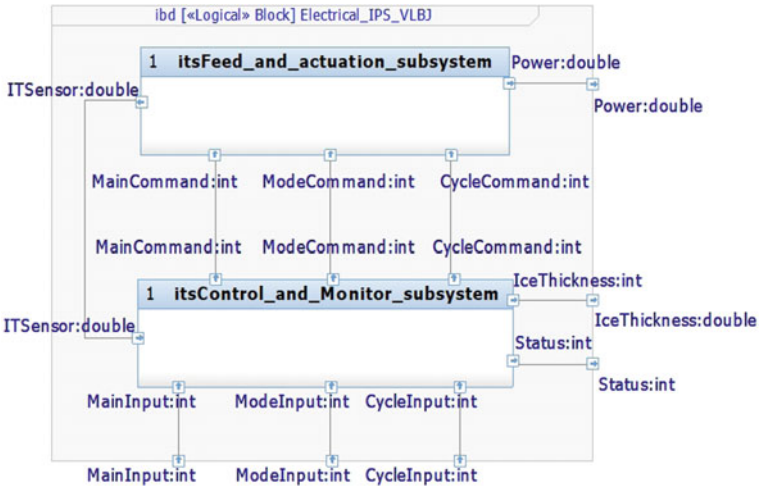


Fig. 7.22 High level IBD for the electrical IPS

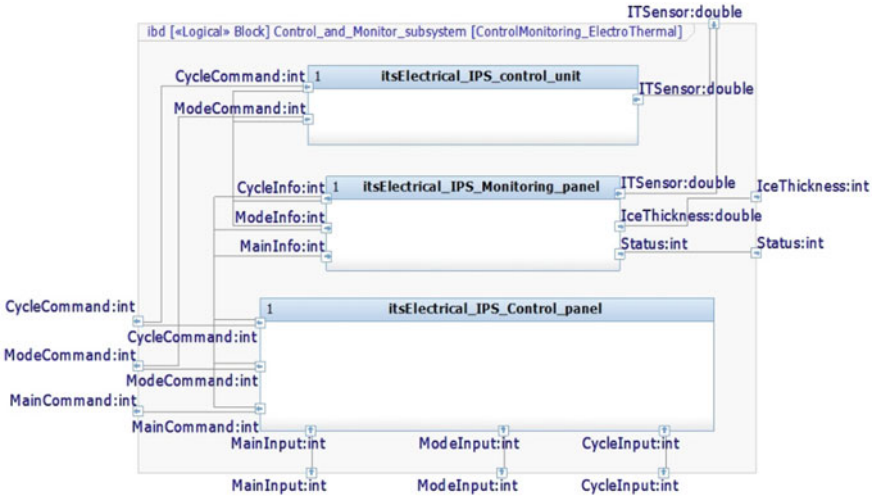


Fig. 7.23 IBD of the control subsystem for the electrical IPS

The power required to inflate or deflate the boot is a combination of those parameters. It shall be assessed only during the dynamic simulation of the Physical Analysis. So far, it might be remarked that some physical characteristics of the two solutions are not yet clear, at present. This uncertainty is never a mistake, but just the consequence of the structured process applied. According to it, the Logical Analysis shapes the system structure, but it does not predict its real behavior

However, some hypothesis can be formulated to start a preliminary trade study, to be concluded, later within the Physical Analysis. The BDDs related to the

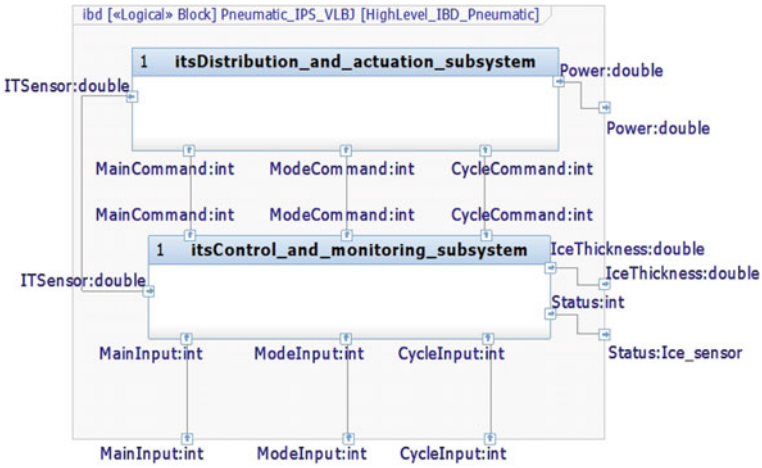


Fig. 7.24 High level IBD for the pneumatic IPS

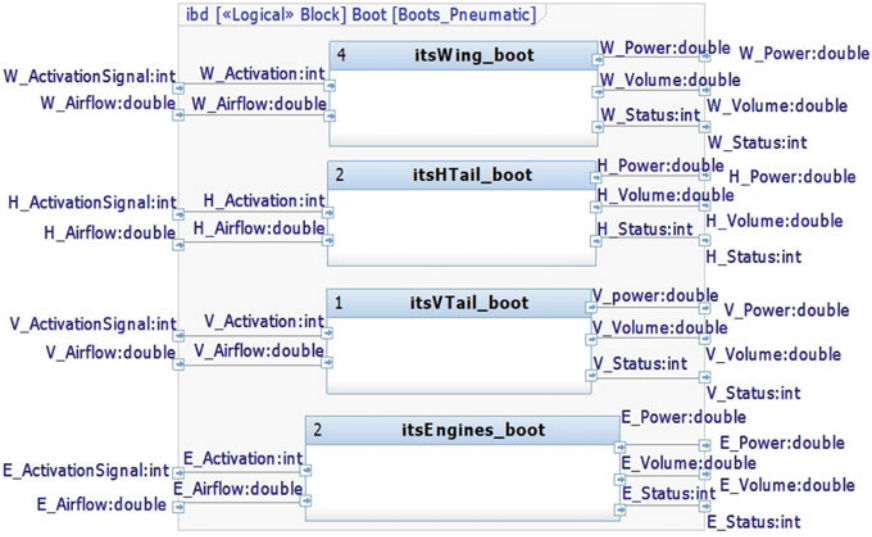


Fig. 7.25 Actuation subsystem IBD for the pneumatic IPS

different solutions proposed are used to formulate some suitable *attributes* of the system blocks. Those attributes will be used to detect advantages and disadvantages of each solution. They may refer to some physical parameters, like dimensions, weight and volume, or even to energy and power consumption. Other issues can be considered, as the operational capabilities, maintenance and cost. For the IPS, the attributes described in Table 7.1 have been considered.

Table 7.1 Attributes considered for preliminary trade study of IPS alternatives

Attribute	Description
Dimensions	This attribute refers to dimensions of the overall system, which has an impact on the available space on the aircraft
Weight	This parameter represents the weight of the system (in kg)
Power consumption	This attribute refers to the maximum power required in operation (in W)
Aerodynamic impact	This parameter is referred to the impact on aircraft aerodynamics when the system is operating
Installation	This attribute refers to the complexity of installation of the different parts of the system
Reliability	This attribute concerns the reliability of the system and of its components
Maintainability	The maintainability is the capability of being subjected to maintenance activities. This parameter refers to the complexity of the maintenance tasks
Cost	The attribute related to cost considers the whole lifecycle cost of the system (i.e. acquisition cost, operating cost and disposal cost)

Attributes can be embedded within the blocks compartments of the BDD, but some dedicated formalisms to represent the trade study can be exploited. A *generalization* can be used to represent the alternative solutions of the IPS, as is shown in Fig. 7.26.

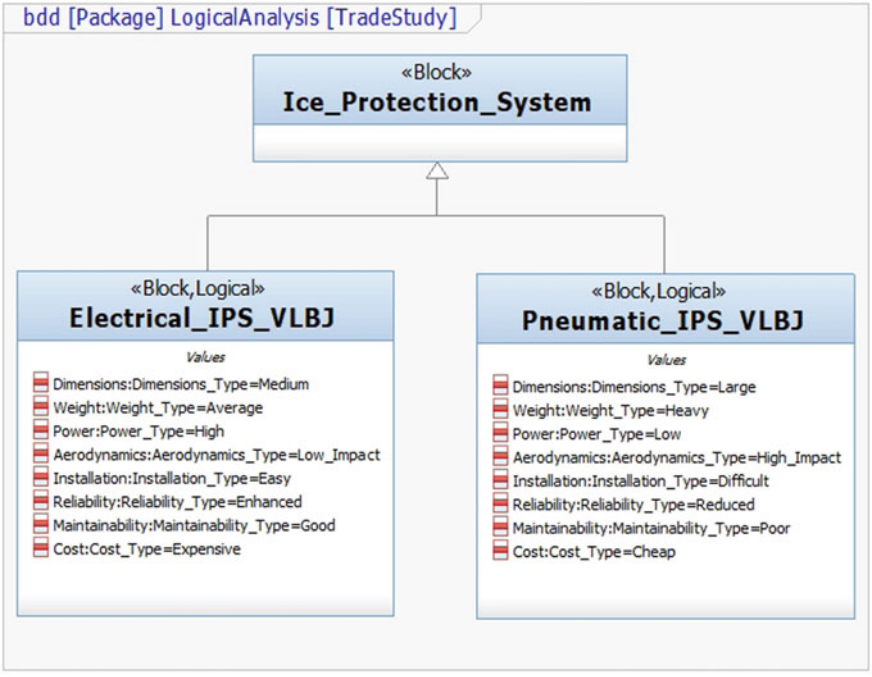


Fig. 7.26 Trade study formalization with BDD for the IPS case study

In this case, it is easier performing the comparison directly on a single diagram, like a dedicated BDD. As a result, it is possible to realize that the power consumption of the electro-thermal system is very high, if compared to that of the pneumatic system, as well as the acquisition cost, but many other parameters are better. Particularly, the dimensions and weight are lower as well as the impact on aerodynamics. The installation is easier and, since the boots are subjected to a high degradation, the reliability is lower for the pneumatic system. The maintainability of the pneumatic system is more difficult, because of a restricted accessibility to boots. The electro-thermal solution seems more promising, also considering the customer need about the reduction of fuel consumption and the decoupling of the deicing system from the pneumatic one.

However, the Logical Analysis provides only a preliminary overview, and those conclusions shall be verified by the Physical Analysis. A current goal of this example is showing a suitable procedure to formulate the KPI, although it is not unique.

7.7 Requirements Traceability in the Logical Analysis

The requirements traceability process usually relies either on visual diagrams or matrix views of links instantiated between requirements and logical blocks. In this case, no differences are present between the behavior-based and breakdown-based process, since the logical blocks are linked to requirements in both the approaches. The traceability links are here strong relations between entities, since the *satisfaction dependencies* are used, which are definitely stronger than the *trace* link used in the Functional Analysis. The process can be done in parallel for the two solutions, to make easier the comparison between the two coverage analyses. Different systems may have different satisfaction relations. It depends on how the system is composed, since different components should satisfy also different requirements.

In case of the IPS similar satisfaction relationships are instantiated, although the proposed solutions are based on two different technologies. The system architecture is quite similar. Figures 7.27 and 7.28 show some diagram and matrix views, concerning the requirements satisfaction for the components of the electro-thermal system.

The result of the requirements satisfaction process is a complete coverage of functional requirements, as shown by the Gateway in Fig. 7.29. Non-functional requirements will be analyzed during the Physical Analysis, and in other analyses like those performed to predict the system reliability.

It is now possible to perform a complete navigation of the model elements. Particularly, a consistency check of the completeness of data analyzed can be done. In practice, it is possible identifying the data flow from requirements to logical analysis, in terms of requirements *traceability* and *allocation* from the use cases to the logical blocks, as in Fig. 7.30.

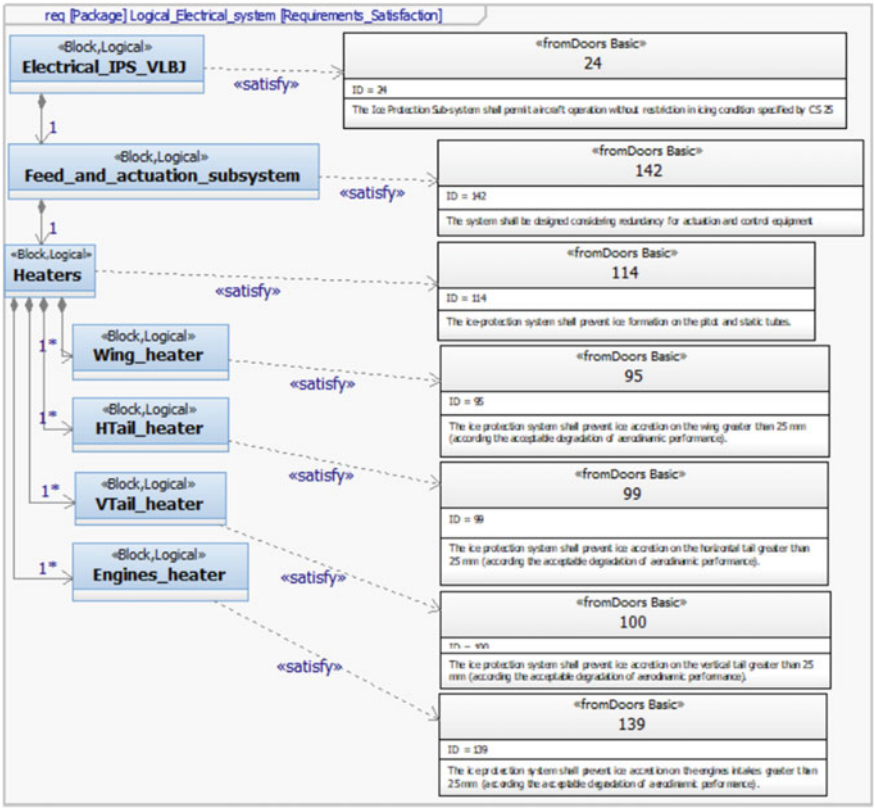


Fig. 7.27 Requirements satisfaction instantiated within a Requirement Diagram in the Logical Analysis of the IPS

	70	118	119	120	121	122	123	124	125	126	133	132
Electrical_IPS_VLBJ											✓ 133	
Feed_and_actuation_subsystem												
Electrical_feeding_line												
Main_electrical_feeding_line												
Wing_electrical_feeding_line	✓ 70			✓ 120								
HTail_electrical_feeding_line	✓ 70				✓ 121							
VTail_electrical_feeding_line	✓ 70				✓ 121							
Engines_electrical_feeding_line	✓ 70											
Heaters												
Wing_heater	✓ 70					✓ 122						
HTail_heater	✓ 70					✓ 122						
VTail_heater	✓ 70					✓ 122						
Engines_heater	✓ 70					✓ 122						
Sensors												
Wing_icing_sensor												
HTail_icing_sensor												
VTail_icing_sensor												
Engines_icing_sensor												
Control_and_Monitor_subsystem							✓ 123					
Electrical_IPS_control_unit								✓ 124	✓ 125			✓ 132
Electrical_IPS_Control_panel								✓ 124	✓ 125			

Fig. 7.28 Requirements satisfaction instantiated within a matrix view in the Logical Analysis

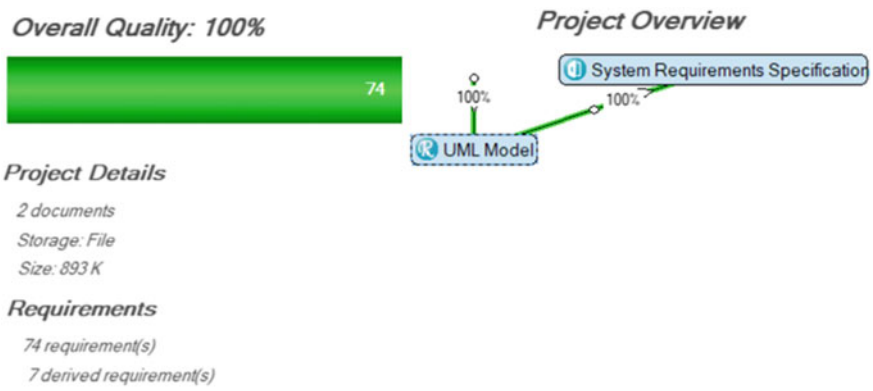


Fig. 7.29 Analysis of coverage quality through the IBM Rhapsody® Gateway® within the Logical Analysis

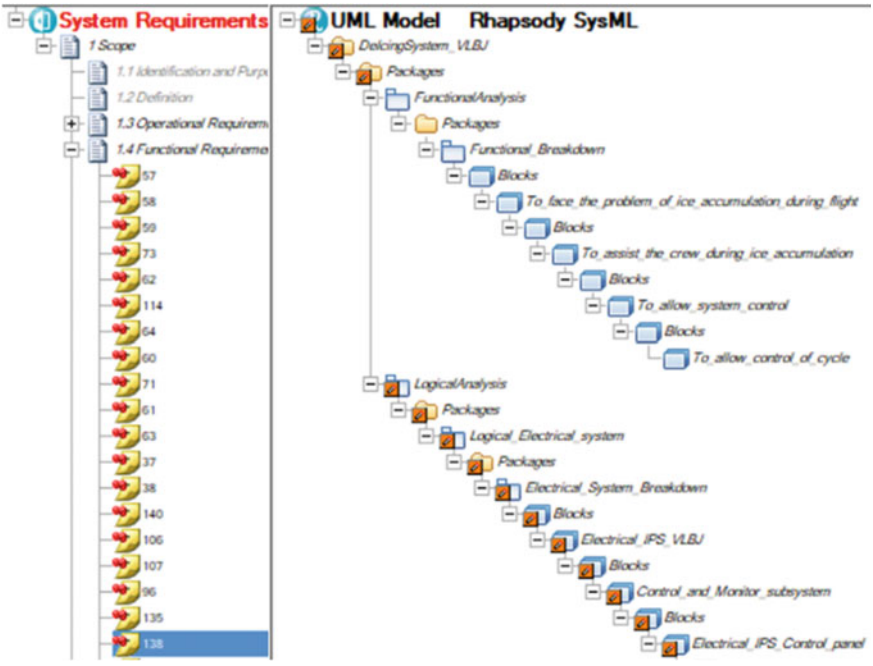


Fig. 7.30 Traceability of requirements, functions and logical components through the IBM Rhapsody® Gateway®

This activity completes the functional modelling process and the logical analysis, whose results are summarized in the next paragraph.

7.8 Results and Final Considerations About the Logical Analysis

The Logical Analysis allows defining a first set of candidate system architectures to prepare the following studies concerning the detail design. Provided that the same MBSE approach is applied, the implementation depends on the way followed in the Functional Analysis. A main result of this analysis concerns the *allocation* process, being responsible of identifying one or more suitable solutions, capable of performing the functions derived during the previous step. In addition, it includes the final stage of the requirements traceability study, as far as the *satisfaction dependencies* are considered. Finally, the formalization of the *trade–off study* is also another important issue. It is aimed to define the candidate solutions and their weighed properties.

Some main steps and basic features of the Logical Analysis, above described for the test cases are herein summarized.

- The Logical Analysis starts from the functions or *operations* derived during the Functional Analysis. They are allocated on logical blocks either through a direct exploitation of some behavior diagram, like the AD, SD and SMD, if the behavior-based approach is applied, or directly using some breakdowns, implemented within the BDD, when the breakdown-based approach is considered.
- The logical structure of the system is then enriched by some internal interfaces and network, implemented through a series of IBD, that allows defining variables, ports and a suitable proposed architecture.
- The requirements satisfaction is the final step, since the *traceability* process starts during the requirements and operational analyses and is concluded when a system architecture capable of performing the required functionalities is identified.
- A *trade-off study* to select the best candidate solution among those alternatively proposed, which satisfy the functional requirements, is then started. It analyzes some non-functional issues, like the performance, reliability and other physical characteristics. The trade-off study is a formalization of the analysis. It shall be detailed through some dedicated tools, depending upon the engineering domain, to confirm or modify the solutions proposed by the Logical Analysis.

The process above described is often performed iteratively, since several levels of design are considered, from system to component. The examples above proposed are simplified and a single loop of iteration was considered. The number of requirements is even reduced, to simplify the process and the logical architecture of the system. The trade–off study is simplified as well.

The system architecture derived through the IBD shall be re-used as much as possible within the Physical Analysis, to reduce workload and repetitions. For this reason, a tuning process is required, especially when different tools are interfaced within the toolchain. The Physical Analysis basically will show how it is possible

using the data coming from the Logical Analysis to implement an executable dynamic model in a common commercial software tool. Nevertheless, the interoperability of tools is currently an open issue and a direct re-use of the data is seldom possible. A hybrid approach is applied, relying on the *heterogeneous simulation* approach, based on some standard.

Despite the importance attributed by the literature, usually low, the Logical Analysis plays a crucial role in the MBSE process. It is the connection point between the SE and the classical tools of engineering modeling and simulation. Moreover, numerical and physical modeling could be even more effective, since all the issues poorly represented by numbers are included and formalized by this approach. The iteration process which allows refining the system configuration on the base of the numerical simulation is identified and the two environments of the functional and physical modeling can be synchronized, through this toolchain and in both directions. When the Logical Analysis is considered as an option, and the Physical Analysis is assumed to be the unique core activity of the design process, something is missed. The traceability assurance, usually neglected by the numerical simulation, is checked by the Logical Analysis.

Chapter 8
Physical Analysis

Abstract This main activity of engineering is described and analyzed within the frame of the whole product development and in connection with the Functional and Logical Analysis. The two test cases are explored to describe different examples of physical modeling, considering the dynamic behavior of a rotor, actively suspended and the performance in de-icing activity of the on-board system of the commercial aircraft selected for this example. Some numerical methods are even described as they are applied in the two proposed test cases.

8.1 Introduction

The Physical Analysis is a main activity usually performed within engineering, since it is related to the performance evaluation, dynamic and structural simulations, finite elements analysis, CAD modeling and similar. It includes all the possible tasks related to numerical modeling, in different fields, and it was born to reduce the cost related to testing on real components and systems, substituting the real artifacts with numerical models. It is applied for sizing the system and to predict its behavior in operation. It identifies limits and allows characterizing the real system.

Since the origins of technical computing, the physical analysis supported the model and simulation based approach to design. It strongly relies on a virtual representation of reality, under a set of assumptions, aimed at approximating the real phenomena. Each field of application resorts to some specific mathematical formalisms and equations, to implement the modeling process. Some criteria are preliminary defined to check whether the system description proposed by the numerical model is consistent or not. A specific set of commercial software supports this activity; therefore, results and simulations of the Physical Analysis undergo an organic debugging, which easily detects any inconsistencies or mistakes introduced inside models.

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_8

All these aspects make the Physical Analysis a perfect engineering task, being characterized by:

- velocity, especially when considering the ratio of provided results per unit time, if compared to real-life tests;
- approximation of physical laws;
- introduction of some specific hypotheses to make easier the derivation of final results;
- standardization, concerning both the formalisms and the data exchange, among different modeling environments of the scientific community;
- assisted implementation process, which simplifies the detection of errors and bugs, within the model to accelerate the review;
- model-based representation of reality, as a main target of the design process, especially for complex systems;
- reduced cost, since it is cheaper than a set of test campaigns.

However, the Physical Analysis very seldom provides a set of alternative solutions to be tested, without following a formalized and standardized Systems Engineering approach. Numerical simulation is usually applied to some defined system architectures. The risk of neglecting any important variant, which may be better in terms of performance, is high.

The evolution of this activity is a suitable inclusion of the Physical Analysis within a complete MBSE process, to enhance its capabilities and to allow a consistent and seamless connection with previous analyses. This improvement could assure the *traceability* of requirements, derived through the operational and functional analyses, and the *allocation* process, which is typically finalized within the logical analysis. Therefore, the Physical Analysis is herein proposed as a final step of the design process, with the aim of studying even some non-functional aspects which characterize the system. Sizing and dynamic behavior prediction are even tasks of the examples herein introduced.

8.2 Handoff Between Logical and Physical Analyses

A proper handoff is required to assure the data consistency, as it was already done for the functional and operational analyses, for the physical analysis, which follows the Logical one. In this case, the focus is slightly moving from the traceability to the re-usability of data in multiple environments. A seamless toolchain shall be guaranteed in terms of data consistency and traceability, but the compatibility of diagrams created through the SysML formalisms and the format of numerical models are both critical issues. The Physical Analysis is usually performed within different environments, often exhibiting some non-negligible problems concerning their mutual compatibility. Moreover, it is necessary to maintain the link with the Logical Analysis active, since updates require the back synchronization of models.

Some features of the Logical Analysis shall be modified to be used for detailed numerical analyses. Modifications are related to properties of data objects, so extensive updates are not required.

Some interoperability issues are always a concern, when multiple software environments are considered, especially when they are not originally planned to communicate each other. Different means to assure the data compatibility are available, depending on the application, but, for the purposes of this chapter, a direct compatibility of the Systems Engineering and dynamic simulation tools will be assured, i.e. only minor modifications to the data object to be exchanged between environments should be required.

In interoperability cannot be ever guaranteed and other methods are applied to solve some compatibility problems. The interoperability standards are exploited, when available, or general purpose import/export facilities are introduced. The handoff between the two design phases depends on the way applied to share the model; particularly, the capabilities of tools involved allow a complete exchange of the full architecture or even just of some parts of it. As far as a complex toolchain is concerned, i.e. more than two software, a data exchange based on multiple levels of interfaces among software can be accounted. The capabilities of one tool, concerning the compatibility, can be exploited to link another one. This subject will be deeply described in Chap. 9, where some strategies of either direct or indirect data exchange will be addressed, as well as the so-called *heterogeneous simulation*.

In this chapter, test cases are developed using the dedicated import/export facilities, which allow the data exchange independently on the software used. Tuning of data coming from the Logical Analysis is investigated. Problems related to the direct exchange of data while connecting the Physical Analysis to the MBSE design process are explored.

8.3 Formalisms and Models of the Physical Analysis

Provided that it is impractical introducing the whole set of existing numerical or physical models, in a single paragraph, and a dedicated book should be written, some characteristics of tools and some methods available to perform the Physical Analysis will be here described. The engineering domain affects quite a lot the way of building the numerical model. Many classifications are proposed for this task, depending on the goal of models, of their main characteristics, on the assumptions applied. Dynamic and static models, continuous and discrete, parametric and geometric, simplified and complete, are some typical examples.

In this handbook, the need of interfacing the Physical and Logical analyses is considered and the necessity of introducing a model based on some physical attributes, somehow providing a new kind of information, is faced. In addition, it is required of characterizing the physical nature of the model itself. Two examples will be assessed, namely the *sizing model* and the *dynamic model*. This selection is also aimed at clarifying some differences existing between those two models, since

they are often used indifferently, although they represent different aspects of the physical analysis.

The sizing model. It is commonly used during the preliminary design phase. It is aimed at identifying some specific characteristics and parameters of the system, which are exploited to define some limits of performance or a physical envelope, from which the class of system is derived. Power, stroke, heating and other physical quantities are objects of the sizing activity. Computation is quite straightforward, because some simple equations are written and solved to calculate those design parameters. These models are implemented, under a set of hypotheses about the operating conditions, and other boundary conditions, being necessary to find the values of those parameters. For this reason, sizing models are referred to as *parametric*, i.e. few main variables can be changed to modify the mathematical problem, when needed. Typical sizing models are time invariant or at least they do not include any prediction of the system dynamic behavior. Only a preliminary sizing activity is performed, i.e. the class of the system for a given characteristic is determined, by identifying the amplitude of some parameters for given operating conditions. These models are numerically solved by some mathematical codes, which exploit functions expressed within formalisms created for several specific engineering domains. This is the case of programming languages as the Fortran, or of some general purpose calculation environment as the Matlab® and others. Results are shown in charts or described in reports, which help in identifying any critical condition and in highlighting the most impacting parameters.

In aeronautics, for instance, main sizing parameters are the power, energy consumption, flow rates of fluids, volume and weight, forces, moments, inertia, autonomy.

The dynamic model. This model predicts the system behavior in operation, with a degree of approximation which strictly depends on the available knowledge about the system under design. It is used in many design phases, but specifically it is exploited for a quantitative investigation once that the system is defined and for the verification and validation. These models are strongly time dependent, and often are based upon some differential equations, which need to be numerically solved. The operating conditions are very often simulated by a dynamic model of the scenario or environment, in which the system shall work. Like the sizing models, they are implemented within high level mathematical codes. They can be either in combination with the sizing models or separately. When the geometry is modelled by the sizing model, it is imported or used by the dynamic model to perform the numerical analysis. The *parametric* form makes easier any required modification and updating. Results are represented as variables, being usually a function of time. In many cases, connection with CAD models is important to derive the geometry of the components. Exporting data is then very useful, ad some standard formats are used to exchange data among different environments and tools.

System dynamics is used for several purposes, as the assessment of control laws, investigating the system dynamic stability in operation, to prevent structural damage, and analyzing several coupling effects, like the thermomechanical or electromechanical ones, or the interaction with fluids.

Physical models can be preliminary classified within those two groups, although each technical domain resorts to some specific analysis, leading to a definition of many other sub-categories.

The test cases herein studied widely exploit sizing and dynamic models. The design of aeronautical on-board system relies on sizing models rather than on dynamic models, more used for the whole flight mechanics, whilst the rotor dynamics prediction is strongly based on dynamic models.

8.4 Requirements Allocation and Verification

The role of requirements during the Physical Analysis should be preliminary defined, especially in relation with the traceability process instantiated within the MBSE approach.

Models developed within the Physical Analysis are exploited for both the SRS update, for a back synchronization of non-functional requirements, and for the requirements verification. They may provide new requirements, depending on the results of analysis, or update some existing ones. They are even a reliable verification mean. As an expression of numerical modeling, they can be used straightforward to verify the system performance and some physical requirements previously defined. Verification activity can be either manual or automatic. The requirements verification process is nowadays an available option within commercial software dedicated to dynamic simulation.

A key issue to assure a complete seamless approach, by the Physical Analysis, is the consistency with requirements allocation of the Logical Analysis. During the last phase, all requirements have been allocated on some logical blocks through *satisfaction dependencies*. Functional requirements can be considered verified, through this kind of relationships, while for non-functional requirements this kind of verification is more problematic. To be consistent with the whole process, also the elements of physical models shall be linked to the SRS, to assure the formalization of the verification process, allowing also a consistency check of the selected requirements.

The inclusion of requirements within physical models is a strategic approach to keep the traceability under control, by creating a single self-contained artifact of physical blocks and requirements, directly referenced to the SRS. Nevertheless, to achieve this goal, the compatibility between simulation software and requirements manager must be assured. This is a matter of interoperability of tools, which adds to the need of data exchange of the system architecture, from the logical analysis. If the compatibility is not achievable, the requirements manager and the Functional Analysis tools can be used to provide traceability hubs separated from the physical model, which remains the real verification tool.

In the examples, since most of the physical tools are compatible with the requirements manager, the integral consistency check of requirements is assured. The requirements analysis is now complete, after that the Physical Analysis is

implemented. The requirements verification is a complex procedure. Several studies are currently performed about this topic. Chapter 10 will introduce some more details about it, although a complete description of the verification and validation process is out of goals of this handbook.

8.5 Expected Results and Final Remarks

Main objectives of the Physical Analysis are a determination of the system behavior and the identification of the main design parameters, which affect its performance. Results are expressed by several models and formalisms, basically through some numerical values and charts. They are used to update and verify the non-functional requirements and to complete the MBSE process, with a domain-oriented approach, strictly mathematical. Results are derived from sizing and dynamic models, expressively built for this analysis and depending on each case study. Compatibility of these models with the Logical Analysis and previous stages shall be guaranteed to assure a seamless toolchain.

It might be pointed out that the Physical Analysis always played a key role in engineering, although it is known as a numerical modeling. The quantitative approach defined by the simulation clearly states some results and predictions about the system performance. It may be said that it cannot be skipped. It is true that coupling the numerical modeling with the MBSE process allows enhancing furthermore the powerfulness of these tools. Quantitative analyses become formalized and organized in tight connection with requirements. As this handbook describes, the design process shall benefit of the correlation of functional and physical worlds, being not only recommended, but somehow required. Particularly, outputs of the Functional Analysis are incomplete, without a numerical simulation, as well as in engineering complex system the physical analyses need the inputs provided by the functional modeling to completely characterize the system, through a linear traceability process. When dealing with complex systems, testing many physical models, performing a sort trial and error exploration is not enough. A logical process to identify some eligible candidates can drive the physical analysis to perform a consistent *trade off*.

The sequence of analyses related to the MBSE process, described in these chapters, actually deploys the "system thinking", whose goal is an objective—oriented definition of a system, through a sequence of well-defined development phases, characterized by processes and goals, and aimed at facing the design problem in all its parts. Collaboration among system developers is behind this approach. It is no longer acceptable working as separated units, neglecting the big picture. Systems Engineering enables to work within the frame of a collaborative network, along a defined path, with a tight cooperation of all the actors, dealing with different analyses and activities of the whole product lifecycle development, to realize a truly successful system. In this interpretation, the connection between physical and functional modeling is a key issue, for the methodology and for the tools.

8.6 Implementation of the Physical Analysis of Complex Systems

8.6.1 Didactic Test Case

As it was already introduced in previous chapters, the so-called didactic test case describes the design of a system aimed at shaping the steel wire rod produced by a rolling mill as a straight bar into a coil to be easily stored and delivered. The requirements and functional analyses already pointed out some relevant functions and constraints of this system, but the final product breakdown structures needs a preliminary screening of the state-of-the-art of the technologies suitable for this application as well as a preliminary definition of some proposed architectures, to be compared through a trade-off analysis, leading to the selection of the best solution.

The idea. It might be remarked that the basic idea of a new product could be simply found everywhere by observing the nature, the environment or other technical systems, not only those already used in the inherent technical domain. As an example a tourist walking in Paris just close to the Centre Pompidou could realize how the spoiler system may work, just by looking at the water jets of the fountains used to create some amazing figures (Fig. 8.1). The main water jet comes from the bottom as a straight flow and the fountain describes a spiral of water by rotating about its vertical axis, while a pipe imposes a defined angle between the inlet at the water level and the outlet located in correspondence of the big mouth. This basic idea turns out into a preliminary layout of the system, as described in Fig. 8.2.

In practice, a spoiler will include a main rotating shaft and a head. The whole system shall compose the laying head. The rod provided from the production line inside the holed shaft will be driven by rotation and centrifugal forces will be applied inside a curve guideline made of a pipe which leads to form the coil as the

Fig. 8.1 Water jet at Centre Pompidou in Paris

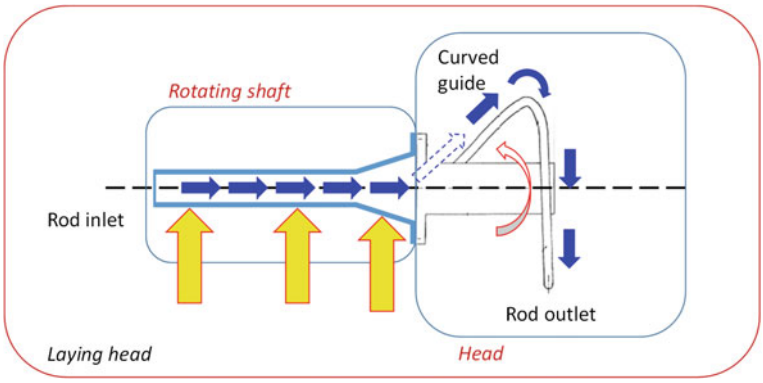


Fig. 8.2 A basic sketch of a coiler system

wire rod is produced out. Some details might change case by case, as the shape of shaft, or the structure of the head, or even the fact that those two elements are either separated (simply connected by bolted joints) or a unique structure. Actually the main issue in innovation is the suspension system and the motor animating the rotation.

From this point of view it is required that the designer performs a preliminary screening of the existing technologies in suspension systems.

The technological scouting. A screening of the existing suspension technologies might suggest that several solutions could be applied to the test case. To distinguish very roughly some approaches, suspension could be *passive* (no external power is needed), *semi-active* (there is a control component which needs an external feeding of power, while a main component suspends the floating body without requiring external power), *active* (the whole system needs power to be operated).

In case of mechanical suspension systems, purely passive, some classic solutions consist of mechanical bearings (ball, needle and rollers), bushings, and lubricated bearings. All those solutions introduce a direct contact between the rotating part (namely the rotor) and the fixed frame of the system (the so-called stator). In case of mechanical bearings with rolling elements the contact is between the rolling part (ball, needle or roller) and the raceways of the rings, being applied one on the rotor and one on the stator, respectively. Lubricated bearings provide a lift to the rotor through the fluid in pressure inserted between the rings, while bushings simply are in contact together with both the structures, but material is self-lubricated and prone to be worn quite fast. In present case, those solutions looked unsuitable because of the customer need of interrupting the contact between stator and rotor and reducing the wear of materials.

Air bearings are even known and applied, but the weight and the actions of the rotor might be incompatible with this application. Contactless or partially contactless solutions were highly looked for. In this case, active and semi-active bearings could be analyzed as well as passive contactless suspensions.

If one analyzes the last type, homopolar magnetic bearings based on the repulsion of some permanent magnets could be applied. Basically, some radial and axial actions are provided by a distribution of permanent magnets on the lateral surface of the rotating shaft, in correspondence of the bearing, and on the stator, inside its housing. However, this solution does not allow calibrating in operation the amount of action applied, nor to assure a complete dynamic stability of the suspended system, since Earnshaw demonstrated that at least one axis should be actively controlled to assure a stable positioning of the shaft (Brusa, 2016).

Magnetic coupling could be better exploited by resorting to diamagnetic materials, which provide a stable levitation, even if, unfortunately, they are typically superconductive materials, which need an equipment to reduce the temperature to operate effectively the suspension system. In this context this solution is inapplicable.

The active bearings could be based on the actions provided either by the electric or magnetic field. The electrostatic couplings based on the forces exerted by the two opposite electrodes of a capacitor are too weak for the actions required in this test case. Therefore, the electro-dynamic active magnetic bearing (Lorentz's actions) and the active magnetic bearing based on the generation of electromechanical actions through the magnetic field created by some coils (Maxwell's actions) had to be considered. In principle, both those technologies are a suitable candidate for the test case, although it is known that the active magnetic bearings based on the Maxwell's action were already applied in several other industrial systems and some available technical standards appeared more complete to the customer, to be integrated and compared to those governing the design of the whole rolling mill. Moreover, the risk of ungoverned losses and induced currents in case of the Lorentz's actuators looked unsuitable to fit the requirements applied to the whole steelmaking plant, although they are suitable in other technical applications.

Limitations applied by the customer to the presence of lubricants excluded solutions based on electro-rheological or magneto-rheological fluids, as well as those including a structural support to bear the weight of the rotor and an electromechanical actuator just to control the rotor balancing and whirling motion, because of the need of absence of any contact.

For those reasons, after a preliminary screening, the technological solution proposed concerns the active magnetic suspension with radial bearings based on the Maxwell's electromechanical coupling, herein described. Eventually, an axial thrust bearing could be added to prevent the effect of a load applied along the direction of motion of the wire rod.

It might be remarked that in the application of the SysML (or similar) language to the modeling activity, a graphical input or a diagram to link the above mentioned screening to the other diagrams should be helpful, although it wasn't yet standardized. As an example a technology chart like that depicted in Fig. 8.3 could help. It represents some typical key words to describe some different technologies available for each design issue, as the suspension layout, the electromechanical coupling, the energy conversion and materials. Designer might select a preferred path within the proposed solution to draw the BDD and IBD of the MBSE model.

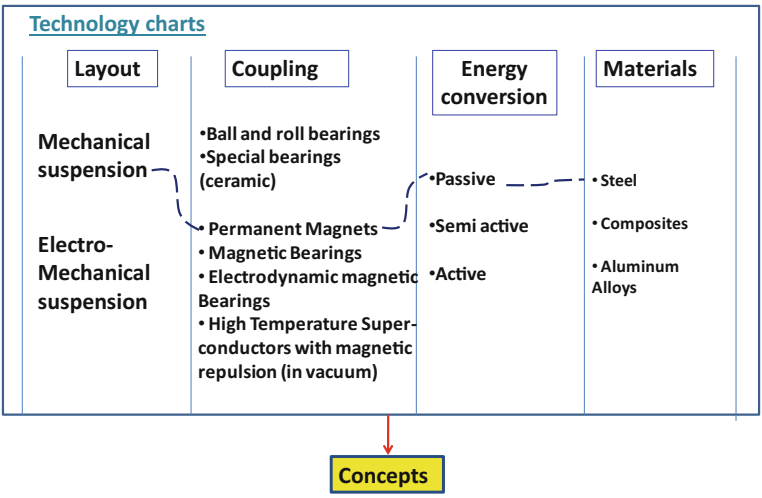


Fig. 8.3 An example of technological screening performed through a chart

The proposed solution. As Fig. 8.2 shows at least two devices are required to support the rotor, and their sizing could be different if one realizes that the weight of the structure is dominant in the region of the rotating axis close to the head. The Maxwell's actuator is composed by some active axis like that described in Fig. 8.4.

The rotor corresponds to the floating body between the two polar expansions of the magnetic circuit. It has a mass *m*. Each polar expansion is equipped with a coil fed by electric current, which generates a magnetic field. If the body is either made or equipped with a material reactive to the magnetic field, i.e. ferromagnetic, an electromechanical action is applied by each coil separately. Currents are fed to the coils in a such a way that two opposite actions are generated, thus opposing to the weight of the floating body or to the dynamic actions generated during the rotation.

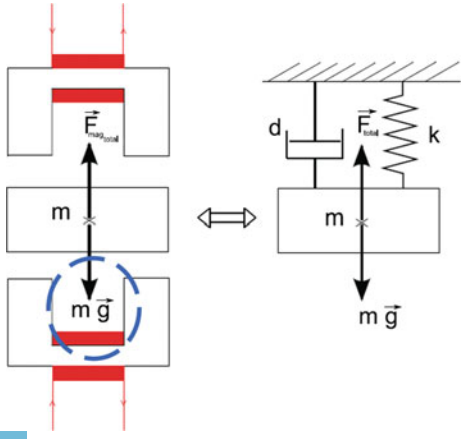


Fig. 8.4 A basic sketch of a single active magnetic axis

The system looks similar to a mechanical suspension being characterized by a stiffness k, relating the displacement to the action applied, and by a damping d, relating the speed of the vibrating mass to the force induced. If the axis is aligned with the weight force it has to suspend the body and to control its vibration in the plane depicted in figure, whilst in case of horizontal direction the weight disappears from the described equilibrium of the above figure. Commercial solutions provide several active axis to allow reducing the flux leakage and distributing the electromechanical action along the circumferential direction of the active bearing, thus leading to several active axes. Four axes (0, 45°, 90°, −45°) are quite common.

To control the rotor dynamic several strategies are implemented. A very simple and classical approach resorts to a PID controller, based on a closed—loop monitoring of the rotor displacement along the controlled direction, i.e. either radial or axial. As a didactic case, this example shall show a quite simple solution, to make easier catching some issues of the physical analysis performed. Therefore, in Fig. 8.5, a sketch of the control applied to a single active axis is shown.

The rotor shaft is the main dynamic system. It is characterized by a mass, m, some inertia, J, a stiffness, k, which might be either associated to the supports only, or separated for the supports and the shaft itself, if it is assumed as a flexible body more than a rigid body in motion. Similarly, there are some kinds of dissipations, which apply a damping action. Those associated to the rotating parts provide the *rotating damping*, while those associated to the fixed parts are the *non-rotating damping*. That definition remarks that the action of a rotating damping is seen rotating together with the shaft, while the stator applies a damping action which is always seen as fixed in direction in the non-rotating reference frame.

Some forces, F, are applied to the rotor, and typically an unbalance condition defined by the eccentricity of the center of mass, ε, exists. Basically, the displacement of the center of mass of the cross section of the rotor in correspondence of the bearing along the monitored direction is measured by a sensor. Let's define as x. This measure is amplified by the sensor through a gain K_s. Then, it is compared to the reference position of the cross section, r, being the target of control. As the difference between the measured position and the reference increases, the error to be controlled grows up as e. It is the input of the PID control, i.e. of a combined action applied directly to this value (proportional), to its rate in time (derivative) or to its integral over time (integral). In some cases, the last one is neglected and only a PD

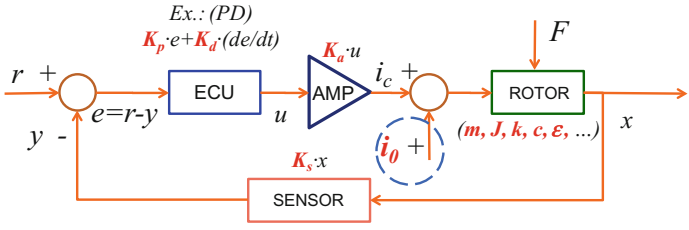


Fig. 8.5 PID control of a single active magnetic axis

control is actually applied. The ECU (Electronic Control Unit) applies the control action, through a set of gains corresponding to each component previously described (K_p, K_d and eventually K_i). A command u is generated and sent to the power amplifier where it is converted from pure signal into a control current i_c, through a gain K_a. The control current is directly fed to the rotor to create the electromechanical action, if no component of the weight has to be compensated, or it is associated to the bias current i_0 expressively added to cope with that need.

In general, that model is replicated for each degree of freedom actually considered in the rotor design. In case of the spoiler system, a couple of radial bearings, operating along two perpendicular radial directions, and one along the axial direction, if required, are foreseen.

The energy conversion and the goals of physical simulation. Once that the ECU provides a command to the power amplifier and the control current is physically produced, by using an external feeding of energy (which motivates the definition of this control as an active configuration), the conversion from electric power into mechanical one is performed through the magnetic field. It is known that the electromechanical action, F, is proportional to the square value of the fed current i and inversely proportional to the square value of the distance between polar expansion and rotor, namely the gap t:

$$F = k_{mag}\left(\frac{i}{t}\right)^2 \tag{8.1}$$

Constant k_{mag} depends on the magnetic properties of coupling (magnetic permeability), geometrical properties of the bearings (area) and coils (length, number). If that force is linearized about an equilibrium condition of the floating rotor, one can write:

$$F = F_0 + k_i i + k_x x \tag{8.2}$$

where a constant component of force F_0 appears and two other ones depend on the current (k_i) and on the rotor displacement (k_x), respectively. If the layout described in Fig. 8.5 is applied to each axis, the equivalent mechanical stiffness of the suspension k is:

$$k = k_i K_a K_s K_p - k_x \tag{8.3}$$

where the component of force depending on the displacement is unstabilizing, being negative as sign indicates in above Eq. (8.3), and the damping coefficient equivalent to the non-rotating damping, c, is:

$$c = k_i K_a K_s K_d \tag{8.4}$$

where K_a is the gain of power amplifier, K_s that of sensors, while K_p and K_d are the gains of the control system, being associated to the proportional and derivative actions, respectively.

The above described model is useful to define all of components required to assembly the system and perform the due integration. Those elements appear inside the BDD and IBD of the system and compose the PBS. Moreover, for a complete verification of requirements, it is needed that some performances are predicted through the numerical model and a simulation, by solving the equations of motions of the system, being described through several degrees of freedom [at least the two displacements and the two rotations which allow defining the so-called whirling motions of the rotor in the two orthogonal planes containing the main rotational axis (Genta, 2005)].

The numerical model of the system shall define some typical reference targets of the design activity as:

- the critical speeds of the whole rotor, ω_{cr1}, ω_{cr2} …
- the instability threshold of the whole rotor $\omega_{inst.th.}$.
- the limitations in strength of materials, like the spin speed inducing a plastic behavior within the material of connected discs.

Particularly, the *critical speeds* correspond to a resonance of the system, i.e. the amplitude of whirling motions grows up fairly fast, but above those values (the so-called super-critical regime) the rotor undergoes the self-centering condition. In this case, its center of mass is aligned to the geometrical center found by connecting the centers of supports, and the amplitude of the reactions of bearings to the forces imposed by unbalance is lower. Above the so-called *instability threshold*, the amplitude of the whirling motions grows up exponentially and the risk of failure is extremely high. Above the threshold, every value of spin speed excites an unstable behavior. Moreover, above a certain spin speed, the actions induce a large stress inside the material, especially in discs, wheels, etc., which may lead to a plastic behavior or even to rupture. Therefore, the angular velocity of the first yield condition is usually predicted. Obviously, this effect is associated to other strength conditions concerning the fatigue, creep, impact, rubbing and so on.

Those targets are strictly linked to the design parameters of the rotor and of its suspension system. In case of a plane model of the rotor, i.e. the so-called Jeffcott's rotor, if one assumes that the system is isotropic (equal properties along the two perpendicular radial direction) the critical speed is:

$$\omega_{cr} = \sqrt{\frac{k}{m}} \tag{8.5}$$

while the instability threshold is:

$$\omega_{inst,th} = \sqrt{\frac{k}{m}}\left(1 + \frac{c_n}{c_r}\right) \tag{8.6}$$

The last equation shows that the instability threshold grows up with a higher non—rotating damping as the damping previously described in the active magnetic

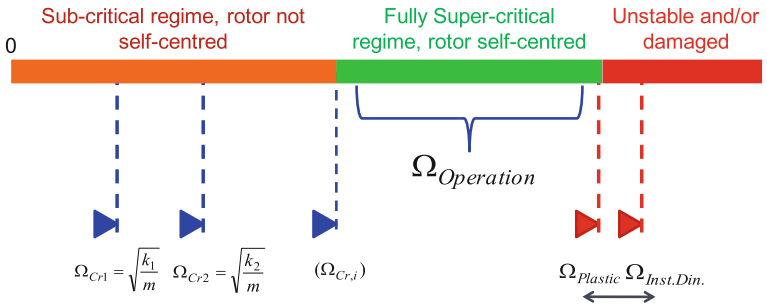


Fig. 8.6 Sketch of the operation of the actively control rotor

suspension, while is reduced as any dissipation occurring inside the rotating part of the system is increased (c_r).

When more degrees of freedom are considered, those expressions become more complicated, but a numerical modeling of the system is available to compute their values. In particular, the gyroscopic effect is added and interacts with the rotor dynamics.

In practice, the physical model of the rotor system is aimed at defining the range of spin speeds allowed for the operation. It should assure always to operate in self-centering condition, i.e. in supercritical regime, but never above the limit of a structural damage of materials or of the instability threshold (Fig. 8.6). A suitable design should carefully size the structures of the rotor system, by a numerical model allowing a prediction of the stress ad strain occurring into the materials and a suitable setting of control parameters to effectively control the system dynamics.

Physical models. The didactic test case belongs the so-called field of structural rotor dynamics. To give a rough overview upon the real contents of a physical simulation applied to that kind of system, an essential description of the main analyses usually performed in that domain is herein proposed.

As a matter of facts, designer is interested in this case to:

- define the *geometrical model* of the whole system to check its compatibility for assembly and integration with the main plant;
- predict the *dynamic performance* in rotation, to prevent the effects of critical speed, dynamic instability and unbalance;
- set-up the *control* based on the active magnetic suspension, to perform the design synthesis of the whole system;
- prevent failures and related risks, through a consistent *RAMS analysis*;
- predict the *materials behavior* to prevent any unsafe condition in operation.

Those tasks are reached by resorting to some preliminary models:

- *geometric*, describing shape, volume and size of the system;
- *dynamic*, represented in terms of dynamic equilibrium equations of the whole system;

- *structural*, describing the system as an assembly of elastic bodies subjected to a stress and strain distribution within materials under loading condition;
- *of system control*, to define its strategy applied to the dynamic behavior of system. It includes a detailed description of actuation and sensing operations and of related devices applied to provide that service;
- *of system reliability*, through several kinds of approaches now available in the literature.

All those models should be harmonized, interoperated and effectively connected to the requirements and the functional models previously described.

In the didactic test case, the activity above mentioned was developed as is herein briefly described.

Geometric model. A preliminary geometric model of the rotor is needed to identify shape, size, weight and physical properties of rotor and connected accessories. Geometry is created through a suitable software, by means of a mathematical model, through a rough graphical impression of an industrial laying head. It simply describes each architectural detail, as in the technical drawings provided to the manufacturing units along the product lifecycle development. This activity allows knowing the proposed volume and related mass and inertia to be made rotating and controlled.

Dynamic model. Once that the digital model of rotor is ready, the whole system consisting of stator, suspension and rotor is analyzed. A structural model is developed, through some mathematical issues. The static interaction between rotor and stator can be easily investigated by assimilating the rotor to a rigid body and computing the reactions required to the suspension system, when the rotor is simply stand-alone and supported by some bearings. This approach is aimed at defining a preliminary configuration of the so-called "bias" parameters of the suspension, when active, like in this example. Reactions are computed as in beamlike structures, when static equilibrium equations are written. Moreover, it is possible to define even the displacement of the rotor from the statoric platform, to check that no interference between bodies occurs.

To analyze the dynamic behavior in rotation, the theory of rotor dynamics can be applied. The suspension is roughly modeled as an equivalent spring-damper system (Fig. 8.7). The rotor could be either assumed to behave as a rigid or flexible body. In this case it was sufficient investigating the rigid body motion, since the effects of flexibility of shaft could affect the dynamic behavior at fairly high spin speed.

A simplified and equivalent model of the rotor is sketched in Fig. 8.8. The main body is just represented by the line axis. Rotation occurs about the Z axis, at spin speed ω, while coordinates X and Y describe the so-called whirls of rotor in the radial plane. The two bearings of the suspension system are defined as A and B, respectively. The electric motor applied between bearings to make the system rotating, shown in Fig. 8.7, is either introduced as an action depending on time in transient analysis, or just considered as an operation constraint allowing a regular and constant spin speed, in the frequency domain.

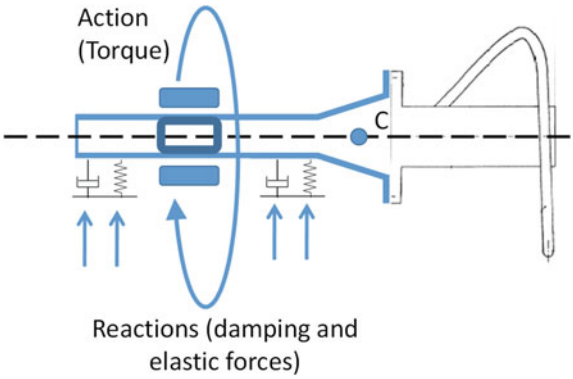


Fig. 8.7 Sketch of the modeled test case

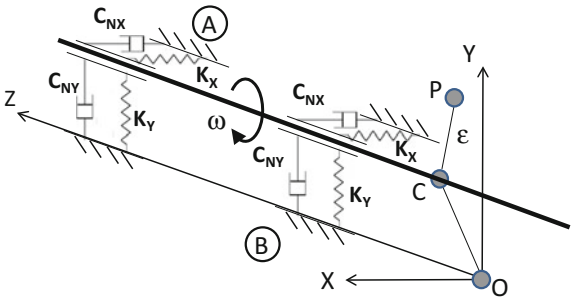


Fig. 8.8 Discrete model of the test case system

Usually, the dynamic equilibrium of rotor is written in a fixed reference frame whose origin is O, by identifying the position of center of mass, P, eventually unbalanced of an eccentricity ε, from the geometric center of the rotor cross section, C. The linearized equations of motion, when only four coordinates are used, namely two translations x and y along the two radial directions and two rotations φ_Y and φ_X in planes ZX and ZY, respectively, and the system is undamped are:

$$m\ddot{X} + K_x X = m\epsilon\omega^2 cos(\omega t + \alpha)$$

$$m\ddot{Y} + K_y Y = m\epsilon\omega^2 sin(\omega t + \alpha)$$

$$J_t\ddot{\varphi}_X + J_p\omega\dot{\varphi}_Y + K_{\varphi X}\varphi_X = -\chi\omega^2\left(J_t - J_p\right)sin(\omega t)$$

$$J_t\ddot{\varphi}_Y - J_p\omega\dot{\varphi}_X + K_{\varphi Y}\varphi_Y = \chi\omega^2\left(J_t - J_p\right)cos(\omega t) \tag{8.7}$$

where m is the rotor mass, K_x, K_y, $K_{\varphi X}$, $K_{\varphi Y}$, are the stiffness coefficients applied to the four dof's, ω is the spin speed, ε is the static unbalance and α describes its position in the radial plane through a polar reference. The third and fourth equations describe the role of gyroscopic effect, through the transversal, J_t, and polar, J_p,

moments of inertia of rotor, the dynamic unbalance χ, and its relative location to the shaft, described by β.

The above equations can be transformed in matrix form as:

$$M\ddot{q} + (C_n + C_r + \omega G)\dot{q} + (K - i\omega C_r)q = \omega^2 \begin{Bmatrix} m\epsilon e^{i\alpha} \\ \chi(J_t - J_p)e^{i\beta} \end{Bmatrix} e^{i\omega t} \tag{8.8}$$

The system inertia is introduced by matrix M. The non-rotating damping, favorable to stabilize the rotor, is C_n and is provided by bearings. A rotating damping, C_r, is even included. It is induced by any kind of dissipation occurring within the rotor, like the friction between wire rod and rotor structures in present case. It concurs to make unstable the system, above the instability threshold and in supercritical regime, since it affects the stiffness term, by introducing a contribution which is negative and rotated of 90°, as the imaginary unit indicates. The gyroscopic effect associated to the polar moment of inertia is introduced by matrix G. Equation (8.8) describes a pure mechanical and uncontrolled system. It is used to find all the design parameters above mentioned. A periodic and harmonic solution is assumed as:

$$q = q_0 e^{st} \tag{8.9}$$

being q the set of complex coordinates, q_0 the amplitude of dynamic response, and $s = \sigma + i\lambda$ the complex frequency of whirling motion, whose real part, σ, describes the rate at which the amplitude increases in time. A negative value is beneficial since the whirl is damped and gradually decreases, while a positive value is a symptom of dynamic instability. The imaginary part, λ, is the frequency of whirling motion. The homogeneous equation associated to Eq. (8.8) allows calculating the critical speeds. The above equation is written with a null right hand term and the solution of Eq. (8.9) is introduced, by setting $\lambda = \omega$. The unbalance response is found in terms of amplitude of whirling motions, by solving the complete forced equation of motion Eq. (8.8).

The instability threshold of forward whirls is found by plotting the amplitude of all the whirling motions associated to the dof's, as a function of spin speed, ω, and finding when they suddenly grow up, in correspondence of a change of sign in the real part of s, as in Fig. 8.9, where values were made nondimensional with respect of the scale of graphs (s^*, s°, ω^*). In the so-called Campbell diagram, when the dynamic response is that of Eq. (8.9), the real part of parameter λ describes the frequency of whirling motion, while its imaginary part describes the decay rate of whirl. Transient dynamics can be investigated by solving in the time domain the same equation of motion and plotting the trajectory of point P in the radial plane as a function of time (Fig. 8.10).

Electromechanical coupling. Right now, the rotor was simply considered as a mechanical system, without control action. When an active magnetic suspension is applied and radial active magnetic bearings are used to support and control the rotor, Eq. (8.8) can be updated as follows:

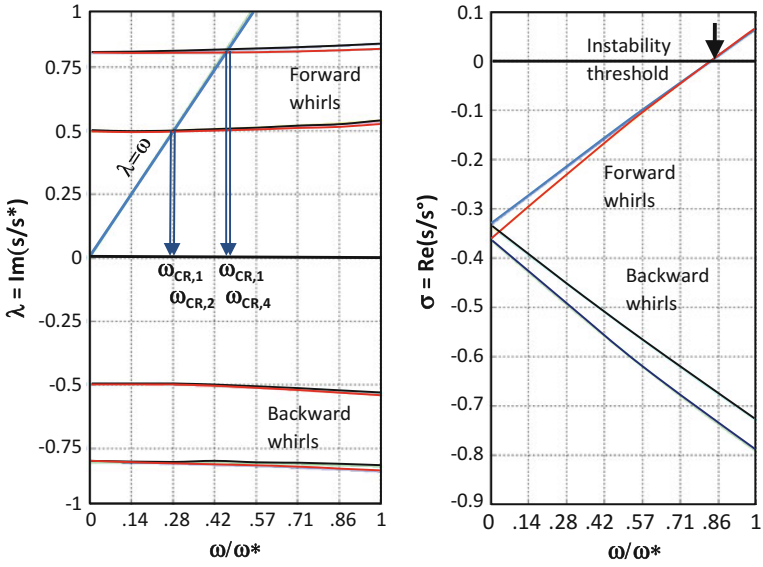


Fig. 8.9 Example of numerical results of in rotor dynamics modeling

$$M\ddot{q} + (C_n + C_r + \omega G + C^*)\dot{q} + \left(K + K^* - i\omega C_r - K_u^*\right)q = F_n + F_0^*\omega^2 \begin{Bmatrix} m\epsilon e^{i\alpha} \\ \chi\left(J_t - J_p\right)e^{i\beta} \end{Bmatrix} e^{i\omega t} \tag{8.10}$$

As it was previously described, the control action applied by the magnetic field created by the magnetic bearing consists of a constant effect, due to bias current, referred to as F_0^* which contrasts the weight indicated through vector F_n. In case of effective calibration of the system, those two contributions should compensate each other and disappear from Eq. (8.10). The variable contributions, in a simple control like that assumed, are associated with damping control matrix C^*, stiffness control matrix K^*, and K_u^* whose definition could be understood by referring to previous Eq. (8.2), Eq. (8.3) and Eq. (8.4). Moreover, since the application of a contactless suspension is aimed at avoiding a mechanical connection between stator and rotor, the unique effect of non-rotating damping is provided by C^*, as well as stiffness only consists of K^*, and K_u^*. Therefore, the final model to be implemented corresponds to:

$$M\ddot{q} + (C_r + \omega G + C^*)\dot{q} + \left(K^* - i\omega C_r - K_u^*\right)q = \omega^2 \begin{Bmatrix} m\epsilon e^{i\alpha} \\ \chi\left(J_t - J_p\right)e^{i\beta} \end{Bmatrix} e^{i\omega t} \tag{8.11}$$

Structural model. The above equations allow studying the dynamic response of the controlled system in open loop, closed loop, in terms of critical speeds, dynamic stability, unbalance response and performance of the active suspension. Actually,

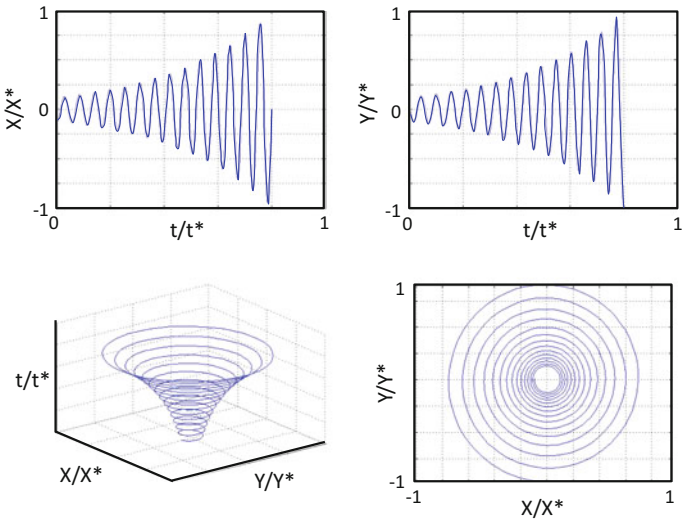


Fig. 8.10 Example of the numerical modeling of the rotor dynamics (orbits in presence of dynamic instability)

preventing the risk of plasticity inside the materials of rotating parts and damage, like fatigue, static rupture or buckling if a compression is applied to a slender shaft, is a key issue of safety engineering and of design. Those phenomena could be investigated by resorting to a preliminary stress analysis applied to the whole system, in several operating conditions. The finite element method is appropriate for this activity. It allows computing the displacements and rotations of elastic body, after a preliminary discretization into finite elements, in correspondence of the so-called nodes first, and then, in other parts of the structure, together with stress and strain. Usually, the geometric model is discretized in finite ad simple elements and then loaded in correspondence of nodes. The code automatically writes some equilibrium equations, associated to elements, to be numerically solved to find, for instance, the stress distribution as in Fig. 8.11, where a small wing is modeled, discretized, and stress analysis is finally shown. Analytical methods could be applied to the shaft of the didactic test case, but a numerical approach is better for the laying head and some other components to prevent the occurrence of yielding and rupture or fatigue collapse.

Tools. In the didactic case the tool chain basically includes a *geometric modeler* to draw the system, whose result looks like an input for a *dynamic simulator*, where both the rotor dynamics and the energy conversion provided by the active magnetic suspension under the command of control are modeled and a *finite element code*. In principle, a professional tool for CAD could be linked to a dynamic simulator like Modelica®, Simulink® or similar and then to a FEM code like the MSC Nastran®, Abaqus®, Ansys® or similar. In this case, it was possible resorting to an unique simulation environment like the DYNROT© code, being a third party of the

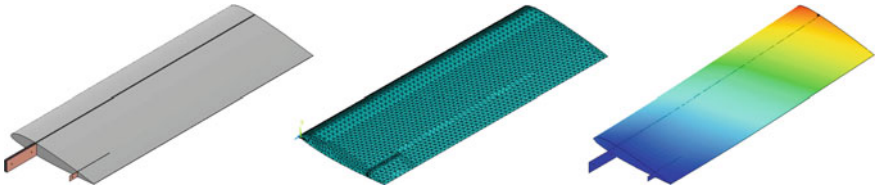


Fig. 8.11 Example of finite element modeling of a wing (geometry, discretization and equivalent stress distribution under uniform load)

MATLAB®, in which the main routines to analyze the rotor dynamics, the active control produced by magnetic bearings and the stress occurring inside the main rotating shaft, as is identified in Fig. 8.2, were implemented. This selection avoided the difficult synchronization of different software.

The design process could be deployed as follows. A proposed layout is preliminarily drawn. It leads to know the mass, inertia, dimensions, weight and volume of the rotor. All those numerical data are listed into an input file. It is either obtained by the geometric modeler when used, or inside the rotordynamics code directly, through the mesh of elements. Weight suggests the static reactions of suspension system and leads to know the bias current required to apply the desired forces. Immediately, this current is a reference for the actuator, both in terms of magnetic circuit and of temperature induced by the Joule effect. A maximum eccentricity corresponding to the worst unbalance condition is defined, depending on the class of unbalance which the rotor belongs. This input suggests the range of control currents to be added to the bias force and to complete the actuator design. Moreover, the range of radial displacements to be monitored is even defined, since the goal of control system is interacting with the rotor as soon as it leaves its centering condition. Basically, those data allow defining some gains described in Eq. (8.1) to Eq. (8.4). Particularly, k_i, k_u, F_0 are preliminarily selected. A technological screening on the available power amplifiers and sensors suggests some suitable values of K_a and K_s, while targets about the critical speeds and the instability threshold can be used to find a preliminary set of gains for the control system, particularly K_p and K_d. Those inputs can complete a proposed layout for the actively controlled system and the rotor dynamic analysis could be performed to verify requirements about the unbalance response, the dynamic instability, the maximum spin speed allowed and the distance between stator and rotor, even by resorting to the stress analysis, as it was previously detailed.

8.6.2 Industrial Test Case

The Ice Protection System case study is a typical example of equipment which requires a detailed physical analysis to establish a reliable sizing process and an assessment through dynamic simulation of operating conditions. This is a

safety-critical system, since it must face a dangerous problem, like the ice accretion on airfoils and nacelles, during the aircraft flight, because it may lead to the aircraft uncontrollability, if it is not detected and stopped. The ice accumulation on wings, empennages and engines has a negative effect, like weight and drag increasing, and reduction of lift. In addition, unbalance due to asymmetric aerodynamic loads could be introduced. A suitable solution shall be then identified either to avoid the ice accumulation (anti-icing) or to melt and/or break the ice, once it was grown on the aerodynamic profiles (de-icing).

All aircrafts are equipped with simple or complex IPS, which resort to some technologies, depending on the application, category of aircraft and performance requested. This example provides some main features of this system, with particular focus on the physical analysis and simulation required to assess its characteristics and behavior, starting from the results obtained during the definition of system architecture, within the Logical Analysis.

Problems and solutions. Everybody knows that ice can be melted by providing a certain amount of energy heating a solid volume or by applying a load to break it. Moreover, some chemical reactions can be helpful to prevent the ice accretion, for example, exploiting some dedicated fluids, exhibiting particular characteristics at low temperature. Those considerations motivate the number of IPS proposed and applied not only in aviation domain, but even in other industrial fields, like automotive engineering.

The physical phenomenon is quite simple, but selecting the best IPS solution depends, case by case, on some different issues, to be carefully evaluated. Performance, reliability, installation and cost are examples of criteria to be considered in design. The nature of ice can be even different, depending on the operating conditions (ice accretion rate, dynamic conditions etc.…) and on the environment (altitude, pressure, temperature, etc.…). This affects a lot the choice.

Concerning aeronautics, ice accretion is caused by the presence in clouds of supercooled droplets of water, being liquid even in case of temperature lower than 0 °C. They freeze instantly, when impacting the aircraft surfaces. The presence of water in clouds is generally measured through a parameter called Liquid Water Content (LWC), in grams per cubic meter, whilst the diameter of droplets is assessed through the Mean Volume Diameter (MVD), in micrometers. The phenomenon is also influenced by temperature and by the type of cloud encountered during flight. The most common environment includes droplets of around 15–40 μm, at temperature ranges from 5 to −20 °C, with maximum associated LWC of 0.8–2.8 g/m^3, when dealing with stratiform and cumuliform clouds, respectively. Those atmospheric conditions, concerning clouds, are referred to as Continuous Maximum and Intermittent Maximum atmospheric icing conditions, while the whole environmental parameters herein cited are listed within the Appendix C of the CS-25 regulation concerning the atmospheric icing for Certification Specification of Large Aeroplanes (EASA 2017) (Fig. 8.12).

During last years, some severe icing conditions were met by several aircrafts around the globe, leading to very high accretion rate and extension of surfaces undergoing the ice accumulation. The certification of aircraft, under these extreme

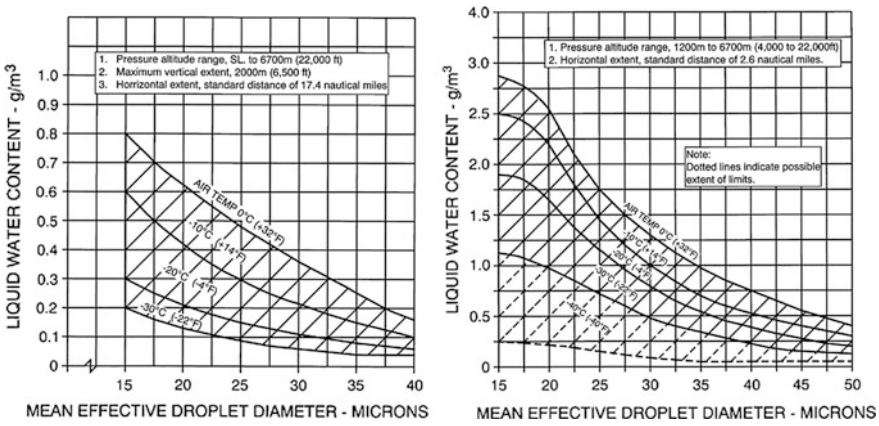


Fig. 8.12 Continuous Maximum (left) and Intermittent Maximum (right) LWC profiles defined within Appendix C of CS-25

weather conditions was then included within a new chapter of the regulation, named Appendix O, concerning the Supercooled Large Drops (SLD) icing conditions. The SLD environment includes freezing drizzle and rain, whose drops may reach more than 500 μm (even 1000 μm sometimes), with LWC and temperature envelopes described by Fig. 8.13.

The determination of accretion rate, resulting from those environmental conditions, the extension of protection on the aircraft most impacted zones and the effectiveness of the IPS system are typical issues to be faced by relying on physical simulation. Data coming from the regulation are used as a main input to the system model, since they contribute to describe the operational environment.

A typical IPS is here considered. The Ice Protection Systems use the power source available on the aircraft to provide an effective anti/de-icing mean to face the ice accretion phenomenon. They resort either to electrical or pneumatic power provided by on-board systems (Electrical Power System and Pneumatic System respectively). Some combinations of these power sources may be used to derive hybrid systems, which are currently under study, together with some other solutions (Goraj, 2004), like the use of alcohols and fluids, as it is exploited on some small aircrafts. The electrical and pneumatic solutions represent the most widely used in aeronautics. Some properties are herein described.

- Pneumatic IPS—this system uses pneumatic air coming from the engines bleed, which is fed at high temperature and pressure. The air flows through dedicated piping, under the leading edges of aerodynamic profile, to melt the ice layers, in direct contact with the skin. Aerodynamic drag then detaches the ice pieces from the external surfaces.
- Pneumatic IPS with inflatable boots—this system uses pneumatic air coming from the engines, to inflate some rubber chambers, located on the skin of airfoils

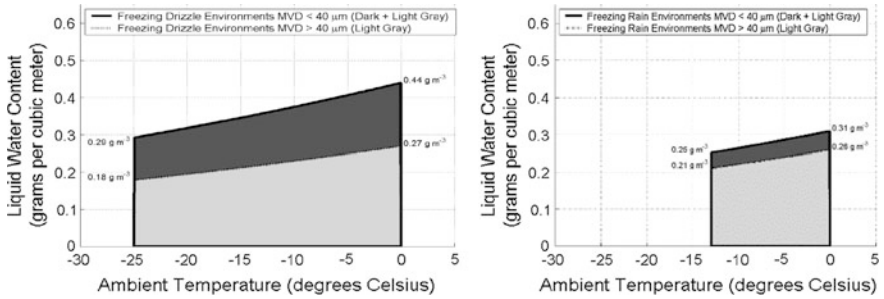


Fig. 8.13 Freezing drizzle (left) and freezing rain (right) LWC profiles defined within Appendix O of CS-25

to break the ice by mechanical action, basically due to the changed shape of profiles.

- Electro-thermal IPS—this system uses the Joule effect provided by dedicated resistances located on the skin of the airfoil. The classic system uses some alternated continuous heated resistances and some cyclic heated resistances to heat different zones, thus creating some different temperature gradients, to melt ice layers in contact with surfaces. Aerodynamic drag detaches then the ice from the airfoils.
- Electro-mechanical IPS—this system uses a combination of electrical heating on the skin and electro mechanical actuators, located below the skin, to produce a dynamic action, able to detach ice layers from the airfoil.
- Electro-expulsive IPS—this system uses piezoelectric actuators, located under the skin, to produce concentrated loads on the structure, to detach ice layers from the airfoil. Electrical heating is not used.

Some other technologies are even available, although they are right now far less applied.

- Shape Memory Alloy based IPS—this system uses some specific shape memory alloys. They change their chemical phase and shape, when heated through an electrical stimulation, and provide some actions, induced by strain. This system looks similar to pneumatic boots, in terms of shape change.
- Ultrasound IPS—this system uses sound waves to destabilize the ice layers in direct contact with the skin, through some dedicated actuators, installed inside the airfoil.
- Antifreeze fluid based IPS—this system uses a porous channel system to pump glycol fluid on the aircraft surface, lowering the freezing point of the water and acting as anti-ice mean.

Not all the described IPS solutions are actually operational. For example, electro mechanical and electro expulsive systems were tested only in laboratory or on some prototype, in maiden flights. Electro-thermal and pneumatic IPS are equipping the

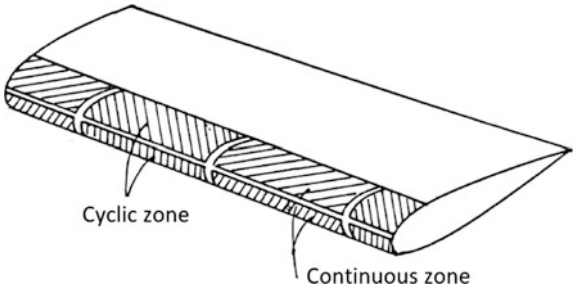


Fig. 8.14 Typical arrangement of electrical heaters (Chiesa, 1995)

most of commercial aircrafts in service. Fluid based IPS is also applied, in general aviation aircraft.

Selected IPS for the case study: electro-thermal IPS. This system is a combination of anti-ice and de-icing strategies to face the problem of ice accumulation. It uses the heat produced by resistors, through the Joule effect, to raise the temperature of surfaces where they are bonded. It is a medium-high power consumption system, requiring a lot of energy per unit time. Electrical power is exploited in two ways, to provide a total ice protection. The total protected surface is divided in two main areas, which have different functions. A continuously heated zone is close to a cyclic heated zone (Fig. 8.14).

The two zones are alternated to create a sort of grid. The continuously heated surfaces are located on the very front part of the leading edge and in between the cyclic zones to produce an anti-ice mean. Cyclic zones work as a pure de-icing device and provide high power, in a very short time, to melt the ice layers, in direct contact with the skin. The detachment of ice due to aerodynamic effect is facilitated by the presence of adjacent continuous zones, which prevent that ice blocks remain stuck on the limit of the cyclic zones. The portion of protected surface is limited to the 15% of the mean aerodynamic chord of the airfoils.

The number of continuous and cyclic heated zones and the maximum power peak constitute the main sizing parameters of this system (Chiesa, 1995). To reduce the power consumption, the optimum number of cyclic and continuous heated zones shall be determined. Timing and sequence of activation of those zones shall be even derived, to avoid too high peaks of power consumption.

The selection of number of zones protected and the determination of maximum power can be performed looking at the simple sizing laws that characterize the behavior of continuous and cyclic heated zones. Power is a function of the number of zones and has a minimum in correspondence of a certain number of cyclic zones. The power P_{cont} required to heat continuously some zones is directly proportional to the surface S_{cont} protected with the anti-icing strategy, through the specific power q_{cont}, which is around 21 kW/m^2, in case of continuous heaters (i.e. the higher the number of continuous zones, the higher the power):

$$P_{cont} = S_{cont} \cdot q_{cont} \tag{8.12}$$

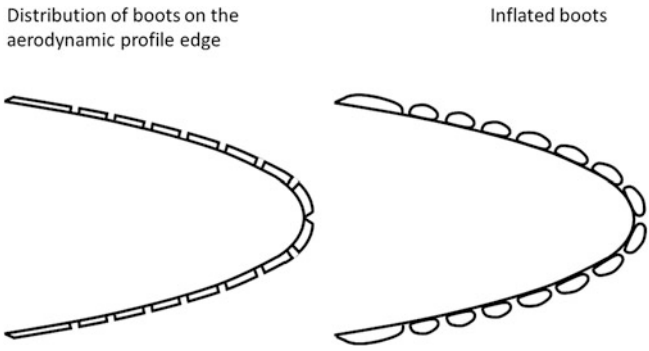


Fig. 8.15 Typical arrangement of pneumatic boots (FAA, 2012)

The power required to heat one cyclic zone P_{cycl} (only one at time) reduces with increasing number of zones n_{cycl}, since the higher the number is, the lower the extension of the single zone over the cyclically protected surface S_{cycl}. The specific power q_{cycl} in this case is around 28 kW/m^2 or higher. Those figures actually depends on the heating time, the smallest the time of actuation is, the higher the specific power shall be to produce the same effect, i.e. the energy per second provided to the zone shall be higher since the activation time is lower:

$$P_{cycl} = S_{cycl} \cdot q_{cycl}/n_{cycl} \tag{8.13}$$

The total power P_{TOT} is then the sum of the two contributions:

$$P_{TOT} = P_{cont} + P_{cycl} \tag{8.14}$$

The number of zones shall be determined for each aircraft section, to find the optimum solution for wing, horizontal stabilizer, vertical fin and engines inlets. Moreover, the optimum condition found may be changed, considering timing issues, which constitute the second sizing problem. The number of cyclic heated zones found may be too high, thus requiring a huge activation sequence, in terms of time. To reduce the power peaks, only one zone is usually actuated at time, avoiding superpositions, except for very low-power zones. If a long time passes between two subsequent activations of the same zone, the ice accretion may be too high and heater may not be capable of melting the inter-cycle ice (ice accumulated

between two activations) (FAA, Effect of Residual and Intercycle Ice Accretion on Airfoil Performance, 2002).

A compromise shall be found, between the reduction of power due to the extension of heated zones and the timing of the actuation sequence. A practical solution, if the optimum number of cyclic zones shall be kept, consists in raising the specific power of cyclic heaters, reducing the activation time for each zone. However, some technological limits are present, for the commercial heaters used for this kind of system. Therefore, it is not possible to raise too much the specific power without causing some damages or affecting the reliability of this component.

As a final remark, it shall be highlighted that electro-thermal anti-icing is also used to prevent the ice accretion on small appendices like antennas, sensors, horns and propeller blades, when present, while de-icing devices are inapplicable. An additional amount of power shall be then considered within the final power budget, to avoid problems during sizing.

Selected IPS for the case study: pneumatic IPS with inflatable boots. This system uses some inflatable chambers, located close to the leading edge of the airfoil, which break the ice during an inflation/deflation process, by resorting to the force produced by a dynamic change of size of boots and of the geometry of airfoil (Fig. 8.15).

The pneumatic power is taken from engine bleed as a high-pressure airflow. The amount of airflow required to inflate the boots is a main sizing parameter, although its determination is never easy. During the aircraft operation, the power provided shall be enough to inflate the boots, which exhibit an intrinsic resistance due to flexibility of rubber material, against the effect of air pressure flowing at very high speed and against the adhesion resistance of the ice layers accumulated. The balance of forces can be investigated:

$$p_{regulated} \cdot S_{boot} = F_{boot} + 1/2 \cdot \rho \cdot V^2 \cdot S_{boot} + F_{ice} \tag{8.15}$$

where the action applied by the boot, on the left hand, is in equilibrium with the reaction of polymeric material, F_{boot}, the aerodynamic effect, associated to the aircraft speed, V, and the action of ice accumulated, F_{ice}.

One of the most difficult contributions to be evaluated is the ice adhesion. Generally, it is identified through some tests on ice, performed within the wind tunnel, since it is rather difficult modelling these phenomena, to suitably predict their action, as a function of speed, type and amount of ice, atmospheric conditions, rugosity of surfaces, etc.…

Considering some operating parameters, pressure and temperature shall be controlled to avoid any damage of boots. A pressure regulator is required as well as a distribution subsystem, between engines and actuators. The amount of airflow taken from the engine, usually from compressor stages, shall be reduced as much as possible to avoid destabilizing the turbomachinery. Volume and dimensions of protected areas depend on the size of aircraft and aerodynamic appendices. In any case, the chordwise extension is generally lower if it is compared to the electro-thermal IPS, only the 10% of the mean aerodynamic chord.

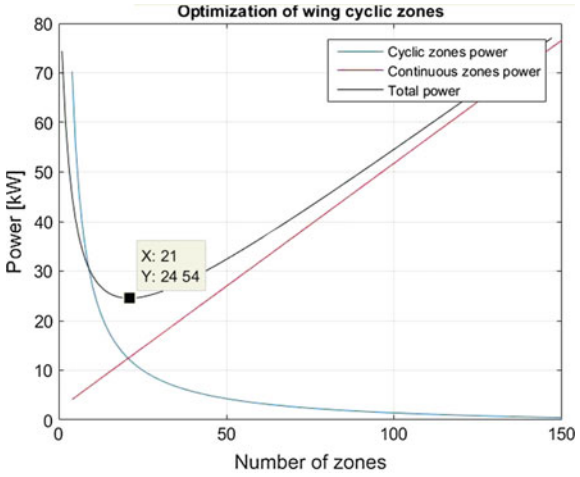


Fig. 8.16 Determination of minimum power consumption of the wings

This system is widely used for commercial turboprop aircrafts, general aviation and small business jets. However, it has some drawbacks in terms of aerodynamic interface, since boots actuation slightly increases the resistance of profiles and may affect the overall balance, in case of asymmetric activation, as fuel consumption. The air taken from the engine compressor, increasing the fluid pressure, while consuming a portion of the energy coming from fuel, is subtracted to thrust. Considering the aircraft subsystems, the need for a pneumatic system increases the overall complexity of the architecture and requires a dedicated power feeding. The modern aircraft design is looking for more-electric (MEA) or even all-electric (AEA) subsystem architectures, aimed at reducing or eliminating the presence of non-electrical power source, like pneumatic and hydraulic ones, to reduce the overall complexity and to assembly a more homogeneous set of subsystems, with standardized interfaces.

Considering the reliability, boots suffer fatigue, more than in case of electrical heaters. Presence of an electrical anti-icing to support the pneumatic system is always required for small appendices; therefore some additional devices, using the electrical power sources, shall be designed separately.

A great advantage of this system is the low power consumption, if compared to the electrical IPS and the general simplicity of architecture, which has been used for more than 50 years in aeronautics, starting from World War II.

Physical models. For the proposed case study, three physical models are provided as an example of implementation. A *sizing model* for the electro-thermal IPS and two *dynamic models* for both the candidate systems are examined.

Sizing model for the electro-thermal IPS. This model is aimed to determine the number of zones protected, the maximum power peaks and timing, for an electro-thermal IPS. It consists of an ad hoc built executable code in the Matlab® environment, receiving the geometric data of protected aircraft surfaces from the

SRS, the profile of airfoils and some inputs about the specific power, defined by the user, to compute the above mentioned results.

Considering the aircraft described in Chap. 4, some main results are collected in charts, which visualize the optimal number of cyclic heated zones, and the related power peaks as a table view. Figure 8.16 shows the chart related to the protection of wings.

The best result includes 21 cyclic heated zones, with a power consumption of about 24.5 kW (peak), considering wings only. Figures 8.17, 8.18 and 8.19 show some similar results for horizontal stabilizer, vertical fin and engines.

When the optimal number of protected zones is set up, assuming the maximum power peak, computed including the total continuous power needed for the continuous activity of heaters and appendices, the maximum power requested by the activation of the most impacting cyclic zone (wing) is about 51.6 kW. If 74 cyclic heated zones are assumed, even considering the components exploiting the maximum specific power available on the market (62 kW/m^2 for an activation time of only 2 s) the total sequence lasts 148 s. This appears too much, considering the inter-cycle icing issues.

The number of zones can be reduced, accepting a slight increasing of power consumption. This analysis is summarized in Table 8.1, where power peaks are listed, for a modified configuration and power deltas describe the additional power required compared to the optimum conditions.

The power increases, because of the non-optimal number of cyclic heated zones, but it is limited to 7% of the power peak considered for the worst case. However, since the total number of zones is almost halved (38 instead of 74) the required time, to perform the same actuation, is about 76 s, instead of 148.

If power consumption is a minor problem, the areas protected can be furthermore reduced. This compromise needs to be evaluated through a dynamic model, aimed at assessing the actual ice accretion, under the conditions specified by the CS-25. Results obtained from the preliminary sizing model in terms of specific power, sequence of activation and protected surfaces, listed in Table 8.2 for the modified case, should be evaluated together with the numerical results of simulation.

It is important to highlight that extension of protected areas depends directly on the actual value of airfoil surface, through a fixed percentage of spanwise and chordwise coverage. Thus, considering a specific configuration, the total protected surface is always the same, but, depending on the number of cyclic heated zones selected, the amount of continuous and cyclically heated surfaces changes accordingly. Between two cyclic zones, a continuous heated zone is required to assure ice detachment, making the number of continuous zones a direct function of number of cyclic ones.

Results obtained through the sizing model can be translated into some non-functional requirements. For example, it is possible to generate requirements for the number of zones, the protected surface, the timing and, especially, the power levels. Off course, the update and verification of requirements can be possible only by exploiting a dynamic simulation of the system, implementing the operating conditions specified by regulations.

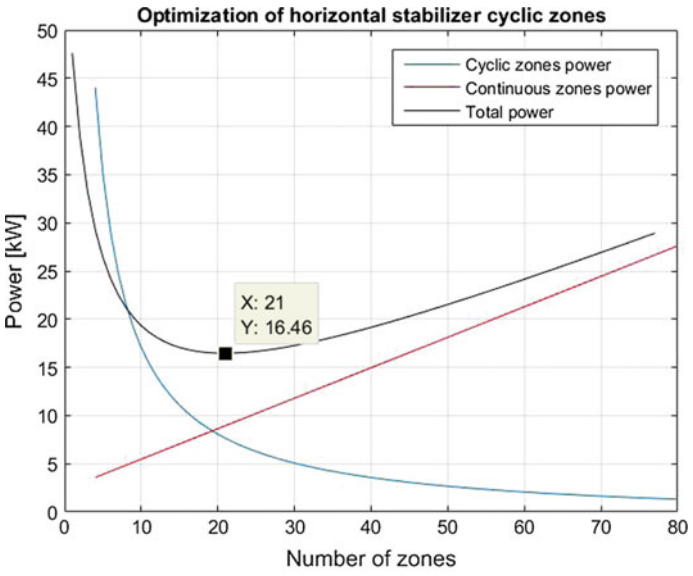


Fig. 8.17 Determination of minimum power consumption for the horizontal stabilizers

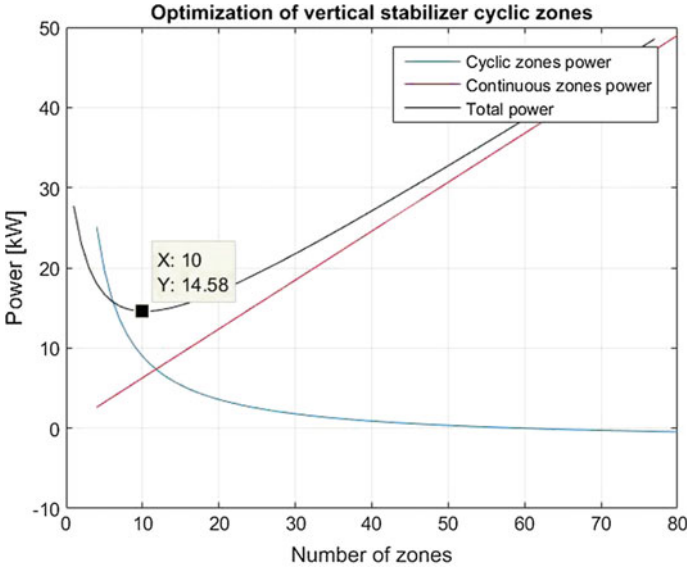


Fig. 8.18 Determination of minimum power consumption for the vertical stabilizers

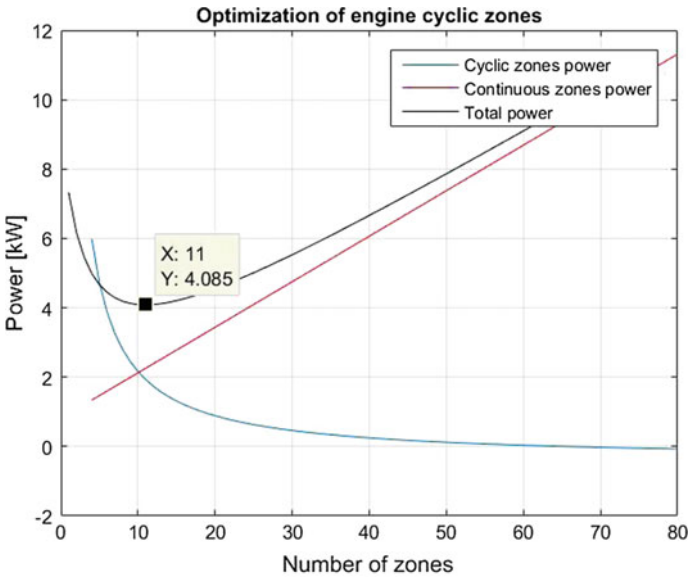


Fig. 8.19 Determination of minimum power consumption for one engine

Table 8.1 Summary of determination of number of zones with associated power consumption

Zone	Optimum	Selected	Power peak (kW)	Power delta (kW)
Wing	21	14	45.2	+1.5
Horiz.	21	10	41.3	+2.9
Vert.	10	6	41.2	+1.2
One Eng.	11	4	36.1	+0.9

The connection of two models is guaranteed by the re-use of the parameters defined during the sizing process. For the proposed case study, the Simulink® is used as a modelling environment, to represent the system dynamics. In this case, the Matlab® and Simulink® allow directly using the same workspace making the dynamic model capable of accessing the data previously computed. The consistency of data is thus automatically guaranteed.

Dynamic model for electro-thermal IPS. This model has been derived to assess the ice removing efficiency of electro-thermal IPS, under the conditions specified by regulation; to verify the hypothesis about power consumption and to evaluate the necessary control strategies for the actuation of protected zones. It derives from the system architecture, formulated within the Logical Analysis and, being further developed, it provides an executable version where numerical results can be analyzed. Requirements allocation on physical blocks is also supported to organize the verification campaign. The model is developed in the Simulink®

Table 8.2 Areas of protected surfaces for different aircraft zones

Zone	Cont. surface (m^2)	Cycl. surface (m^2)	Subtotal (m^2)
Wing	0.67	4.23	4.9
Horiz.	0.47	2.70	3.2
Vert.	0.39	1.47	1.9
One Eng.	0.13	0.37	0.5
TOTAL	1.66	9.14	10.8

environment, which is fully compatible both with the requirements manager tool (DOORS®) and with the functional modeling tool (Rhapsody®), thus allowing a good interoperability within the selected toolchain.

Three main stages of model development are herein described.

- Preliminary implementation phase, consisting in the update of logical architecture, to be compatible with physical analysis tool, in terms of data types, and in a first implementation within the physical modeling environment.
- Detailed implementation phase, referred to as the main physical modeling activity, where the architecture is updated and the final model is simulated to compute the numerical results.
- Requirements allocation and verification phase, where requirements are connected to the physical blocks to organize the verification activity.

As far as the preliminary implementation is concerned, the main source of data is the communication network, represented through some IBDs in the Logical Analysis, where the logical blocks, whose hierarchical architecture was implemented through the BDDs, are connected in terms of data exchanged and mutual interfaces. Independently from the tool used for the Physical Analysis, generally, an update of the data model of the Logical Analysis is required, to perform the export, as, in this case, is directly done towards the Simulink® model.

Some main model elements shall be translated into physical components, as diagrams, blocks and ports. A proper format shall characterize these elements to be recognized, during the export operation. The Rhapsody® tool includes a specific model *profile* which includes the *stereotypes* necessary to adapt the data model for Simulink® export.

Particularly, it may be remarked some detail.

- If a block is identified by an IBD, thus containing other low-level blocks, represented as parts within the diagram, it shall be characterized by the ≪Structured Simulink Block≫ stereotype, to indicate that it is a sort of container of other elements.
- The low-level blocks, which do not contain other parts, thus not requiring an IBD to be represented, shall be identified with the ≪Simulink Block≫ stereotype. This allows defining the two concepts of container and of simple element, so that the physical tool can recognize the global system architecture. Basic Simulink® blocks are represented in Rhapsody®, with a default

behavior, i.e. with a standard SMD generated when the stereotype is set. In the test case, this feature is not implemented, since the behavior of blocks will be described directly in the Simulink®.

- If blocks stereotypes are instantiated in a correct way, ports and signals are exported with no need of a specific update. The final model in Simulink® will represent the ports and their connections as they appear in the IBD.

The way in which the diagrams appear shall not be modified, but the way followed to export the elements shall be carefully defined. It is always recommended to start by exporting data from the low-level elements to structured blocks. If this action is not performed correctly, the Simulink® might be unable to generate the diagram, because of a lack of data, i.e. high-level blocks will not have low-level diagrams fully characterized. In practice, for the electro-thermal IPS case study, this means that, for example, the logical blocks representing the control unit, control panel and monitoring panel shall be exported, as some simple Simulink® blocks, before the subsystem containing them, as the control and monitoring subsystem. Moreover, blocks shall be contained within a dedicated package to be exported. A direct export of a block located under another block within the Rhapsody® model tree is not yet possible.

It looks nice creating a copy of the logical breakdown and to organize it through some packages, divided by levels of the system architecture. This process shall be performed for all blocks. During the export process, a flow port must be connected to another flow port only. Multiple connections create problems, while trying to replicate the diagram. The use of simple signals is suggested, more than structured signals, carrying more than one variable.

Finally, some comments can be made looking at the diagrams generated in Simulink®, as Fig. 8.20 shows, concerning that derived from a high-level IBD of the electro-thermal solution.

Despite the format, the content is the same of Fig. 7.22, described in Chap. 7. These blocks can be populated with low-level and dedicated behavior diagrams, depending on the system architecture, like in Fig. 8.21, the low-level diagram of the control and monitoring subsystem.

Diagrams can be directly derived from the Logical Analysis or even new elements can be included manually. In this case, a preliminary model is enriched with some new or modified blocks.

The detailed implementation phase starts here and consists of an advanced physical modeling process, that will lead to the final simulation of the whole model. The system architecture may be deeply affected by this phase. This happens because, during the Logical Analysis, the system architecture has been hypothesized, just looking at funtions. When physical contents are concerned, the view may change and, consequently, the whole structure of system may be impacted. The interoperability between the Simulink® and the Rhapsody® assure a backward synchronization, thus allowing exporting data from the first tool to the second one. Figure 8.22 shows in detail the actuation subsystem, as is implemented in Simulink®.

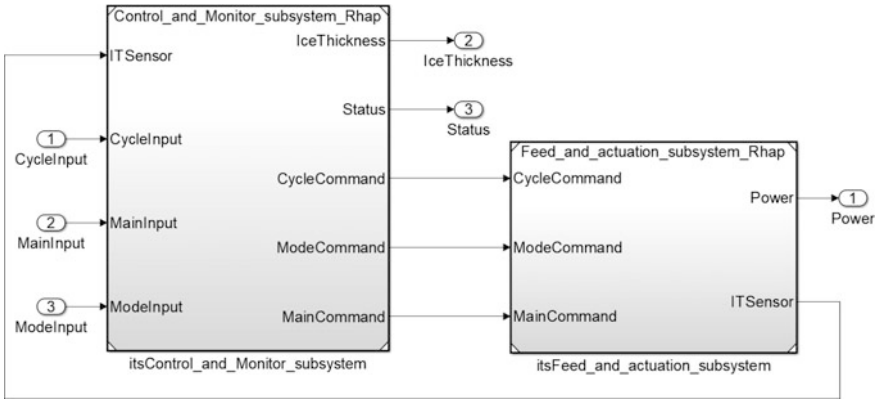


Fig. 8.20 High-level preliminary Simulink® diagram for the electro-thermal IPS

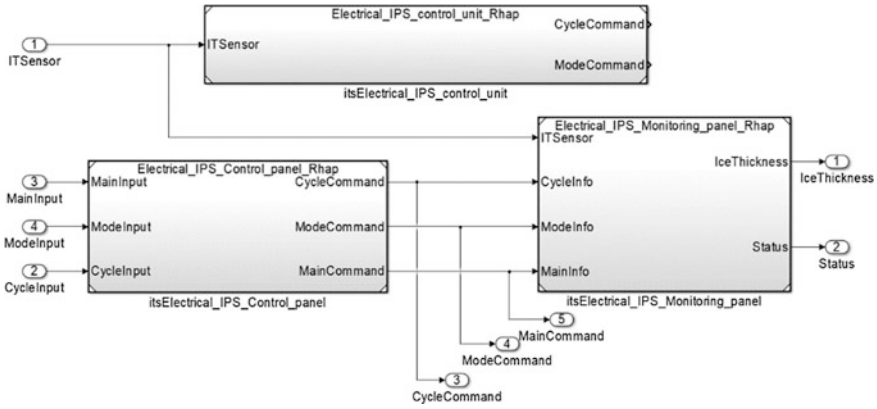


Fig. 8.21 Preliminary Simulink® diagram for the control and monitoring of electro-thermal IPS

Controllers send some signals to activate the heaters represented by green blocks. A value of specific power is selected, depending on the cycle. The output is the power consumption, whilst the sensor is not included in this view. Timing, specific power and surfaces data are all coming from the sizing model. Blocks are derived from the Logical Analysis, thus assuring consistency of data. A simulation can be then performed, to evaluate the power consumption and ice melting capabilities. Figure 8.23 shows the predicted power consumption. The peak is considerably higher than that hypothesized by the sizing model. The small appendices increase of about 5 kW the total power, since some additional continuous heaters shall be considered. However, the most impacting phenomenon is the superimposition of heat-on and heat-off transients of heaters. Notably, since they provide a high specific power for low time, the total power peak is almost doubled, if

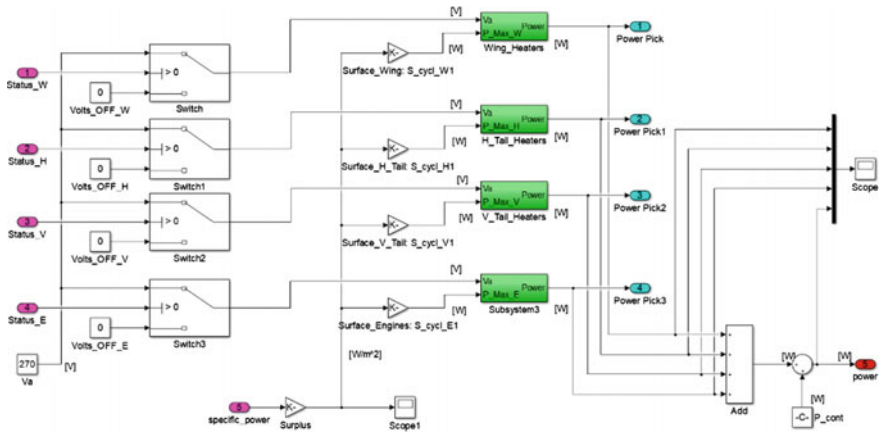


Fig. 8.22 Final aspect of the Simulink® model for actuation of the electro-thermal IPS

compared to previous one. This is a typical dynamic effect, which cannot be evaluated by the sizing model. Power is the sum of several contributions and the average value is never zero, because of the continuous heating process, which motivates the power trend, never starting from zero.

In Fig. 8.24, the ice melting rate for the considered zones is shown. The peak is about 0.16 mm/s, considerably high if compared to a typical accretion rate. The melting efficiency is high and the ice is reduced, after a small number of activation.

The accretion rate is, actually, an order of magnitude lower than the melting one. Estimations obtained from models indicate 0.05 mm/s, in case of the Appendix O environmental boundary conditions and 0.012 mm/s, considering the Appendix C (EASA, 2017).

Requirements traceability within the Physical Analysis is an effective process aimed at assuring the consistency of data, with the functional modelling, especially considering the satisfaction dependencies of the Logical Analysis, and at preparing the verification activity. Linking requirements to physical model elements helps to identify where they will have a relevant impact, in terms of allocation, and to verify their compliancy.

The Simulink® and DOORS® can be interfaced by enabling some embedded features of Verification and Validation toolbox (Mathworks, 2014). This allows to navigate the SRS, which remains stored in the DOORS®, directly from the Simulink® environment, linking the requirements to every model element. As an example, it is possible to consider the requirement related to the maximum power consumption. The requirement 133 originally fixed a maximum power peak of 65 kW, thus allowing a sort of tolerance of about 20 kW over the derived results, obtained through the sizing model. In this case, the power peak derived by the dynamic model is about 70 kW, which is no longer acceptable. This requirement was originally allocated on the main logical block of electro-thermal IPS, but, now,

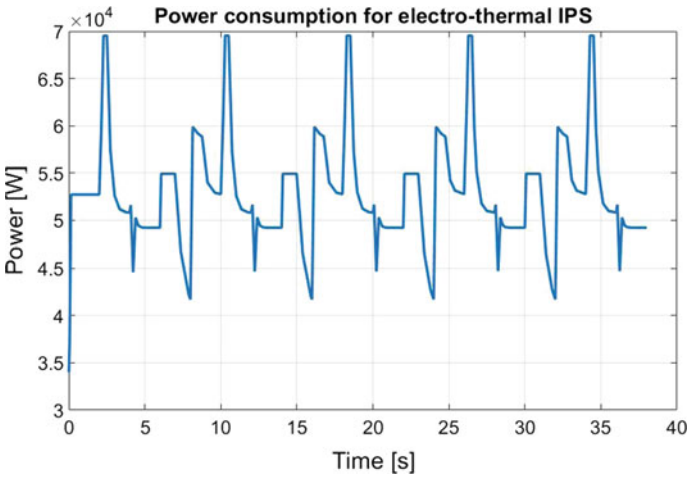


Fig. 8.23 Power trend over the activation sequence for the electro-thermal IPS

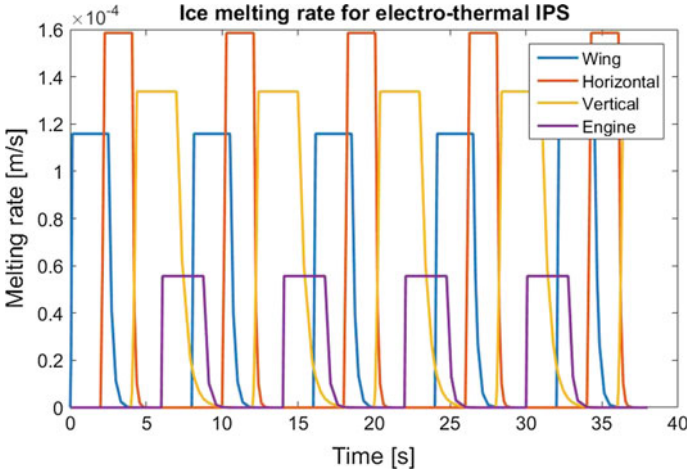


Fig. 8.24 Ice melting rates for the different aircraft zones, for the electro-thermal IPS

it is possible to locate the source of problem within the physical model. It is the power signal coming out from the heaters and constituting the main output of Fig. 8.10 (red on the right). Therefore, an output number 5 of actuation subsystem of the electro-thermal IPS model violates the requirement 23. Some solutions can be applied, to fit that requirement. The electrical inductance of components can be changed, to reduce its electrical inertia to the variable current fed. The system can be differently designed, to find a better timing or a more suitable value of specific power, for heaters. Figure 8.25 shows the typical window of the software tool,

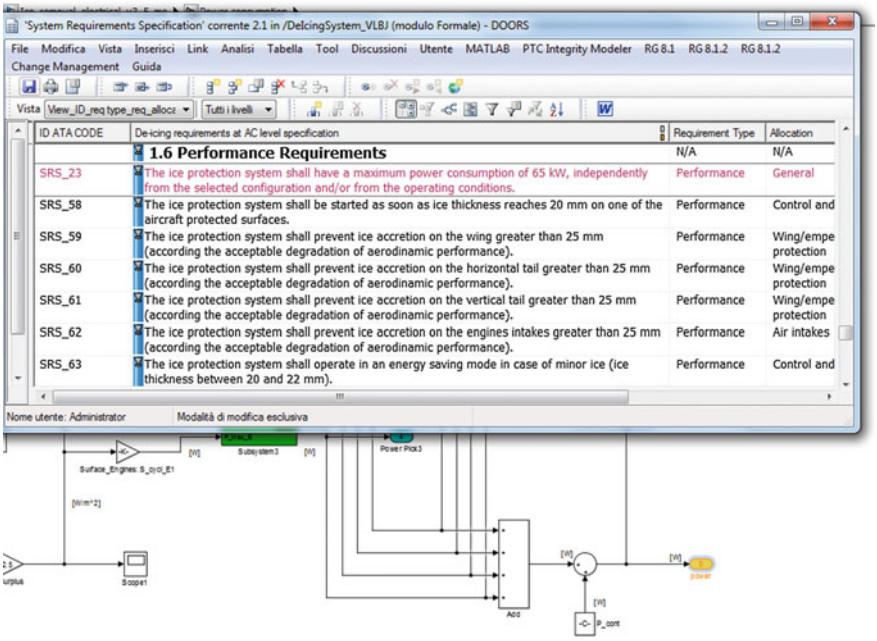


Fig. 8.25 Requirements allocation within the Simulink® environment

where requirements can be navigated and the connection with physical model element is built. Blocks and requirements are highlighted automatically, to show when the link is active.

Dynamic model for the pneumatic IPS based on inflatable boots. This model is aimed at assessing the deicing efficiency of the pneumatic IPS, based on inflatable boots. It evaluates also the power consumption and the dynamic behavior of boots. The control strategies to actuate the boots are even verified, during the simulation.

The approach used to implement the model is very similar to previous one, applied to the electro-thermal IPS. It includes some phases, namely, the preliminary, detailed and requirements implementation. The same tools are exploited and the same rules concerning the format of data coming from the Logical Analysis (IBD parts, stereotypes and ports) are applied.

This example points out how the hand-off between the Logical and Physical analyses is performed. Few modifications have been applied to the IBD, to perform the simulation. Figure 8.26 shows the high-level diagram for the pneumatic IPS, as imported in Simulink®.

It looks similar to the diagram of the electro-thermal IPS. The distribution system is different, since it applies a pneumatic power source, and the control logics is here adapted to the test case. In Fig. 8.27 the distribution sub-system is detailed.

The main command switch is connected to the distribution line, because a shut-off valve is used to activate and de-activate the pneumatic system. The airflow

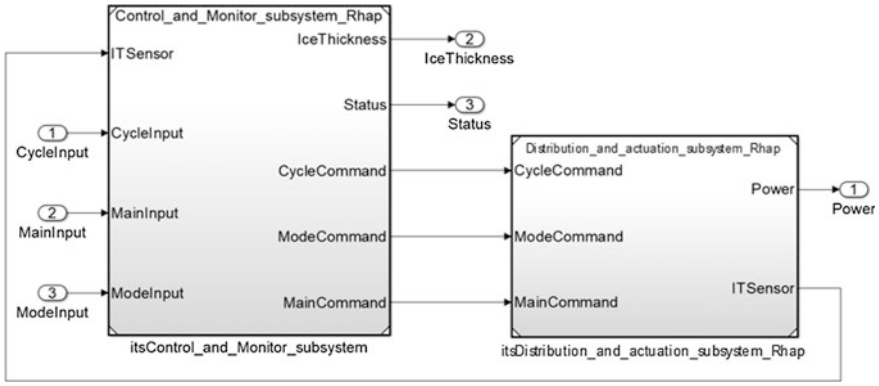


Fig. 8.26 High-level preliminary Simulink® diagram for the pneumatic IPS

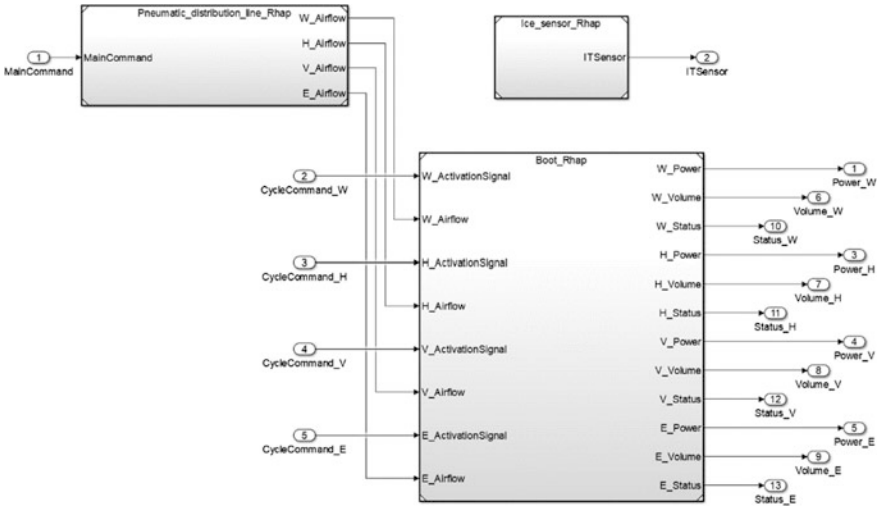


Fig. 8.27 Preliminary Simulink® diagram for distribution and actuation subsystems of the pneumatic IPS

is sent to boots. Depending on the selected cycle and control law, the status of each boot is identified. The power consumption to break the ice is evaluated and compared to airflow and pressure required to increase sufficiently the volume of boots. The ice sensor is connected to the control system, to send the required information about ice thickness. The structure of boots can be detailed and their architecture can be prepared for the Physical Analysis. This is done because some mathematical issues will be added, needing some refinement of the model configuration and of ports layout. Figure 8.28 shows the preliminary Simulink® model of boots of the pneumatic IPS.

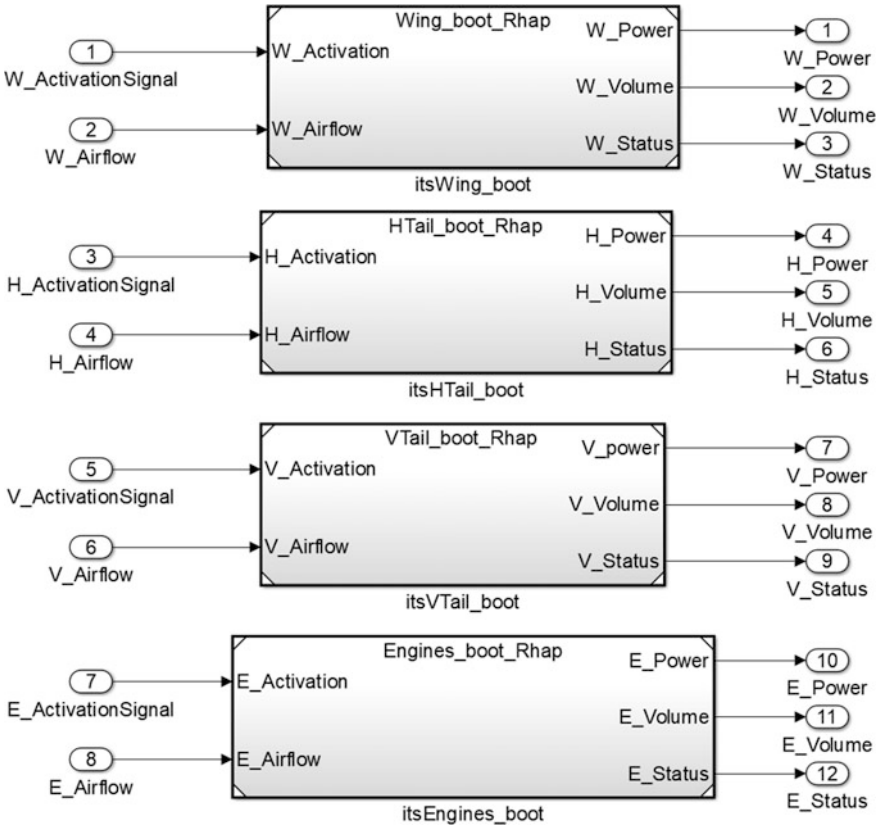


Fig. 8.28 Preliminary Simulink® diagram for boots of the pneumatic IPS

Each boot receives a continuous signal, which is a physical entity, as the airflow (in kg/s) and a discrete signal, being the control input, which triggers the actuation. Boots are then characterized by a status, which identifies the working condition, as well as by the requested power and volume, varying during the activation cycle.

Switching to the detailed phase in Simulink® view, it is possible to look at the main diagram, as specified by Fig. 8.29.

In this case, the elements specified by the original IBD can be easily recognized, although they appear organized in a different way. Controller (light blue on the left), valves from the distribution subsystem (pink in the center) and actuation boots (on the right) constitute the core of the diagram. Moreover, the distribution system is further detailed to characterize the DDV assembly, as shown in Fig. 8.30 for wing and horizontal stabilizers.

As it can be seen, the orange blocks, representing the DDVs, are controlling the airflow by a sequenced actuation, implemented through a series of conditional operators. Even boots are characterized by a second order mathematical derivative

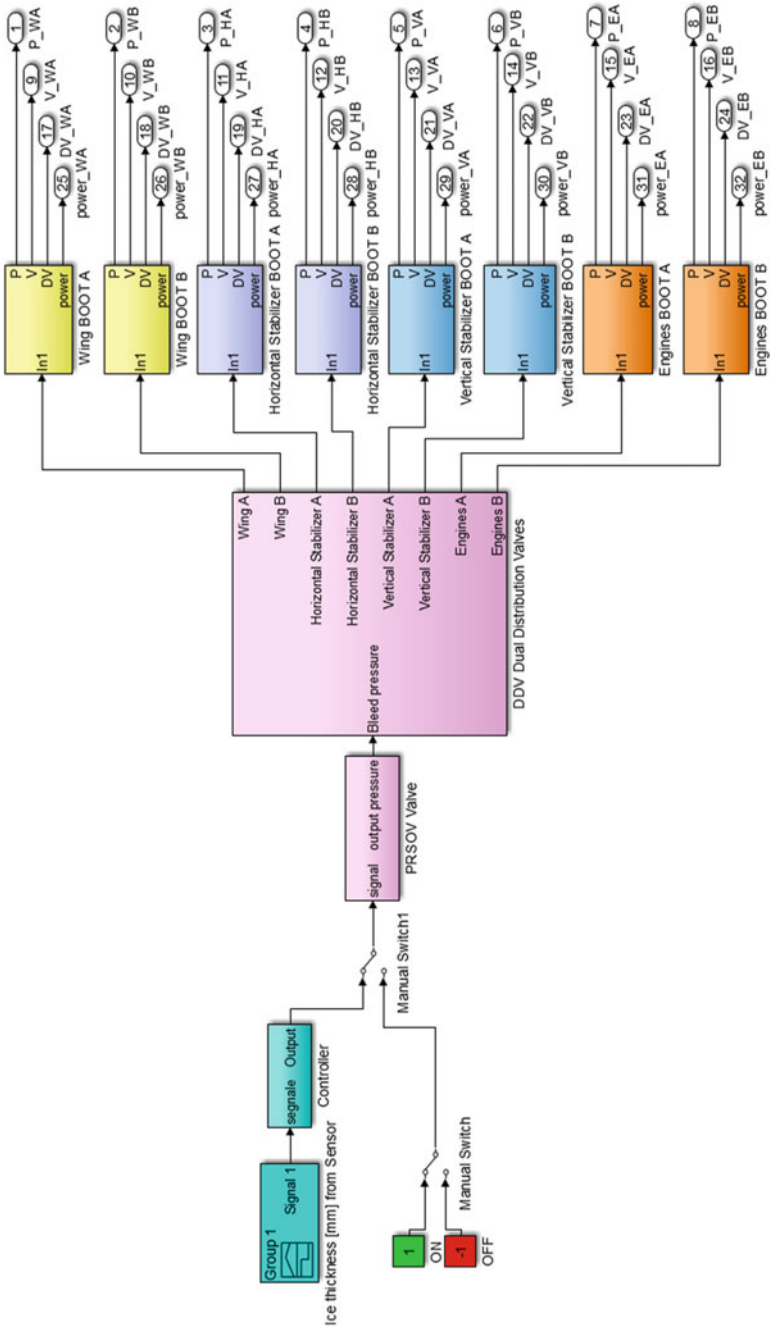


Fig. 8.29 Final aspect of the main Simulink® model for the pneumatic IPS

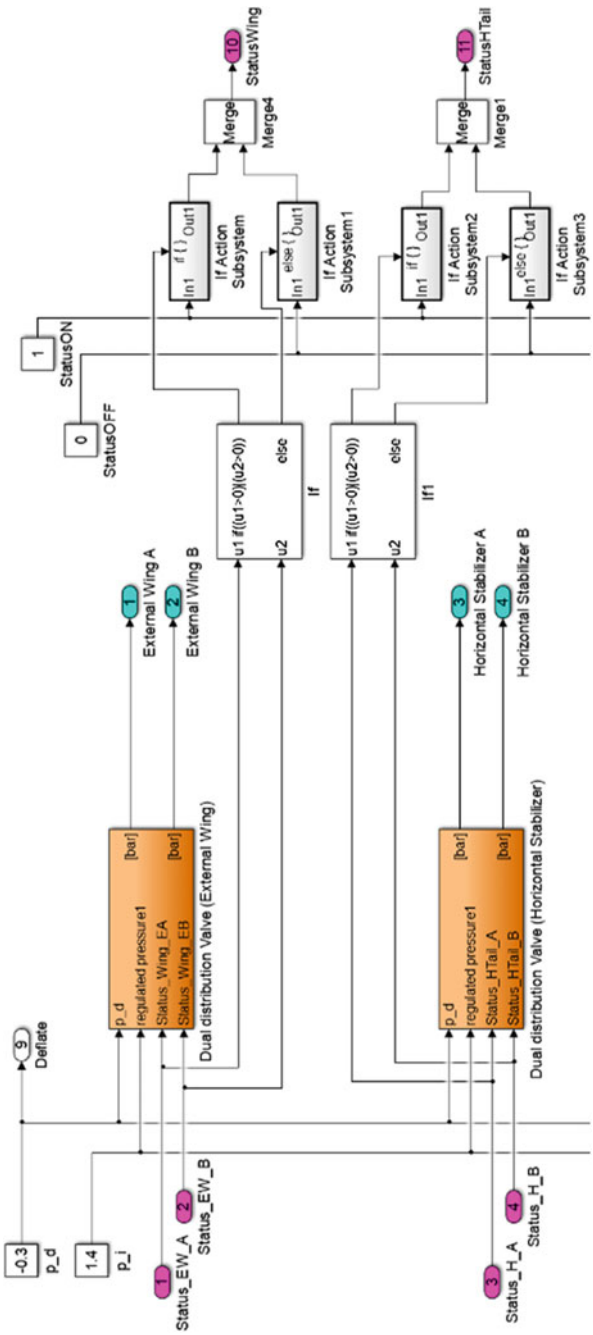


Fig. 8.30 Final aspect of the Simulink® model for the DDVs of the pneumatic IPS

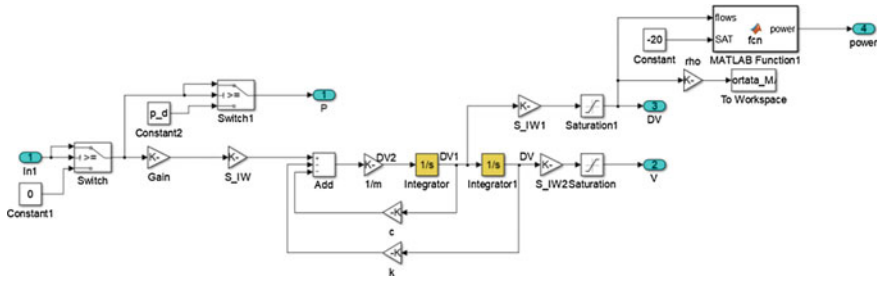


Fig. 8.31 Final aspect of the Simulink® model for a boot of the pneumatic IPS

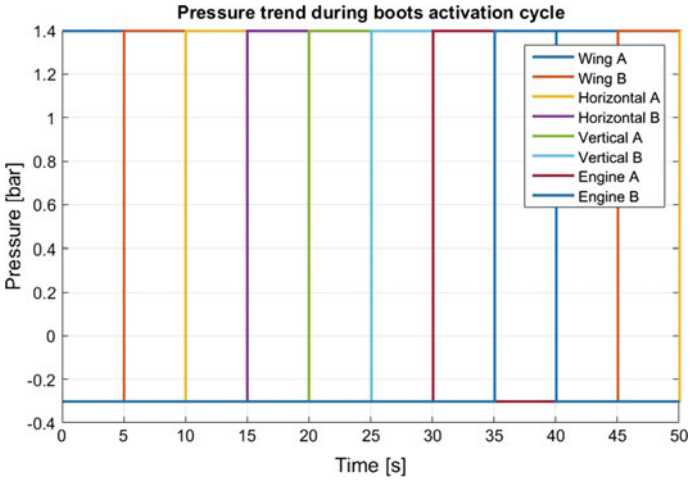


Fig. 8.32 Pressure trend for the different boots of the pneumatic IPS

equation, like the very well-known mass-spring-damper model. Main targets of the analysis are here the inflation rate and the volume evolution (Fig. 8.31).

Considering the numerical simulation, performed by running the detailed model just observed, it is possible to proceed as for the electro-thermal IPS. The trend of main variables can be provided as a function of time. Variables here analyzed are different, because the system dynamics is explored. The ice melting rate, related to the braking action of boots, in nominal conditions is very high. This makes rather difficult using charts to analyze this specific variable. Power consumption, volume, air flow and pressure trends are analyzed, as results of the dynamic model simulation. Figure 8.32 shows, for instance, the trend of pressure.

Relative pressure is regulated and controlled almost in real time, in inflation (1.4 bar) and in deflation (−0.3 bar). This is not possible for the airflow, which exhibits a certain time delay, especially during the deflation. A maximum peak of

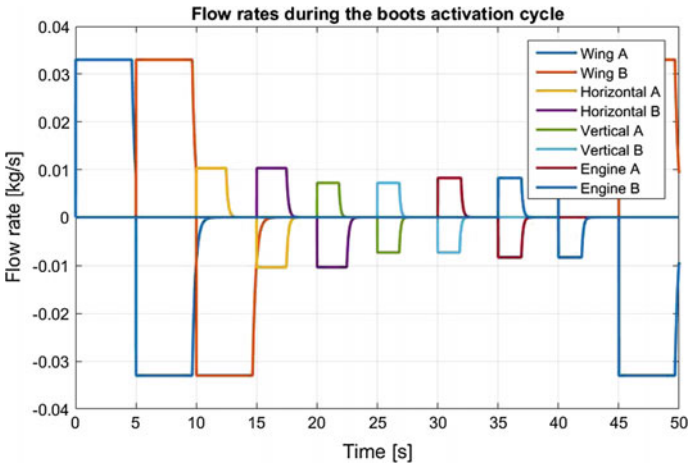


Fig. 8.33 Flow rates for the different boots of the pneumatic IPS during actuation

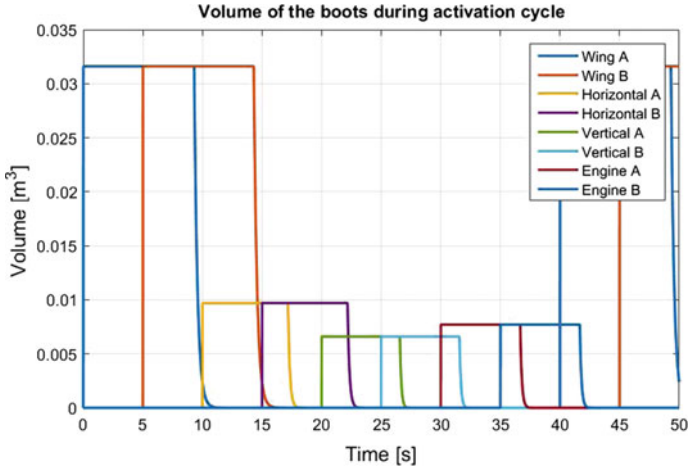


Fig. 8.34 Volume trends for the different boots of the pneumatic IPS during actuation

about 0.03 kg/s (Fig. 8.33) is found. Rates are saturated, when they reach a defined threshold.

The trend of volume depends on the airflow and on the applied pressure. Results are shown in Fig. 8.34. Values are once again saturated, because of some limitations of the rubber material, occurring when a threshold value is reached.

As it can be observed, wing boot is the biggest one, with a maximum volume of 0.03 m^3. The power chart looks relevant. Figure 8.35 shows the first activation cycle and the beginning of the second one.

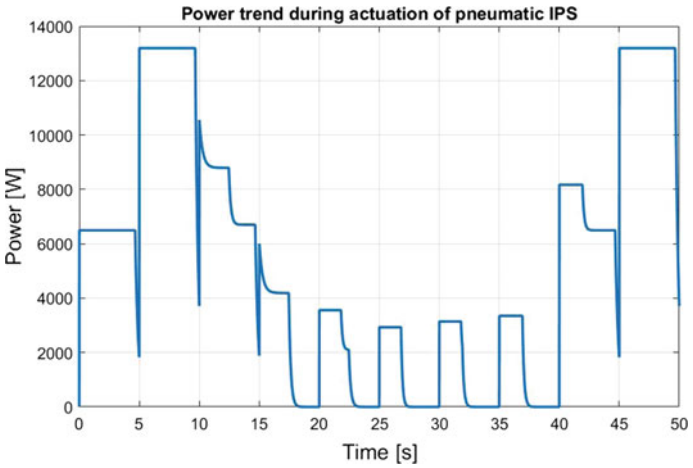


Fig. 8.35 Power trend for pneumatic IPS actuation

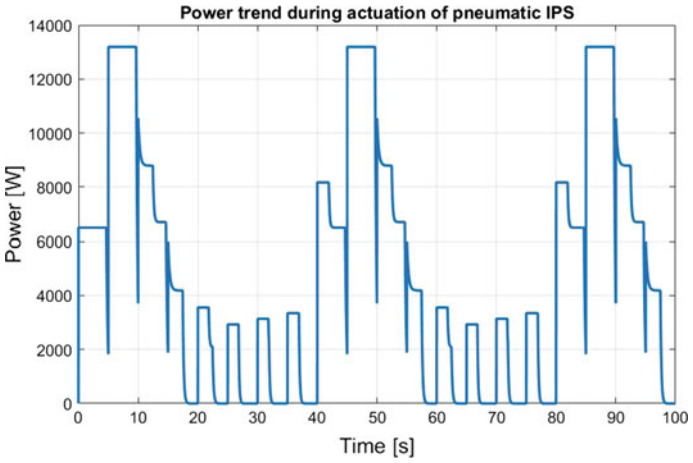
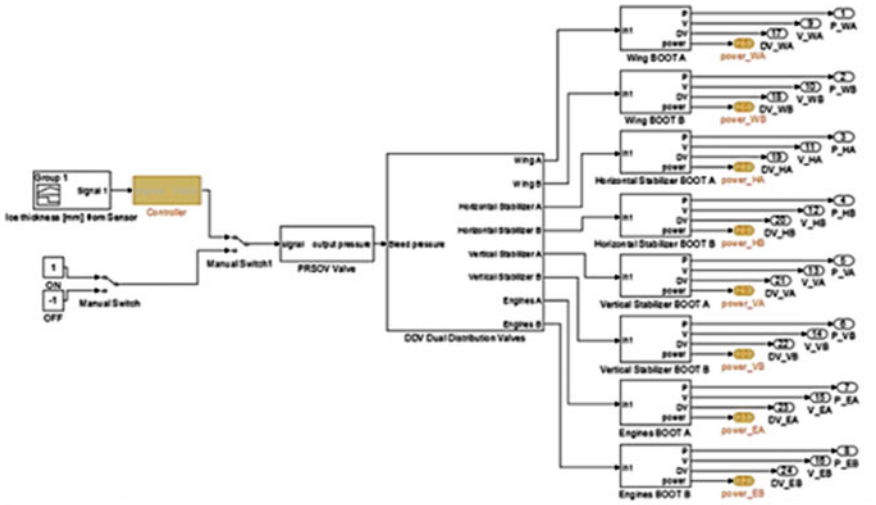


Fig. 8.36 Power trend for pneumatic IPS actuation

The maximum peak is about 13 kW and is reached by wing boot B. The first actuation shows a lower value, if compared to the following ones. This is clearer if three cycles are examined (Fig. 8.36).

During the first actuation, this boot is the first to be inflated, but later, the last boot actuated is deflating, while the first one is inflating again. A superimposition is generated, causing the higher value of power detected, which is not the highest one. As it happened for the electro-thermal IPS, this detail could not be captured by the static models, while the dynamic simulation is suitable to find these phenomena.



Linked Object	Requirements Traceability Data
power_EA	1. "1.6.0-1 The ice protection system shall have a maximum power consumpt" object 133
power_EB	1. "1.6.0-1 The ice protection system shall have a maximum power consumpt" object 133
power_HA	1. "1.6.0-1 The ice protection system shall have a maximum power consumpt" object 133
power_HB	1. "1.6.0-1 The ice protection system shall have a maximum power consumpt" object 133
power_VA	1. "1.6.0-1 The ice protection system shall have a maximum power consumpt" object 133
power_VB	1. "1.6.0-1 The ice protection system shall have a maximum power consumpt" object 133
power_WA	1. "1.6.0-1 The ice protection system shall have a maximum power consumpt" object 133
power_WB	1. "1.6.0-1 The ice protection system shall have a maximum power consumpt" object 133

Fig. 8.37 Summary of traceability links among requirements and Simulink® model elements

Considering the requirements allocation phase, a similar approach can be followed to connect the model elements to the SRS. The power level is fitting the requirement 23, already applied to the electro-thermal IPS. An additional feature of the Simulink® traceability tool is the generation of reports, which include the system requirements. This possibility allows highlighting the model elements including some statements, which provide a clearer perception of link between the SRS and the dynamic model. As an example, Fig. 8.37 shows the report generated for the dynamic model of the pneumatic IPS, where links to the SRS objects are expressed for each model element.

8.7 Results and Final Considerations About the Physical Analysis

The Physical Analysis somehow concludes the path shown for the two test cases. It provides a complete characterization of the physical behavior of the system previously determined, in terms of architecture, through the sequence of Operational, Functional and Logical Analyses. All along the process, requirements have been updated and their allocation has been evaluated, in terms of satisfaction and, where possible, verification through models.

The suggested approach is intensively recursive, since it consists in multiple exploitations of the design process, from high to detailed level, from system to component level. In this handbook, just a single run has been shown, although in both test cases, system, subsystems and components have been considered.

Physical modelling is extremely wide and cannot be summarized in the context of this handbook. The main aim is here showing some details of the formalisms and of the implementation performed within the frame of the industrial production, as the systems described by the examples show. A key issue is the connection between these analyses and logical one, constituting a seamless MBSE design approach. Above all, allocation and traceability properties are assured from the logical architecture to physical simulation. Particularly, the examples show some of the most typical models used within the simulation of mechanical and aerospace systems, although many others could be introduced, applied to several technical domains. The implementation is performed within a proprietary environment, as a stand-alone format. A connection with the requirements database is made by some connectors, already available out-of-the-box for the aforementioned software, as embedded features.

As the next chapter shall describe, this is a simplified view of the physical modeling, still far from a complete interoperability among the analysis tools. In fact, considering the multidisciplinary fields encountered during the design of such complex systems, many tools are generally used to study specific peculiarities from different point of views. Engineering domains are traditionally prone to exploit some typical toolchains, composed by some software conceived for specific purposes. Their integration is still an issue, especially when tools were not originally conceived to communicate. Often the interoperability among tools is a required enhancement to empower the entire process, allowing also the exchange of data among different disciplines and people.

Next chapter shall focus on a *heterogeneous simulation* approach, which allows integrating, with many possible strategies, model elements directly within a host environment, able to act as a sizing or dynamic model. The goal is reducing as much as possible the workload required for tuning data, when switching from an environment to another one, In Chap. 9 strategies and limitations of the interoperability among software are explored and an alternative approach to implement the Physical Analysis just described is investigated.

References

Brusa, E. (2016). *Meccatronica strutturale: Sistemi e tecnologie*. Torino, Italy: CET.

Chiesa, S. (1995). *Impianti di bordo per aeromobili: Impianti pneumatico, condizionamento, anti-ghiaccio e A.P.U.* Torino: CLUT.

EASA. (2017). *Certification specifications and acceptable means of compliance for large aeroplanes—CS-25*. Cologne: EASA. https://www.easa.europa.eu/system/files/dfu/CS-5%20Amendment%2019.pdf. Accessed 28 September, 2017.

FAA. (2002). *Effect of residual and intercycle ice accretion on airfoil performance*. Washington D.C: FAA.

FAA. (2012). *Aviation maintenance technician handbook-airframe. Vol. 1. Ice and rain protection*. Oklahoma City: FAA.

Genta, G. (2005). *Dynamics of rotating systems*. New York: Springer.

Goraj, Z. (2004). An overview of the deicing and antiicing technologies with prospects for the future. *24th International Congress of the Aeronautical Sciences.* Yokohama, Japan.

Mathworks, T. (2014). *Simulink® verification and validation products.* https://uk.mathworks.com/products/transitioned/simverification.html. Accessed 28 September, 2017.

Chapter 9
Heterogeneous Simulation

Abstract As a matter of facts, interoperating tools and integrating the functional and physical modeling of systems within a unique toolchain is one of the most challenging issues of the Systems Engineering technology as is currently known. Therefore, this topic is analyzed in this chapter, one that the set of analyzes required was performed. Strategies, tools, limitations and benefits of the heterogeneous simulation are here analyzed, even through the two test cases.

9.1 Introduction

As it was discussed in previous chapters, the MBSE is currently the most effective approach to assure the traceability of data, in design activity, especially when a toolchain of software is exploited. However, the wide availability of commercial tools, conceived to provide different capabilities within several domains of engineering, leads very often to some incompatibility problems. This is well-known, since tools are seldom designed to communicate with each other. Therefore, an issue for the MBSE implementation is the so-called *interoperability*. It turns out, in this context, into a capability of software to share some data with other software, through a standard form, to allow the user working and accessing to those data, by the toolchain. A low degree of *interoperability* could be the most critical bottleneck in implementing the model-based design process.

Several *integration strategies* can be selected to raise the level of interoperability of a tool, depending on its features and the goal of the analyses performed. Many software tools use some connectors or adapters, expressively designed to allow the direct communication with other ones. Otherwise, a suitable integration is required. Very often the user must develop some import/export connection between tools, when their interoperability is not yet provided by vendors.

Unfortunately, within the MBSE there is still nowadays a high heterogeneity of tools, and integration capabilities need to be provided. Some standards for model exchange are already available and help the user to overcome the difficulty of resorting to the point-to-point connectors.

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_9

Some basic concepts and examples about this topic are proposed in following sections. To simplify the discussion, only the industrial test case is exploited to show some examples of software integration, through the interoperability tools and connectors.

9.2 Strategies of Model's Integration Within the Heterogeneous Simulation

Engineers are usually prone to use very specific software, developed within some specialized technical domain, which characterizes their expertise. These tools enable an effective implementation of the analyses to be carried out, and often benefit of a consolidated tradition of that domain of application. For the user changing the modeling approach to implement the MBSE often looks rather difficult. Moreover, sometimes changing the tools is even required, when a large toolchain is considered. In addition, the introduction of several layers of analysis, as it is done when resorting to the Operational, Functional, Logical and Physical modeling, might excite a certain resistance of operators. The interoperability of tools becomes a strategic mean to avoid all the drawbacks of that change, since it allows connecting the software tools traditionally applied to the quantitative design and the new ones, aimed at enabling the functional modeling.

Forcing the user to master a completely new toolchain is impractical, even because of waste of time and resources for training. Moreover, in large industrial consortia it may be required to interface different tools, to perform the same kind of analysis, because of legacy of each partner. Therefore, adapting the toolchain to support this methodology, through the interoperability of tools is the strategy most applied. In this way, each user may continue working with known tools, just enriching the chain by adding some other ones to enable the development of other analyses, and sharing data with other specialists. Simultaneously, the models exchange and the collaboration within the design activity are both improved.

The so-called *heterogeneous simulation* is expressively aimed at integrating models in another environment, able to host them. It makes seamless the tools integration, since the user navigates through them quite easily. The workload of operators is usually reduced and synergy is improved.

The interoperability effectiveness strictly depends on the software tools interconnected, on the analyses to be performed, and on the toolchain designed case by case. Some strategies can be identified among various possibilities proposed by the software engineering, shown in Fig. 9.1.

Replicate model structure. This is a basic approach to integrate models. It is based on the exchange of entire models, from source to target software, respectively. This strategy allows replicating the original model in another environment, with a reduced workload for the user, since data are compatible. All the features of the original model are maintained. This strategy can be exploited following two implementations, herein described.

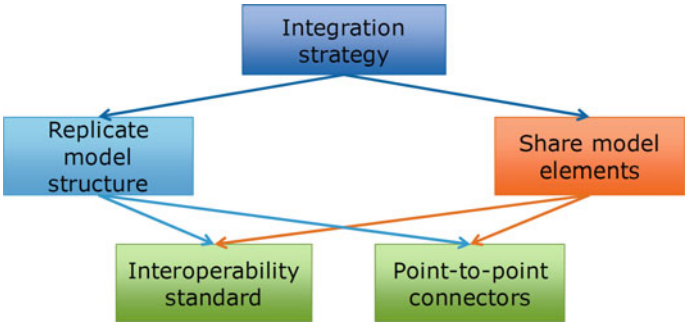


Fig. 9.1 Strategies for models integration

- Point-to-point connectors—this is the simplest way to exchange an entire model. It is based on some dedicated import/export facilities. The source tool exhibits some integration capabilities with a selected list of other ones, able to accept its data (Fig. 9.2). In this case data do not need to be represented in a standard format, being just compatible with the target tool, eventually through some proprietary algorithms able to reorganize their formalisms. The model exported in the target tool is a representation of the original one, although its formalisms is adapted to the hosting environment. Practically speaking, the data exchange process is a sort of *translation*. This solution is generally implemented as a *synchronization* activity. Model elements shall be updated, in case of modification, to keep the consistency between environments. Replicability issues may generally arise, when performing the synchronization, because the number of formats supported for the data exchange may be limited. Often the target tool does not support the formats of the source tool. Therefore, a specific connector is developed. The reliability of adapters as well as the stability of the exchange

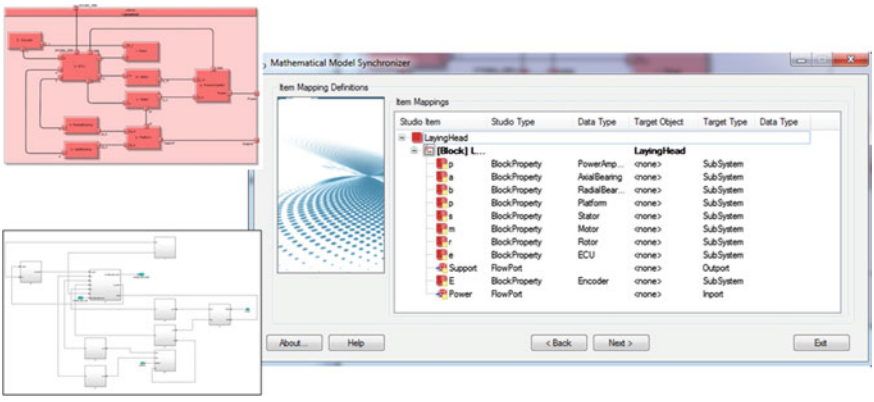


Fig. 9.2 Mathematical model synchronizer within the PTC modeler® traducing IBD in the Simulink® model

process may be an issue. Unexpected crashes or incompatibility issues among different versions and releases of software may be found.

- Interoperability standards—the tools integration strategy can be implemented by resorting to some interoperability standards, to overcome the need of specific point-to-point connectors. The strategy is similar to previous one for the integration process, performed through the *synchronization* of data. The translation is here no longer required, since data are formatted in a standard way, before that they are exported. The target tool is able to understand this format, never requiring any recompilation. Some other differences are here highlighted. Since this approach considers the exchange of the entire model, sometimes this can appear as a *black box* (Fig. 9.3) in the hosting environment. The interoperability standards often allow protection of data, therefore the exported version of the model allows accessing to variables only, but the internal algorithms are never disclosed. This is very important as far as the cooperation among many partners is concerned and the need of protecting the intellectual property, upon some classified information, arises. The exploitation of standards significantly raises the process stability, enhancing the quality of the synchronization and providing a higher replicability. A major drawback is that both the tools must support the standard. However, advantages are considerably larger than those limitations. Therefore, this technique is highly recommended, when applicable.

Share model elements. This approach works directly on the model elements. The model is interpreted as an instance, composed by various parts, which can be

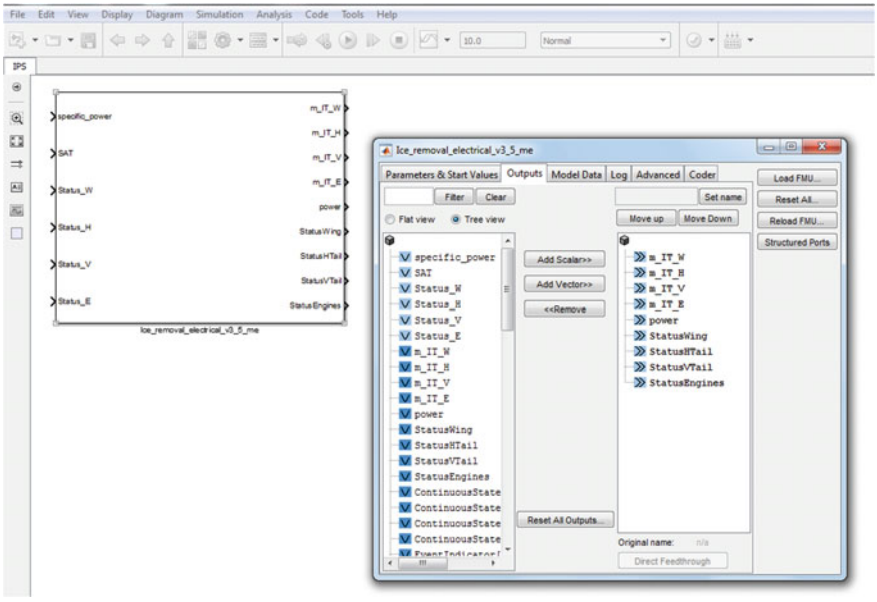


Fig. 9.3 Import of standardized unit in Simulink®

considered separately as independent elements, with specific interfaces, that allow them communicating with other ones. This strategy assures higher flexibility and portability of the model exchange process, being focused on smaller elements, potentially usable within different scenarios. The implementation solutions envisaged are the same of previous techniques. Dealing with smaller parts affects the effectiveness, usually raising the level of reliability of this approach. Model parts are supposed to work independently, when properly connected, isolating specific sections of the behavior of the entire model. This detail is very interesting for the model *verification and validation*.

- Point-to-point connectors—this solution is the same of that above described, but objects to be synchronized are smaller, and the process reliability is considerably higher (Fig. 9.4). This trend is motivated. The entire model is made of different

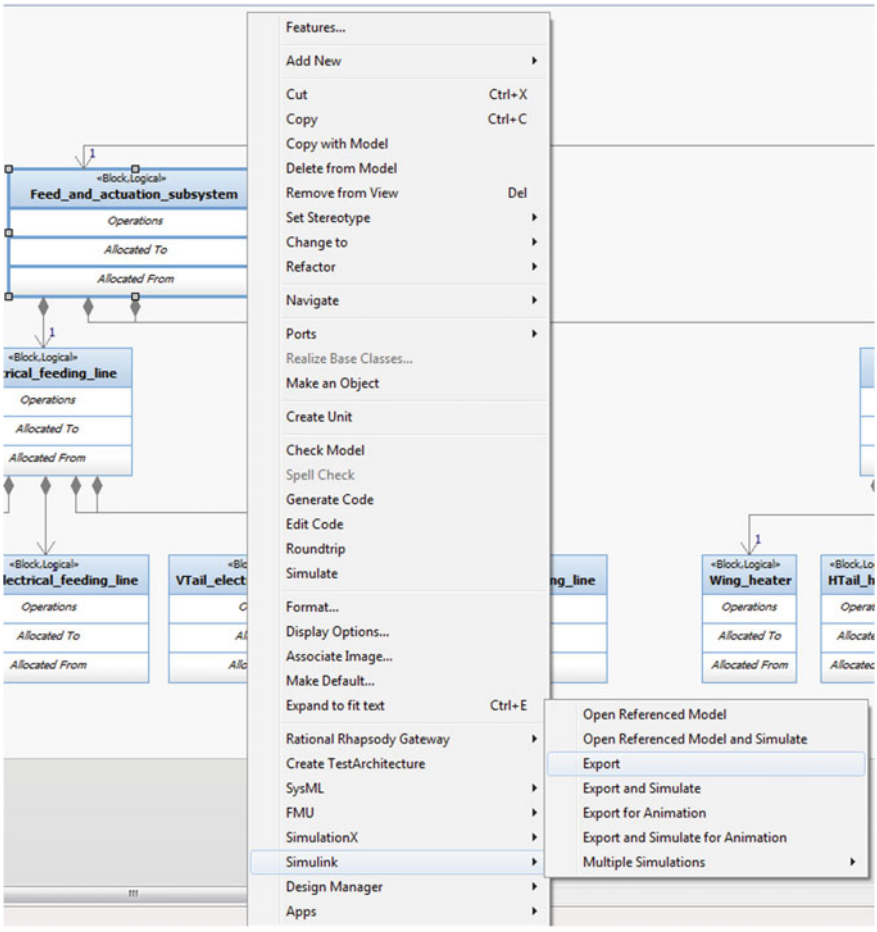


Fig. 9.4 Export of model elements from the Rhapsody® to the Simulink®

levels, being characterized by an internal structure, which is like that described within the BDD and IBD. Thus, several dependencies among interfaces and links shall be synchronized. In case of elemental model parts, the focus is on simpler structures, with reduced number of interfaces, ports and connectors. This makes easier the translation process, reducing also the errors due to unsupported formalisms.

- Interoperability standards—this is the most effective implementation, since it allows the maximum flexibility and a larger replicability. Re-usability is also very high, since model elements can be exploited for different simulation scenarios (Fig. 9.5). Problems related to the black box of the exported objects are here less important, since the percentage of hidden elements is lower. This means that, if a new model is created in the target tool, with proprietary formalisms, and some parts shall be received via standard from a source tool, the model structure can be still navigated, although the model elements imported are hidden.

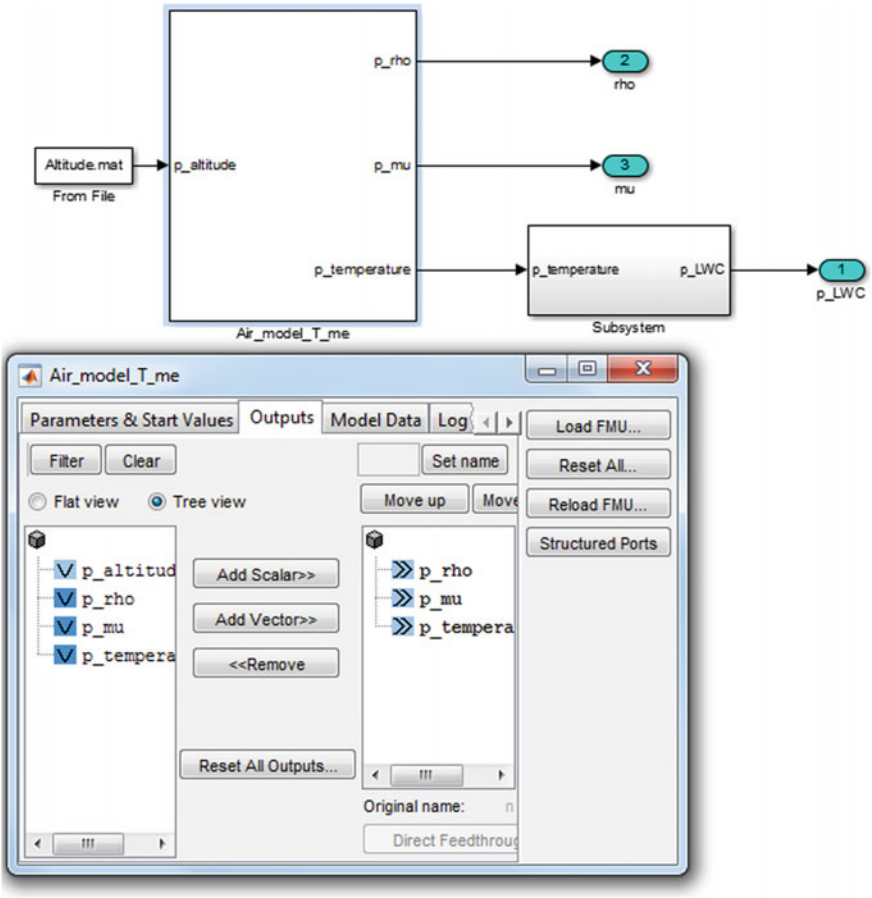


Fig. 9.5 Inclusion of a standardized unit within a proprietary Simulink® model

9.3 Example of Interoperability Standard: The Functional Mock-up Interface

The Functional Mock-up Interface (FMI) (Blochwitz et al., 2011) is a tool independent standard conceived to ease the model exchange among dynamic models, supporting also some *co-simulation* capabilities, through the combined use of XML files and compiled C-code. Born under the supervision of automotive industries, to face the problem of integrating many proprietary tools, and of reusing algorithms for early validation of results, it is now a de facto standard also in other engineering domains. The FMI (FMI, 2017) is based on two main concepts:

- The implementation of a component able, to enclose the proprietary formalisms, supporting, at the same time, the related interfaces, called *Functional Mock-up Unit* (FMU)
- The separation of two actions as the description of data to be interfaced (through XML files) and of the code functionality (binary, C-code).

The FMU is a sort of ZIP file, containing the XML and the binary, being fully characterized and exportable. It represents the core of the original model from which it derives. It can be simulated as a stand-alone component, in a different environment, supporting the standard. The first version of the FMI (1.0) was released in January 2010 supporting the Model Exchange approach (ME). In this case, the FMU contains the model generated within the original environment, but the solver used for the final simulation belongs to the host tool. Another subsequent strategy, referred to as *Co-Simulation* (CS), couples two or more simulation tools (slaves) within a third environment (master). Slaves contain the original models as FMU, but they are solved independently by their original solver. Master algorithms only control the data exchange and the synchronization of slaves. Figure 9.6 summarizes the two approaches.

Co-Simulation scenario offers different possibilities, in terms of implementation on a IT infrastructure. The second process represented in Fig. 9.6 shows only the stand-alone scenario, i.e. a simulation performed within the master tool, that

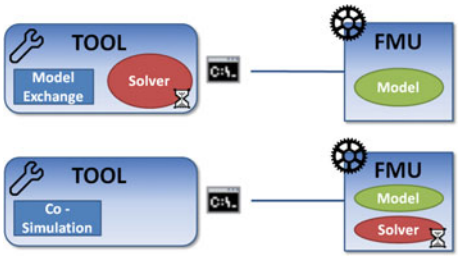


Fig. 9.6 Schematic of model exchange and co-simulation FMI (FMI, 2017)

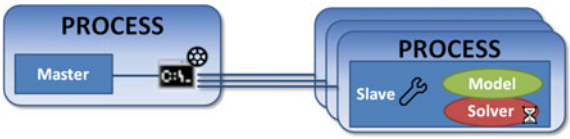


Fig. 9.7 Schematic of co-simulation with different processes running (Blochwitz et al., 2011)

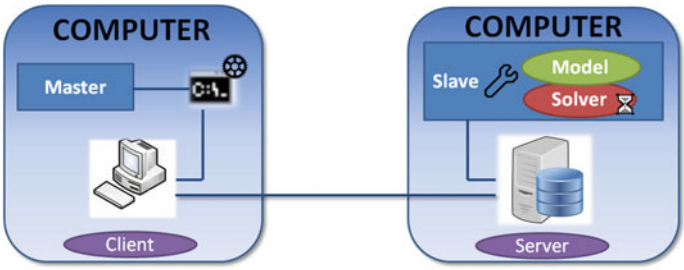


Fig. 9.8 Schematic of Co-Simulation for a distributed simulation scenario (Blochwitz et al., 2011)

contains all the FMU (slaves). However, this is the simplest deployment of software architecture. Often, there is not the possibility or the will of integrating the whole scenario in a single tool, and master/slaves configurations may differ. For example, it is possible to keep the software applications performing master and slaves, running in parallel, in different processes. In this case, a sort of wrapper is necessary within the master process, to integrate the data coming from the slaves, although the rest of the architecture remains the same (Fig. 9.7).

Moreover, if the software architecture is *distributed* among different clients, a more complex scenario shall be envisaged (Fig. 9.8). In this case, an additional communication layer is necessary to handle the communication, based on a session server, managing the simulation scenario.

Considering the simplicity and the effectiveness of this standard, several tool vendors have added dedicated features in their product to support, partially or completely, these capabilities. Its successful spread is also due to the open specification, which is provided as a free resource, potentially upgradable and modifiable. CAD, software for dynamic simulation, multi-body analyses and many others have included import/export facilities for FMI standard, either as embedded feature or through third party connectors. At present time, more than 100 software are supporting the standard, which has been updated to version 2.0 in July 2014.

Following sections provides an example of implementation of the stand-alone process applied to the IPS case study.

9.4 Implementation of the Heterogeneous Simulation in the Ice Protection System Case Study

The IPS case study offers a lot of possibilities concerning model's integration and the exploitation of an heterogeneous simulation. To describe the scenario, several models may be required to represent not only the system behavior, but also the operating environment (atmosphere, ice building physics). Moreover, a further subdivision of the model can be implemented, simulating what really happens in an industrial context, where specialists designing a system are focusing on very specific issues. Some separated models can be hypothesized for the actuation, control and detection subsystems. The mission profile shall be included as well to characterize the behavior of the entire aircraft.

Following sections are conceived to provide an overview of the main models involved within the heterogeneous simulation approach, applied to the IPS case study. Types, proprietary software from which they are built and integration strategy, based on FMI standard, are mainly focused.

9.4.1 Models for the Ice Protection System Scenario

This scenario is based on 12 models, combined within a host tool as a FMU. Simulink® is acting as a main hub for models, which are built in the Matlab®, Rhapsody®, Open Modelica® and Simulink® as well. Successful integrations have been proved with Dymola® (CRYSTAL, 2017) Simulation campaigns have been performed with both the Model Exchange and Co-Simulation FMU. Details about the models implemented are listed hereafter, following the logical order of variables, which flow among them.

Mission profile model. This model is conceived to provide altitude and speed profiles of the aircraft, during the mission. Considering the time supposed to perform the mission, several phases are defined and, notably, taxi out, take-off, climb, cruise, holding, descent, approach, first landing attempt, go-around, second landing attempt and taxi in. It is based on a Matlab® script and implemented in Simulink® through the proprietary S-Function. Optional outputs can be provided as well, like vertical speed, fuel consumption, or load factors, but they are not required for the scenario of the IPS. This model does not require any input, since it is the starting point for the whole analysis.

Atmosphere model. The air model is based on the International Standard Atmosphere (ISA) data for what concerns density and viscosity trends. It requires the altitude as an input, to compute the related profiles. As for the mission profile model, it is based on a Simulink® S-Function implementing a Matlab® script.

Temperature model. The air model may potentially include also the temperature profile, following the ISA data. However, temperature profile is computed

separately by an Open Modelica® look up table. The working principle is the same, thus based on the altitude as an input, but the source code is different in terms of generating environment.

Liquid Water Content (LWC) model. This model implements the LWC trends described within CS-25 (EASA, 2017) appendixes concerning supercooled water. It is based on a Matlab® script, which receives the value of temperature during the mission to identify the correct LWC level, considering that the whole mission may be subjected to icing conditions. It is then implemented, as mentioned before, through an S-Function in Simulink®.

Monitored Temperature model. This is a simple algorithm to simulate the noise generated by the temperature sensor, which receives the real temperature value and computes a result affected by a random error. This value is sent to the IPS controller, which regulates the power intensity, depending on the measured temperature. The model is implemented through a random noise generator built in Open Modelica®.

Ice creation dynamics model. Depending on the density and viscosity of air, the LWC and the speed of the aircraft, received respectively from atmosphere model, LWC model and mission profile model, it is possible to compute the ice build-up on the airfoils. The model considers the characteristics of droplets, which can be derived from the environmental conditions specified by the input, and the related dimensions to derive the impact area and the accretion rate (in m/s). This is the main output of the model and will be compared to the melting rate provided by the IPS to derive the actual ice thickness present on the aircraft zones. One model is built for each zone, so, four units are implemented (wing, horizontal stabilizer, vertical stabilizer and engines). Models are coded directly in the Simulink®.

This first group of blocks is shown in Fig. 9.9, where the FMU built from the original models have been imported in the Simulink®.

Ice accumulation model. This is a very simple model, summing ice accretion and melting rates to compute the ice thickness level. However, it is a fundamental brick of the simulation, since it generates the input required by the IPS controller. Depending on temperature and ice thickness, of the aircraft zones, the controller regulates the power of the IPS, to either increase or decrease the intensity of melting rate. The output of this model, i.e. the ice thickness (in meters), is then used as a main feedback signal to trigger the system. The model is built through some Simulink® blocks.

Monitored ice thickness model. Similarly to what was done for the temperature, even the ice thickness signal coming from the ice accumulation model is combined with a random value to simulate the sensor noise. The output is a value characterized by an error, proportional to the random noise, which is used by the controller to regulate the ice melting efficiency. Blocks are implemented in similar way, through the Open Modelica® random number generators, being customized to match the ice thickness typical values. Again, since four zones are accounted separately, for the ice accretion, four models for the monitored ice thickness are included, within the simulation scenario.

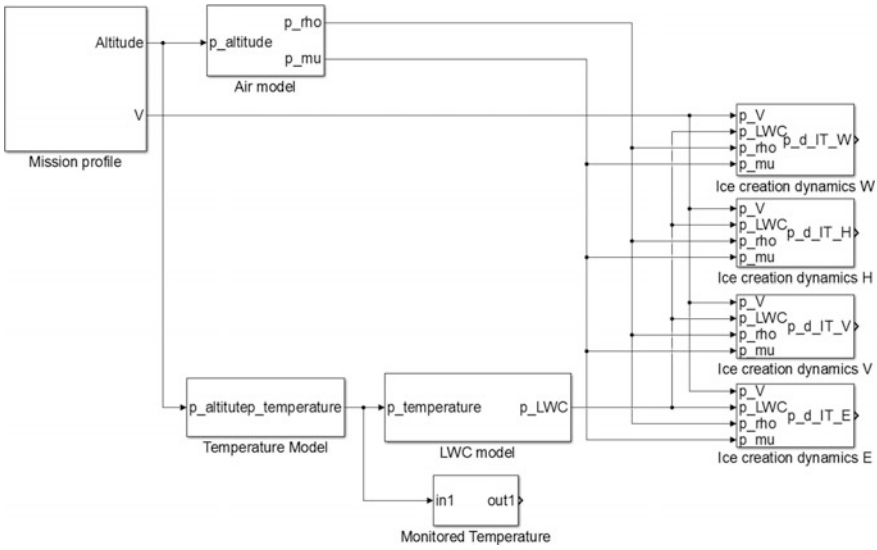


Fig. 9.9 FMU related to mission and environmental data implemented in Simulink®

IPS model. The IPS model depends on the type of system simulated. However, for both electrical and pneumatic IPS, it is made of three execution components:

- Control model—it receives the data concerning the monitored ice thickness and temperature, to induce timing input to the sequencer. For electro-thermal IPS the so-called cycle time is referred to the heat on period of a single zone, whilst for the pneumatic IPS it provides information about the switch off period at the end of cycle, in case of complete ice elimination. The model is built through a discrete event based state machines, developed in the Rhapsody®. These could be interpreted as an evolution of those implemented within the functional analysis for the control use case.
- Sequencer model—this model establishes the timing relationships for the actuation of the different zones, using the cycle time value received from the controller. Actually, it is still a sort of controlling mean, although, in case of electro-thermal IPS, it provides also the trigger for power regulation. The main output is the activation signal to the different zones, for both cases. Together with power regulation, as far as electro-thermal IPS is concerned, the model receives directly the monitored thickness for the wing. This is implemented to reduce the heat-off period of non-wing zones, in case the ice build-up on the wing is negligible. The wing has the highest number of zones, thus requiring a lot of time to complete the whole cycle, causing a high inter-cycle ice accumulation on other ones. This model is implemented in the Rhapsody®, as a statechart, like the previous one.

- Actuation model—this model is strictly dependent on the type of IPS used. It is the core of the whole simulation and may differ very much, depending on the selected case study. It receives the activation input for the different zones and the cycle time. Moreover, to take into account the effects of temperature on the melting efficiency, the real value of the Static Air Temperature (SAT), i.e. the value computed by the temperature model, is provided. In case of electro-thermal IPS, the specific power established by the sequencer is also required as an input. The main outputs are related to the activation of zones, the global power level and the ice melting rates for each zone. Melting rates are compared to the accretion rates, within the accumulated ice models to derive the ice level. Power signal is the main output of the model and, optionally, can be connected to the energy model, which integrates the signal to derive the total energy used to deploy the IPS. Those models are built with Simulink® formalisms.

Energy model. it is a simple integrator, built in the Open Modelica® to derive the total energy consumed, during the mission, to activate the IPS, starting from the power trend provided by the actuation model.

Figure 9.10 shows the final model layout, implemented in the Simulink®, using FMU only. Particularly, the electro-thermal IPS scenario is shown.

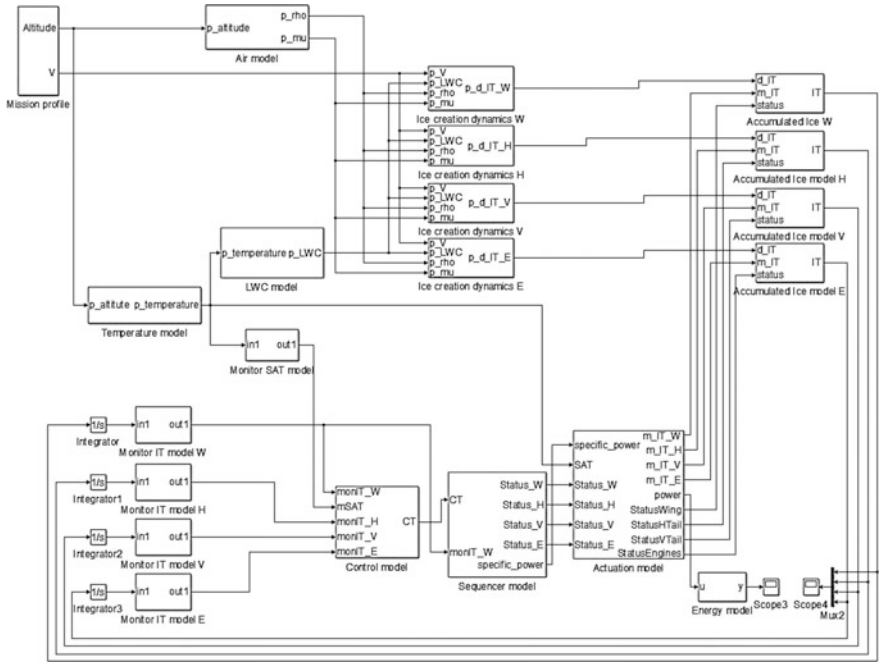


Fig. 9.10 Complete FMU scenario implemented in Simulink®

The results of the simulation run for both electro-thermal and pneumatic IPS are reported in the next section.

9.4.2 Simulation Results and Final Run

The main results obtainable from these models concern the power consumption and the ice thickness level. Other kind of results can be computed, as it was for the stand-alone Simulink® models, coming from the architecture of the logical analysis (Chap. 7), like power per each zone, volume of boots (for pneumatic IPS), accretion rate etc.… It is very important to look for computation problems, compatibility issues or numerical glitches, since models integration may lead to unexpected behavior. Basically, results coming from the heterogeneous simulation shall be compliant with those derived through the classical modeling approach to be considered reliable. The model validation is then fundamental, prior to analyze the results.

Electro-thermal IPS. As it can be appreciated in Fig. 9.11, showing the power trend for the electro-thermal IPS, results confirm the contents of Fig. 8.24 (Chap. 8).

The maximum power peak around 70 kW is confirmed. However, peaks are a little bit different, from the point of view of the wave form, and continuous power level is also different. This is due to the action of the controller, which regulates the intensity following the environmental conditions. This kind of algorithm was not included in previous model, thus making the difference clearer, also in terms of secondary peaks layout. The power peak is compliable, proving the correctness of the model, in both cases. The considered time interval represents the first sequences following the activation of the system, occurring when the ice thickness exceeds a certain threshold. Actually, the ice thickness evolution during the mission can be described, as in Fig. 9.12.

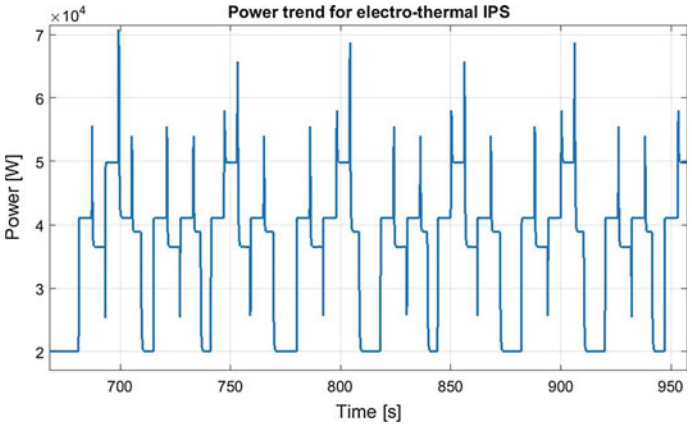


Fig. 9.11 Power consumption for the electro-thermal IPS derived from heterogeneous simulation

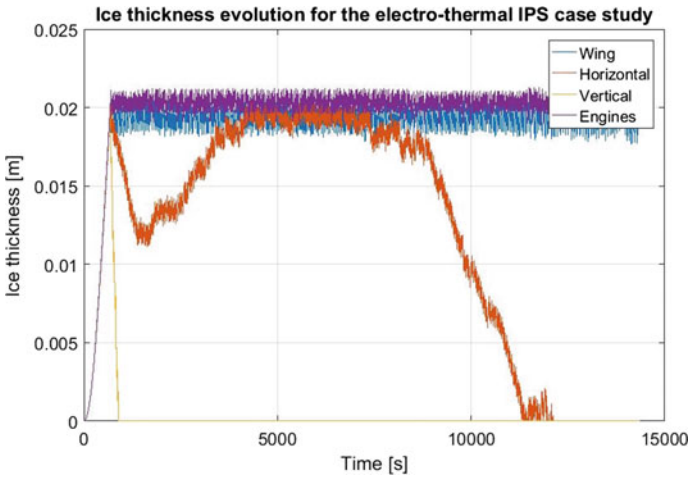


Fig. 9.12 Ice thickness evolution for electro-thermal IPS derived from heterogeneous simulation

The accretion and melting rates, combined together, show that the ice thickness remains under the threshold specified by the requirements although, in some cases, it is not possible to completely melt the ice. In this case, the whole mission is considered.

Pneumatic IPS with inflatable boots. Similar results can be obtained for the pneumatic IPS. However, since the system behavior is peculiar of the technical solution applied, trends are different, if compared to the electro-thermal case study. Figure 9.13 shows the power trends, for a time interval similar to that depicted in Fig. 9.11.

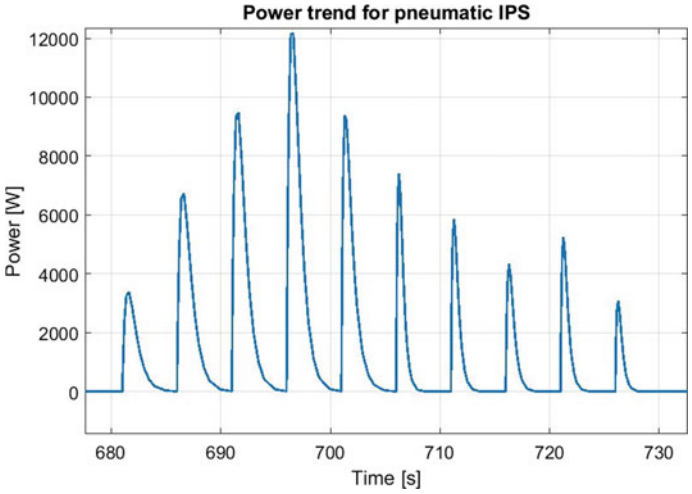


Fig. 9.13 Power consumption for the pneumatic IPS derived from heterogeneous simulation

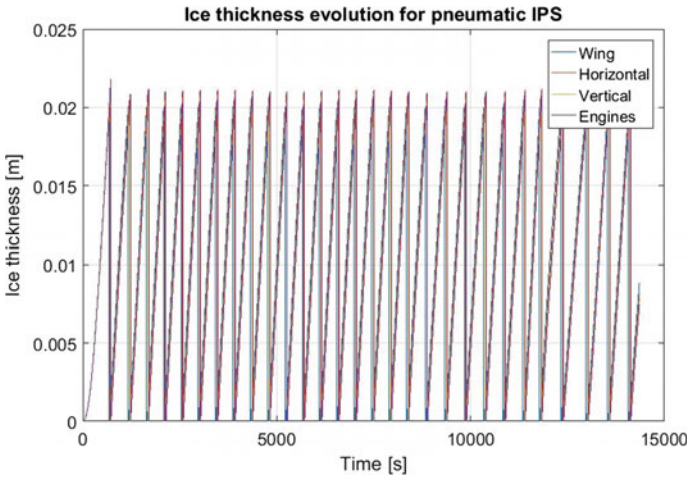


Fig. 9.14 Ice thickness evolution for the pneumatic IPS derived from heterogeneous simulation

In this case, the magnitude of higher peak is similar to the value derived through the classical modeling approach, but the wave form is quite different. Basically, the dynamics is faster and the boots appear inflating/deflating with a higher speed. This remark is a consequence of observing that power peaks are thinner during the activation sequence. This effect is due to the inclusion of the mission profile within the simulation scenario, which brings some variables like atmospheric density and aircraft speed, which affect the behavior of the rubber boots.

The ice evolution during the mission is very different from the electro-thermal case study. As Fig. 9.14 shows, in nominal conditions, the effectiveness of the pneumatic system is very high in terms of ice elimination.

At each actuation, the ice thickness level drops to zero, because of the expulsion of ice layers caused by the inflation of boots. Also in this case, the ice thickness level is compliant with requirements.

9.5 Traceability and Future Evolution of the Interoperability Within Large Toolchains

As far as the heterogeneous simulation is concerned, one may ask how long it is possible to maintain the traceability of different elements involved. Actually, the proposed implementation follows the strategy based on sharing model elements, thus relying on small model elements instead of focusing on the entire formalism. The FMU is generated from the original model, at a certain time, and versioning of the objects and backward synchronization are not possible. However, when the FMU is created again, for example after a modification, the host environment is

capable of reloading the FMU with the new version, without modifying the simulation environment. Moreover, FMU has the possibility of storing data like version number, originating tools, author and date of the modifications, easing the traceability process. The idea is that updated processes shall not be impacted by the use of the standard to represent the simulation, i.e. the user shall not modify his way of working, but a proper structure of the database or of the results folders shall be instantiated to be fully reusable.

Requirements traceability is not subjected to change. Considering the provided example, the simulation environment (i.e. Simulink®) is still capable of accessing the DOORS® database, for linking objects to the SRS. Nothing changes from the point of view of requirements allocation on model elements, although a higher level is considered since, being impossible to navigate the models within the FMU, requirements are allocated on the black boxes representing the model elements and never on components or signals.

The traceability process of both requirements and models can be very difficult in case of complex IT infrastructures. As explained before, for what concerns a distributed simulation scenario, resources may be stored in servers and not be accessible locally. The examples described in this chapter are fairly easy and refer to the local instantiation of this approach. The elements used consist in available resources that are organized through a simple file system structure, whilst requirements are stored in the DOORS® database that, in any case, is still on the client machine where models are built.

When considering a wider toolchain or larger company infrastructures, it is more convenient to rely on *linked data* resources instead of accessing directly the objects. Other kind of standards are available to exploit linked data strategy, to access resources through their links, made available on a secure network within a distributed infrastructure, even for simulation and visualization of results in real time. It is worth citing one successful example of interoperability standard, largely applied within model-based design, known as Open Services for Lifecycle Collaboration (OSLC, 2017). OSLC is an open community building specification for software integration aimed at connecting workflows and environments over the entire product lifecycle. It is divided in several working groups focused on, as example, requirements management, quality management and architecture management as well as lifecycle and linked data patterns. The main objective is exploiting linked data to make resources available among different software for analysis and navigation. The most important and known tasks that can be performed using these standards are the *configuration change management*, performance monitoring and embedded systems design. The main idea is relying on URL to access the results through multiple tools, basing the specification on internet standards, thus re-using already available technology. As the FMI, the OSLC is an open standard, thus being freely available on the web. The community was proposed in 2008 and now it lists more than 60 software platforms supporting this standard, connecting ALM and PLM and successfully linking the different phases of product development. Some of the tools used within the examples are already supporting the OSLC specifications.

This subject is a little bit far from the scope of the handbook, but the Reader can realize the importance of IT solutions to support the MBSE approach, especially considering the connection between ALM and PLM, as well as the interoperability issues related to their effective cooperation. A very brief introduction to the problem was then considered appropriate by the Authors. Nevertheless, it is recommended referring to some more specific literature about the implementation and current exploitation of this technology, to analyze deeper this topic.

References

Blochwitz, T. et al. (2011). *The functional mockup interface for tool independent exchange of simulation models*. 8th International Modelica Conference. Dresden, Germany.

CRYSTAL. (2017, September 28). CRYSTAL Public Aerospace Use Case development Report V3.
http://www.crystal-artemis.eu/fileadmin/user_upload/Deliverables/CRYSTAL_D_208_903_v3.03.pdf

EASA. (2017). *Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes—CS 25.*Accessed 28 Sept. 2017. Cologne: EASA. https://www.easa.europa.eu/system/files/dfu/CS-25%20Amendment%2019.pdf

FMI. (2017, September 28). *Functional mock-up interface*. http://fmi-standard.org/

OSLC. (2017, September 28). Open Services for Lifecycle Collaboration. https://open-services.net/

Chapter 10
System Verification and Validation (V&V)

Abstract A preliminary overview on the connection between Application Lifecycle Management and Product Lifecycle Management is proposed, by introducing the contents and the strategies of the Verification and Validation activities. The chapter should allow the Reader distinguishing the two activities and even the tools and the goals related. A comparison between the typical approaches applied in Software and Hardware Engineering, respectively, is briefly deployed. A preliminary relation with the RAMS analysis, dealing with Reliability, Availability, Maintainability and Safety, is discussed, especially in the two test cases.

10.1 Introduction

The SE deployment leads to a final manufacturing of prototypes and products, and the wide modeling activity developed to perform the product design shall be effective as much as the system fits requirements and brightly fulfills the customer needs. Those are, briefly speaking, the main goals of *verification* and *validation*, respectively, as they were already defined in Chap. 3.

Looking at the whole product lifecycle, a sort of inner loop is established between requirements and system. It is based on a two ways path. The requirements allocation and the system development, production and integration, describe the direct activity performed to build the product, while the verification allows a feedback from the system to the requirements. An outer loop even holds and connects the system to the customer needs. It includes the previous one, but it is extended to a comprehensive check that all the needs are satisfied (Fig. 10.1).

To understand the proposed roadmap it is crucial realizing that, as soon as the product is delivered, the customer satisfaction is manifested by an explicit *acceptance*. To be sure that this agreement could be reached, the system developer should perform a back path to verify the system against requirements and validate it against the customer needs. Practically, it is often said that the product development succeeds if its main goal of providing an *effective system* is reached. It sounds somehow obvious and intuitive, provided that the meaning of system effectiveness is completely stated.

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_10

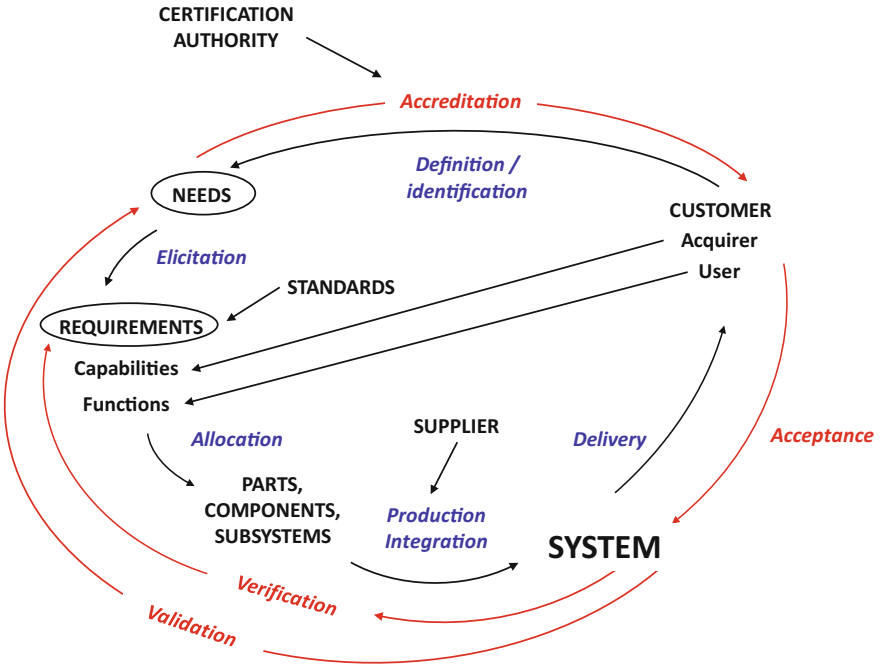


Fig. 10.1 A roadmap of activities related to the V&V

Actually, verifying and validating the system and checking its real effectiveness include a number of items, which sometimes make confusing their description in the literature. After the first action of delivering a final design synthesis of product, seen as a main output of the ALM step of the whole product development, the verification and validation (briefly V&V in the following) should assure a suitable system manufacturing, through a precise quality and configuration control. A clear consistency with the project targets must be even guaranteed. Therefore, the V&V process is strictly connected to the model of development assumed, as in the V-diagram sketched in Fig. 10.2. Basically, the V&V activities belong to the right side of the diagram. Verification operates on the system models, parts, components, subsystems, by resorting to several tools like simulation, tests, measurements, prototypes and proofs, while the system validation is performed on the whole product and may lead to a complete homologation or at least an accreditation of some external certification authority.

Very often the V&V process to be implemented doesn't look so bright at the beginning, especially when the system manufactured is new, or quite different from previous ones. Therefore, demonstrating that the model is representing exactly "what we thought" (*verification*) and the product is exactly "what we need" (*validation*) is never trivial. It might happen that driving a V&V activity seems more difficult than the design itself. This could be the weak ring of an ideal chain between design and manufacturing, or between the ALM and the PDM, within the frame of

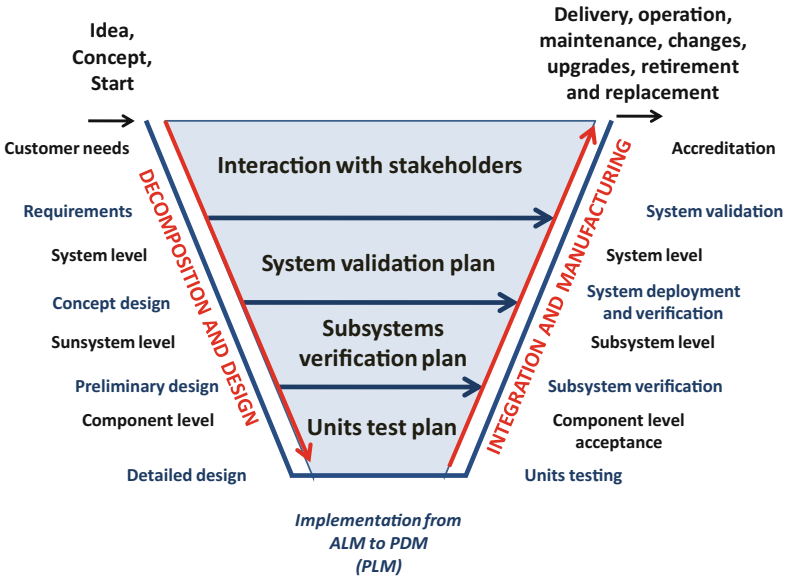


Fig. 10.2 Evidence of the verification and validation role in the product lifecycle management (V-diagram)

the PLM. Moreover, contents of V&V are still a little bit vanishing in the literature of material systems (hardware and constructions), while they were already assessed within the software engineering.

In this chapter an overview of the V&V process is provided, just to give a preliminary impression of the state-of-art as well as of some known roadmaps currently applied. It is clear that a deeper information should be analyzed for a direct implementation, but at least some main features of the V&V process and typical interactions with the MBSE methodology will be preliminarily drawn.

10.2 The Best "V&V" Process

As a methodology based on models the core truth of the MBSE consists in the equivalence between the system behavior predicted by models and the actual performance in service. Unfortunately, modeling is never so simple and when complex systems are observed and tentatively replicated in a virtual reality a number of obstacles might bring to a sudden failure. Basically, every modeler should face a set of *uncertainties*. Characterizing and identifying the uncertainties related to a defined system modeling and evaluating the degree of influence they have upon the final output is a key issue of this activity. Moreover, uncertainties exhibit an intimate correlation with the technical domain explored, and this makes

rather difficult generalizing a suitable process of V&V which could fit all of engineering applications. The best V&V process doesn't exist, although many procedures fit-to-purpose for several applications can be assessed. Nevertheless, some main *target objectives* are common, while *methods* and *metrics* should be adapted case by case to the nature of system.

An increasing attention of designers, manufacturers, customers and stakeholders as well as of some technical standards or technical societies (as the ASME in industrial product design) for a complete assessment of the V&V procedures is now tangible. The development of complex systems suffers the effect of *human errors*, especially when the number of operators is fairly high, and of the number of variables introduced, being known under a limited *confidence level*. The V&V process is aimed at improving the quality of product and reducing the potential errors of humans (Debbabi, Hassaine, Jarraya, Soeanu, & Alawneh, 2010), as far as it should be made as *standard* and *automatic* as possible. It allows assessing the system models, design and requirements. Moreover, a V&V standard procedure avoids a subjective point of view of different operators, and defines common metrics of evaluation. In industrial product development, it concurs to the final assessment by resorting to both qualitative and quantitative techniques, as long as it is based on a real *heterogeneous simulation*, testing and proofs.

The V&V process should even help in exploring the non deterministic nature of several phenomena, which motivate a large part of uncertainties affecting both the system modeling and the product development. It might be remarked that two levels of knowledge are considered in the V&V methods. A specific implementation to the system under development is obviously felt as an urgent task of the whole PLM. Behind that, every industrial domain should promote a complete standardization of those techniques, to be defined as a reference for the related technical sector, without limiting their effect on the single product. Unfortunately, nowadays a comprehensive action to assess the V&V process within the frame of each technical domain is still going on, but is somehow incomplete as it appears in applied science, fluid dynamics, heat transfer, solid mechanics, and structural dynamics (ASME, 2017). Therefore, methods and tools applicable within the context of the MBSE for industrial systems will be here only roughly described, as they are currently known and implemented.

10.3 Verification, Validation and Accreditation (V&V, VV&A)

Concurring with the following official definitions:

- *Verification* is the "confirmation, through the provision of objective evidence, that specified requirements have been fulfilled" (ISO15288, 2015) and "is a set of activities that compares a system or a system element against the required characteristics. This may include, but is not limited to, specified requirements,

design description, and the system itself" (Walden, Roedler, Forsberg, Hamelin, & Shortell, 2015).

- *Validation* is "the confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled" (ISO15288, 2015) and "is a set of activities ensuring and gaining confidence that a system is able to accomplish its intended use, goals, and objectives (i.e. meet stakeholder requirements) in the intended operational environment" (Walden et al., 2015).

A verbal joke is also used to summarize the two concepts. According to it, *verification* answers to question "are we building the system right?" as *validation* does for question "are we building the right system?" (Shamieh, 2012). This explains why a complete validation to check whether customer needs are fulfilled is only possible on the whole system. By converse, when each part, component or subsystem were modeled and built as it was thought, and simulation, tests and proofs confirm that their behavior and performance are those expected, they look fully verified.

To assess the V&V process, some *metrics* to be applied to evaluate the system and its compliance to requirements and fulfillment of needs should be preventively defined and *targets* should be even clearly stated. Some *tools* are required to perform the activity, and they have to assure to be reliable and complete in operation.

It is worth noticing that within the MBSE approach two products are built or assembled, namely the system *models* and physical components. Therefore the concept of V&V is tailored to be adapted to either of those products. This motivates a double definition of *verification*, since it checks that:

- the system *model* precisely describes the concept thought by designer;
- the *product* is correctly built up.

By converse, *validation* is focused on checking the correspondence between the prediction of system performance, provided by models, and its real behavior.

The output of the V&V activity is either an *acceptance* or a *rejection* of the system as is, with respect to a specific usage. It is usually referred to as *accreditation* or *certification*, and is given by a recognized authority, after a preliminary work performed by the manufacturer himself. When manufacturing of material product is concerned, this could be even corresponding to a *homologation* (see for instance the case of motor vehicles).

10.4 Software and Hardware

The V&V process applies to both software and hardware systems. In case of software, many issues of the V&V were already assessed, far less in industrial product lifecycle management. Nevertheless, a brief overview on some typical

issues of the software V&V process should be useful in introducing some current interpretations implemented for the hardware.

First at all, the object to be analyzed in software engineering is *the code*, made by a list of commands expressed into a programming development environment, through a standard language. In industrial engineering a *material product* is usually evaluated, and requires that even the manufacturing process is included.

A formal V&V of software is based on a *static* evaluation, which examines the source code without executing the *model* (dataflow, control flow, syntactical accuracy and consistency), and on a *dynamic* evaluation based on the machine execution of the model, which resorts to some dynamic techniques, as debugging, testing, animation. A fault and failure analysis, together with a traceability assessment, is even performed.

In case of hardware system, a *dynamic* evaluation is usually required to investigate its behavior. *Numerical simulation* is often used to predict the system performance in several operating conditions, and to preliminarily check the fulfillment of some operational and functional requirements. *Testing* on parts, components, subsystems and system is applied to perform the validation. However, a sort of preliminary evaluation, similar to the *static* one applied to software, is performed even in case of a product coming from the material processing. The *feasibility* of manufacturing and *assembly* are tested through some geometric models, or a virtual animation, and on some prototypes, as well as the dynamic behavior is predicted by numerical modeling and then tested directly on some samples of the system. More recently, even the operation of system is checked through a direct interaction between operator and system, by resorting either to numerical simulators, whose animation and graphical interfaces allow a *virtual engineering* activity, or to real *prototypes*. In this activity even structural requirements are checked and assessed.

10.4.1 V&V in Software Engineering

Software engineering focuses on *structure* and *behavior*. Typical objects of analysis are the code modularity, hierarchical structure, inheritance, and encapsulation. They can be related to some more general attributes like complexity, understandability, reusability, maintainability, and nowadays, even the cyber security. When the software is undergoing a V&V process, it could be directly *tested*, i.e. some real or simulated controlled conditions are set up to verify the operability, supportability and performance, and results are compared with the expected ones to detect any errors. The *simulation* is also used to test speed and computing performance and to generate several test vectors. A sort of benchmarking between the implemented model and a reference one is also used to check the equivalence of their behaviors, within the so-called *evaluation* process.

No transition between digital and physical model is required for software, since the product is already a digital artifact. Therefore, the above mentioned activities are directly performed on the system itself. Nevertheless, the SE languages and

related functional models become useful in representing the *structure* of software, through a visual notation, to define attributes and components. Even the *behavior* can be effectively described and checked through some diagrams, for instance expressed into the UML or SysML languages or similar. System interfaces and mutual interactions between elements can be simulated and the complexity degree can be evaluated, through some suitable and standard metrics.

To give an impression about the V&V process applied to the software product, an example is described in Fig. 10.3 as it was conceived by Debbabi et al., (2010) (see previous page). Inputs are simply the system requirements and properties, and some behavioral and structural diagrams, being modeled through the UML or SysML languages. In addition, some applicable metrics and threshold values are defined.

The process sketched in Fig. 10.3 includes two main action lines: one concerns the software structure and is depicted on the right side of the flow, while the behavior is analyzed on the left side. The structure of software is verified by applying some defined metrics to the structural diagrams and results are checked to promote any eventual refinement. The software behavior is investigated through a longer process. Requirements are first expanded into some logical properties, to be easily used in the preliminary analysis and to create a model. The static analysis uses some metrics to check the semantic contents of the model and allows a refinement of it. The dynamic analysis is performed on the execution of the model

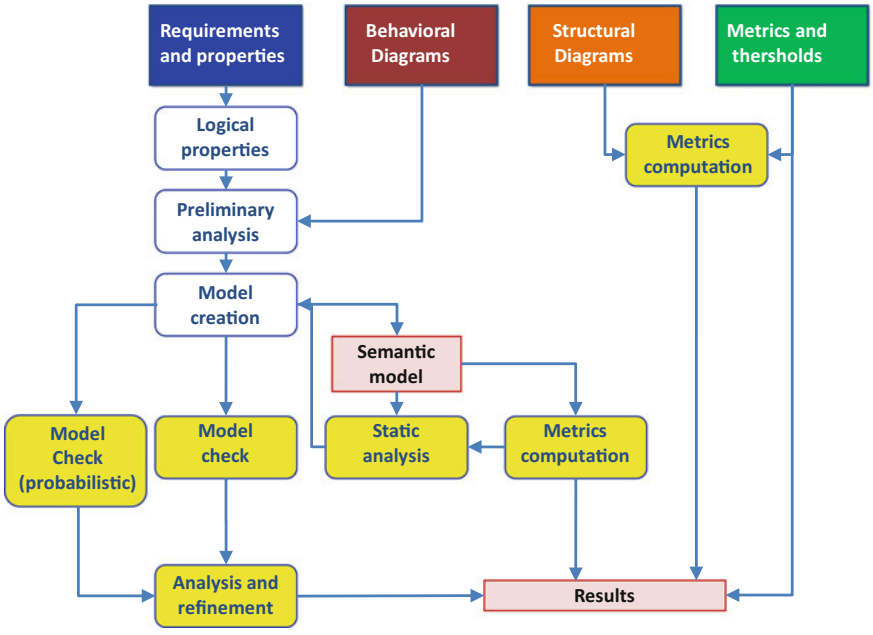


Fig. 10.3 Verification and validation process applied to the software product

and even on a probabilistic check. Results are analyzed and collected, after an eventual refinement. The example above described was expressively proposed by the authors to integrate the MBSE approach with that process.

Technicalities of the above mentioned process are out of the goal of this handbook, but some typical issues of this approach may be appreciated:

- Several *states* of the system are considered by the V&V process, being related to the static and dynamic behaviors.
- A *suitable path* of activities is defined and implemented.
- *Qualitative evaluations* are applied to check the structure of the analyzed system.
- *Quantitative evaluations* are needed to measure the performance.
- *Metrics, thresholds, results* and *logical operations* are used to perform the V&V process.

A clear understatement about the meaning and the reference parameters of the *product quality* has to be carefully assessed. In case of software product, the quality is related to some attributes to be compared later to those of the hardware product. They are even considered as an objective of the design activity.

The most important objectives considered in the literature are:

- *Cohesion*: it measures the strength of functional links between system components. A high cohesion is preferred in high quality system design.
- *Complexity*: indicates the degree of intricacy exhibited by the system.
- *Coupling*: describes how deep the system parts are interdependent. Sometimes a lower coupling allows a higher quality of design. For sure it affects the complexity, maintainability and reusability of system.
- *Maintainability*: it indicates the degree of difficulty and, eventually, the time required to change the system design or its implementation to be adapted, corrected, or to prevent some undesired effects.
- *Reusability*: it defines how easy and fast the system design or implementation can be reused.
- *Stability*: in case of undesired changes of the software, this is the measurement of the risk of getting some unforeseen effect.
- *Testability*: is the availability of a given application to be tested and, somehow, the capability of tests to interact with the code to detect critical imperfections.
- *Understandability*: it measures how deep the stakeholders can understand the system specifications.

Those properties could be even considered in case of hardware, in connection with some other ones. Priority might be different. Complexity, maintainability, reusability, stability, testability are surely required in hardware design. Cohesion and coupling are usually investigated as an assembly capability and integration, although, in smart systems, coupling is specifically referred to the energy conversion mechanism. Understandability turns out into the degree of simplicity of the proposed layout.

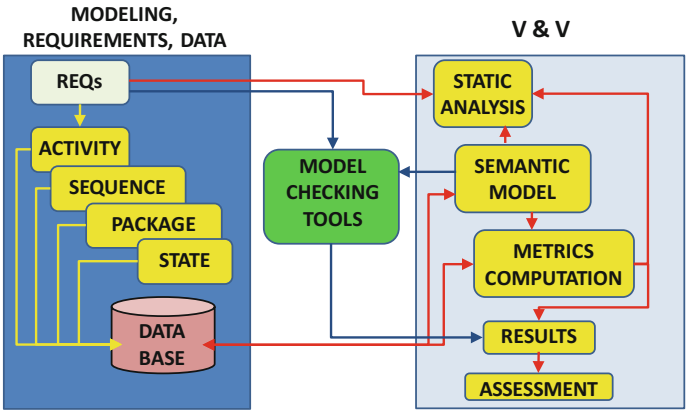


Fig. 10.4 Example of tool chain for the verification and validation of the software product

An interesting issue is the *tool chain* exploited in software engineering. It might look like the sketch described in Fig. 10.4. A first environment should connect the data repository, the requirements and the functional models of the system. A dedicated environment may be set up to perform the V&V. It includes metrics, thresholds, and checking procedures, eventually by sharing data and interoperating with other specific environments aimed at performing some detailed analyses and model checking.

Those approaches demonstrate how the SE and its tools could be used to define a roadmap for the V&V process, and provides some objects to implement this activity, as diagrams and models, which concur with simulation and tests to perform a complete review of the system features. Moreover, the capabilities defined by some architectural frameworks may be used as a reference target, to check whether the product modeled and manufactured fits the requirements and the customer needs.

10.4.2 V&V in Hardware Engineering

The V&V process of material systems was widely discussed in the literature because of some intrinsic peculiarities of product, which are related to its physical nature, in contraposition to the digital artifacts of software engineering. A strong interaction between design, manufacturing, integration and assembling is usually experimented. This makes sometimes rather difficult identifying a straight path to complete the two actions of verification and validation.

Main goals. Definitions of V&V agree with those previously recalled, since in material system development the verification process must confirm that the product design synthesis provides a physical architecture which satisfies the system requirements, as the validation process must demonstrate that the manufactured

system satisfies the customer and fulfills his needs. All relevant issues are considered, mainly safety, cost, quality and performance. Moreover, a full requirements allocation is a first target of the verification activity as well as the overall traceability check.

Verification tasks. According to the literature four main activities are foreseen to proceed with the system verification (Defense, 2001) i.e. the *analysis*, *inspection*, *demonstration* and *test*. Calculated data and experimental test outputs obtained on subsystems and components are exploited by analytical models and techniques to predict the system performance, together with numerical modeling and simulation, to perform the *analysis*. It is often associated to some visual examination of parts, components and subsystems to check some design features in *inspections* made by operators. A full *demonstration* of requirements fulfillment is often done on subsystems, components and parts, for a preliminary identification of the overall performance, but *tests* are finally used to get detailed data and to complete a better verification of system design. Demonstration and tests should check that some *Key Performance Indicators* fit requirements. They should be verifiable and measurable being crucial in system behavior.

Those actions are organized with a suitable verification process. An adapted version of the IEEE 1220 standard was proposed by the Department of Defense of the USA, to be applied to the hardware as herein schematically resumed.

Once that the *design synthesis* is ready, following steps are performed:

1. *Selection of the verification approach* (*input*: *Design synthesis*)
   a. Definition of analysis, inspection, demonstration and tests
   b. Definition of procedures for given V&V environment.
2. *Conduction of the verification evaluations* (*inputs: requirements, functions and architecture*)
   a. Verification of the architectural completeness
   b. Verification of functional behavior and performance
   c. Verification of constraints satisfaction.
3. *Identification of variance and conflicts* (*inputs*: *outputs of previous steps*)
   a. Feedback to requirements and design synthesis
4. *Assessment of a verified physical architecture (final synthesis)*
   a. Definition of the configuration control
   b. Specifications and configuration baselines
   c. Assessment of the product breakdown structure (PBS).

Validation tasks. The verification process is then followed and somehow integrated with the validation activity. A clear example is provided by the ASME V&V 10.1 Standard, being illustrated in Chap. 3 (Figs. 3.3–3.9). Basically three action lines are foreseen. The system is first an object of abstraction, in a preliminary *concept design*, which is widely supported by the MBSE tools and languages. Path

bifurcates then and a *numerical modeling activity* is performed in parallel with a *physical modeling*.

The numerical one includes the composition of a *mathematical model*, being then transformed into a *numerical model*, ready to be *simulated*, in order to compute and store the results. The *physical modeling* activity starts with the construction of a *mock-up*, *experiments* are then set-up, and *tests* are performed. Measured results are collected and stored.

The *verification process* described in previous section applies to some issues of this workflow. It checks first the consistency of code used to numerically model the system. A cross check is then performed between preliminary computations and experimental tests executed on parts and components. The numerical calculation is verified by comparing the results of different analyses (analytical and numerical for instance). The system performance and the compliance with requirements and specifications are finally verified by simulating the system and running some preliminary tests. The *validation* consists in a precise correspondence between numerical results and experimental evidences obtained on the system prototype by simulating some operational conditions.

10.5 A Methodological Approach to the Industrial Product V&V

The physical nature of the hardware product makes longer and somehow more expensive in terms of time, resources and cost, the V&V, but actually it makes it clearer in its contents, being a material system to be integrated and operated. Nevertheless, some words like *compliance*, *performance*, *completeness* and *satisfaction* appear less intuitive and clear, as soon a practical activity on a real system is faced. Therefore, some additional details are herein proposed to clarify better, from an operational point of view, few and basic concepts just behind those keywords.

10.5.1 Practical Issues in Product Development and Relation with V&V

To give a comprehensive answer to the open question about the practical meaning of the keywords above mentioned, details of the system analysis, design and development are needed, as for instance Wasson (2006) describes in his textbook. To convert the main contents of the V&V methodology into some practical and appreciable tasks, it is suggested to interpret the engineering of systems as a set of three actions, namely *verification*, *validation* and *documentation*. To convert into a practical approach to V&V, those actions can be roughly explained as follows, although they were precisely defined as it was above mentioned by the Standards.

Verification, at the end of words, should investigate whether the product development effectively assures the *system integrity*. It means that requirements, being expression of needs, must be manifested in the system product and service (design, manufacturing, operation), completely and even always in the same way, without any kind of mismatch between different steps of development. The consistency of the flow described in Fig. 10.1 is somehow checked, as in a latin text the so-called '*consecutio temporum*' of verbs.

Validation should investigate the *acceptability* of the system, i.e. its operational *utility*, *suitability* and *effectiveness*. Those are the criteria of *customer satisfaction*, together with a nice perception of product as an expression of innovation and user friendliness, somehow coupled to some appreciable aesthetics and a compatible cost. This explains why a clear *business model* should be defined about the product, to be able then of validating the system.

To complete the system development and to reach that satisfaction, the *documentation* delivered should exhibit a high quality, in terms of *understandability, completeness and correctness*, as well as the process, which should allow a reliable and predictable replication of the product in each copy of the system, within the performance constraints. This task is greatly helped by a model based approach, since the generation of documentation naturally descends from the modeling items of the whole system.

Those interpretations nicely fit the deployment of the system development. Practically speaking, after that needs are defined and specifications are written, designer should interrogate himself to know whether the right system was specified. As soon as from specifications the analysis, design and development are performed, question should become whether the system was built right, and then, when the system is deployed, the last question should be whether the right product was built up. In this sense an agreement with contents of previous sections is realized, but three main remarks could be introduced.

As it is easily understandable a tight *cooperation* among acquirer, user, system developer and service provider is strictly required. This motivates the pervasive activity currently done through the SE to connect all of those actors together and continuously, and to take care of the after marketing, through, for instance, the remote service and monitoring provided by the *Internet of Things (IoT)* and related technologies. All those actors were even located in the sketch of Fig. 10.1. It is known that the *customer* defines the needs, which are exploited as an input for the elicitation of requirements. Two points of view are considered. As an *acquirer*, the customer is interested in some capabilities of the product, eventually screened by means of the architecture frameworks. The *user* is prone to define and look for some functions. Development reaches the step of integration with a suitable support of all the *suppliers*, while the accreditation is made by a *certification authority*.

The V&V is an *iterative* action, which allows a continuous refinement of products, particularly when a product line is considered and the model based approach is applied to its development. Those details intrinsically define the *product baseline* as a series of activities which might be associated to different states of the product along its development. This brings to define the system in different

manners, step by step "as is designed", "as is verified", "as is built". Moreover it clearly states the existence of a configuration development, which includes some changes, to be motivated, authorized, tested and managed, through a suitable control, as it is briefly described in the next chapter.

10.5.2 Workflow of V&V

Some basic concepts about the V&V were above expressed, but some doubt may arise about the time at which actions should be performed and how, without a workflow. It is tentatively described in Fig. 10.5, by resorting to the literature and some current implementations. The sketch represents a generic documental approach, but a conversion into a model based solution is easily made, through the MBSE. It links actors, steps and outputs of each development phase, and somehow defines the strategy of V&V applied.

A first milestone is the *System Performance Specification* (SPS). This document follows the *customer needs* identification and a preliminary elicitation of requirements, related to the *operation and mission* of the *system*. It is a matter of assessment and negotiation with the *acquirer*, or directly with customer, when they are undistinguished. This action leads to a commitment, being signed in form of *contract*, for a direct procurement. The redaction of this document is expressively helped by the architecture frameworks and related views and system capabilities.

As soon as the contract is ready, the core activity of *system engineering and development* starts. For this task the methods, process, tools and models of the SE are applied. Particularly, V&V activities are included. If the decisional activities are concerned, some supports can be identified in analyses, trade-off's, functional and numerical models, simulations and prototypes. A tight collaboration with the *user*, or directly with the customer, when undistinguished is required.

The *system verification* is based on a *Developmental Test and Evaluation* activity (DT&E), while *validation* shall be based on the system *Operational Test and Evaluation* (OT&E). Basically, analyses, inspections, demonstrations and tests above mentioned are performed by resorting to some virtual and physical models of subsystems, components and parts to lead the system integration.

All typical issues of product development are verified in an ordered sequence of actions, which may be associated to some dedicated documents. In these activities the system developer checks the *progress* of development, the *maturity* of system and of selected technological solutions, the *integrity* in terms of traceability and allocation of requirements, and the *risk of failure* of the decomposed system, at least preliminarily. Some *main technical reviews* (MTR) and traceability audits (TR) are therefore performed in this step. A milestone is met, when the *critical design review* (CDR) is made. It allows setting up the testing, as some *test readiness reviews* (TRR) check. A first campaign of *system verification tests* (SVT) is performed and results are acquired, even through two dedicated *audits*, one aimed at investigating the system functionality (FCA), for a given configuration, and another one by

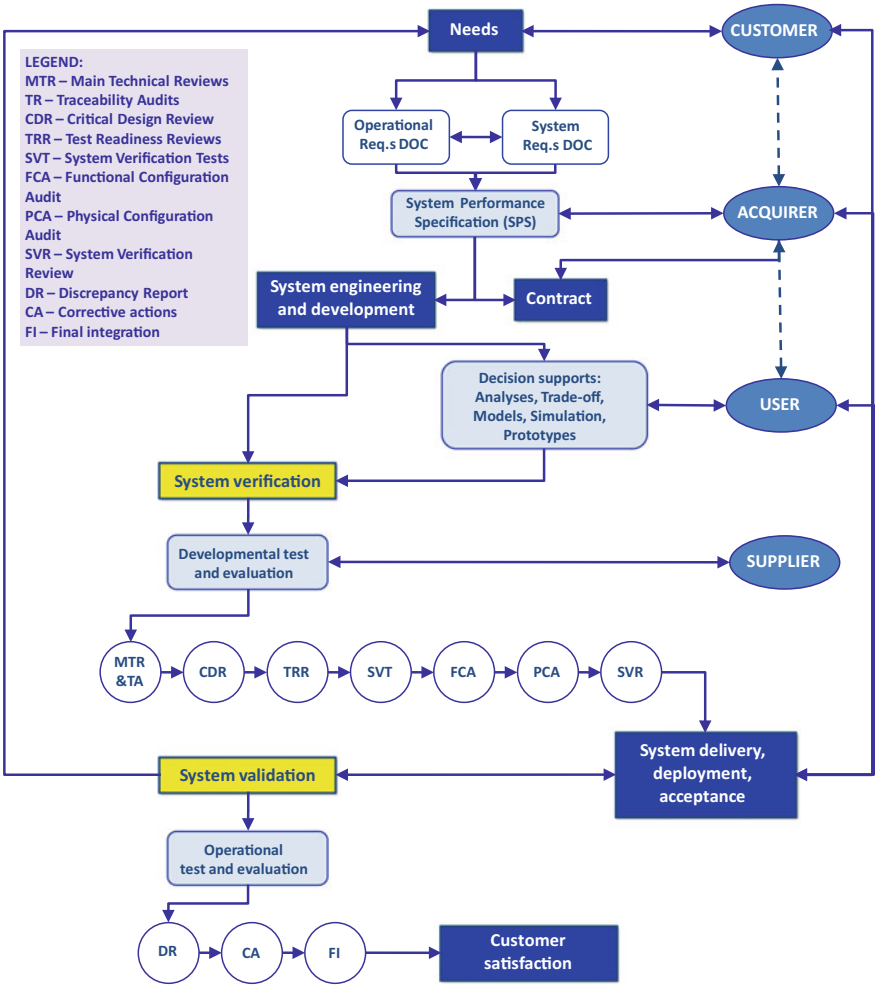


Fig. 10.5 Example of V&V workflow applied to the software development

checking the architecture associated to the physical layout (PCA). As a second milestone the *system verification review* (SVR) provides a final synthesis. This assessment is performed with a contribution of *suppliers*, when required. It allows starting the final production to deliver the system, to be deployed in service and finally accepted by the customer.

The acceptance concerns the validation. It is promoted by the system developer, performed under the control of a recognized authority and provided to the customer. Tests and evaluation apply to the whole system and its *operation*. Any eventual *discrepancy* with specifications are highlighted in a *report* (DR), followed by a number of *corrective actions* (CA) for a definitive *integration*, before delivery (FI), compatible with *customer satisfaction*.

10.5.3 Design Objectives

As in the software, several design objectives were defined to identify some suitable criteria to check the quality of product, and even more can be found in case of a physical product. Objectives help in identifying some views for the design activity and a number of criteria for the test and evaluation performed during development (verification) and in operation (validation). Several standards and books include a complete list of those objectives, and cite their standard definitions. To make easier reading this section they will be briefly explained and recalled.

As soon as these objectives are analyzed, it might be realized that, in many cases, they are related to a system *ability* or *capability*. A classification allows distinguishing the objectives, even in relation with the product design, production, and service.

A brief list of the most cited of objectives is proposed.

Intrinsic abilities.

- *Simplicity*: can be interpreted as the system property of exhibiting a design, a layout or an implementation, which can be easily understood or deployed.
- *Safety*: is the ability of system to work in defined operating conditions by avoiding failure, damage or collapse, which might affect the health and the life of people, the integrity of other systems and its own survivability. Operating conditions have to consider all constraints of operational effectiveness, time, and cost throughout the system life cycle.
- *Security*: it measures the level of system protection against any intrusion and unauthorized access (and somehow of being tampered). About this, it is true that this ability is more related to the software product, but current development of mechatronic and autonomous systems makes this item strategic for the implementation in hardware systems too.

Production abilities.

- *Producibility*: it defines how easy is manufacturing the system, in terms of assembly, inspection, testing and cost.
- *Testability*: it expresses how long a system requirement allows defining some test criteria to be applied in testing, to measure the system performance.

Installation abilities.

- *Portability*: is associated the degree of difficulty found in moving the system from one site to another one, under defined conditions.
- *Transportability*: in this case the capability of system to be carried out by some transportation system is considered.

Operation abilities.

- *Efficiency*: is the "degree to which a system performs its designated functions with minimum consumption of resources" (IEEE 610.12-1990)
- *Re-configurability*: concerns the system ability to be re-configurated by either a manual or automatic action to better support the mission.

Service abilities.

- *Availability*: it defines the level at which the system can be operated when a mission begins, at any unknown time; somehow it describes the system readiness and is associated to the reliability and maintainability.
- *Maintainability*: it concerns the ability to be retained or restored to a specified condition by a maintenance action.
- *Reliability*: is the ability to perform a mission, with a specific duration and operating conditions, without reaching conditions for failure or degradation.
- *Survivability*: is the ability of withstanding some hostile operating conditions and to accomplish the mission.
- *Usability*: it describes how easily a user can learn to operate, prepare inputs, and interpret outputs of the system (IEEE Std. 610.12–1990).

Design objectives are useful to define requirements, to describe the quality of product, to define the metrics for the V&V evaluations. They immediately help in understanding the real contents of the *system effectiveness*, being assumed by the INCOSE handbook as a function of *suitability*, *dependability*, *reliability*, *availability*, *maintainability* and *capability*, and as a measure of how the system satisfies the customer needs and fits requirements (Walden et al., 2015).

Those are just some of the design objectives which can be selected in the industrial product development. Among those, some remarks are due for safety and service abilities.

10.5.4 Smartness and Smart-Nect-Ness

More recently a number of design objectives were added to cope with the need of updating the references to include the attributes of mechatronic systems, being characterized by a deep interaction and coupling among electronics, mechanics, automatic control and computer science. This field introduced a new interpretation of the concept of *smartness*, interpreted as a global capability of systems to be sensitive to the environmental changes, to elaborate a reaction and to deploy it completely. The fast growing up of cyber-physical systems for the smart manufacturing (related to the "Industry 4.0" vision) requires to take care of some capabilities of intercommunication between systems (Lee, 2008). This motivates an enrichment of the smartness concept, toward a smart behavior associated to a smart connectivity, or "*smart-nect-ness*". In case of mechatronic smartness some abilities were already defined, if one summarizes some main contributions of the literature,

as is herein done. By converse, a complete list of abilities to be specifically associated to the smartnectness is not yet available, although it is the focus of many research activities.

Smart functions in a new system might be related to the abilities of *selectivity*, *self-diagnosis*, *self-tuning*, *sensitivity*, *shape-ability*, *self-recovery*, *simplicity* (concerning the energy conversion), *self-repair*, *stability* (materials, control, dynamics), *standby*, *survivability*, and *switch-ability* already defined in Chap. 4, when the elicitation of requirements was analyzed.

As it looks evident, some objectives change their intrinsic meaning when technology evolves from mechanics, for instance, to mechatronics and to cyber-physics. The sensitivity, for example, describes how fast and effectively the system or the smart material reacts to an external stimulus, by resorting to a relation of cause and effect, which might be either linear or nonlinear with the stimulus amplitude. The self-diagnosis in piezoelectric systems expressively describes the ability to identify a damage of material directly through the piezoelectric phenomenon exploited for sensing and actuation. In new systems, remotely connected, it could measure the degree of sensorized functions and the failure detection. The simplicity relates no longer only to the intricacy of layout, but even to that of the coupling mechanism deployed to make the system smart. The shape-ability is the re-configurability above mentioned, but even more evident, being related to the ability of changing the shape of material and system, not only the global layout.

10.5.5 DT&E and OT&E for the Industrial Product

The Development Test and Evaluation (DT&E) and Operational Test and Evaluation (OT&E) practically enable the V&V process to detect any critical issue in the system configuration and in its operation. To accomplish that goal, some quantitative references are required.

To perform a validation, the system developer must define some *Key Performance Indicators* (KPIs) to start the investigation, by selecting capabilities, properties, and measurable variables so strategic that reaching a threshold for these imposes a re-evaluation, a stop or a redetermination of the system.

In addition, some *Technical Performance Measurements* (TPM) are used to detect in system operation when a risk of failure might occur, according to a selection of probable risks, provided by a dedicated analysis, or when some safety critical specification is overcome.

The practical T&E activity is based on the following steps:

- the system developer selects a *performance parameter*;
- for a defined time of operation a *planned value* should be fixed, together with a *profile* for the measurement time;

- a *tolerance band* of values is defined;
- a *threshold* establishes when the value is out of range;
- the *variation* is even measured, in terms of difference between the planned value and the *achievement to date* caught in measurements.

Some estimation could help the validation activities, as well as establishing a technical milestone to be achieved within a time period.

To perform the validation, an *analytical or numerical prediction* of the TMP is plotted as a function of the system operation, either expressed in time or by phase (Fig. 10.6). It is compared to the *current trend* of the TPM measured in operation, during some tailored tests. Moreover, if a *tolerance band* is known, the plot is delimited by upper and lower bounds or *thresholds* to define a *risk area*, eventually by distinguishing different levels, with failures of increasing severity, and a fully *unsafe operation*. Critical events are detected by an *alarm* first and by the occurrence of a *danger*, above and below the thresholds.

The *Technical Performance Measurements* (TPM) are linked to some metrics, usually defined by standards, as the *Measures of Effectiveness* (MOEs), the *Measures of Suitability* (MOSs), and the *Measures of Performance* (MOPs). They are indicators defined by customer, user and acquirer together with the system developer and somehow contained in the operational and system requirements.

Performance usually leads to a quantitative evaluation, which helps both its definition and measurement. It is part of the effectiveness, and is usually related to some physical parameter, like spin speed, payload, acceleration, and torque. An example is herein proposed, analyzing the didactic test case.

Effectiveness describes "how well the system performs a task, meets an operational objective or requirement, under specified conditions" [DoD 5000.59-M Modeling and Simulation (M&S) Glossary]. It measures the degree of mission accomplishment of a system.

Suitability measures an objective performance, in the base of some subjective user criterion. It somehow describes how satisfactorily a system is used, if, for instance, the availability, compatibility, transportability, interoperability, and logistic supportability are concerned.

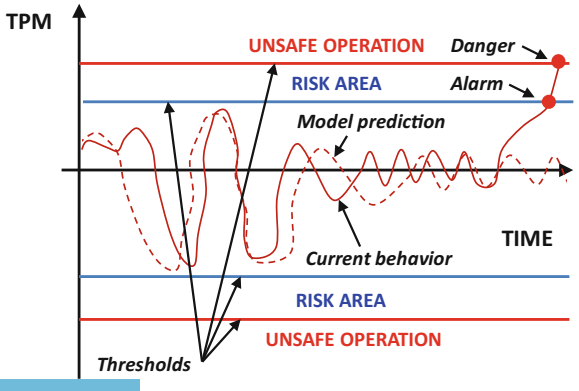


Fig. 10.6 Example of validation performed on a Technical Performance Measurement (TPM)

10.6 The Role of RAMS in V&V

Safety and *security* are so important that a specific branch of engineering, namely the *safety engineering*, is found in education, literature and even in industrial organization. Safety needs to be defined against a set of failure modes, and, for each one, the uncertainties related to the inputs of design (material properties, loading and operating conditions, real constraints and interfaces), the approach followed in the calculation of design parameters (method, assumptions, procedures, computational errors and approximations), and other unknown effects, is covered by introducing a suitable margin, referred to as "safety factor". It is related to a specific design parameter, and usually is defined as a ratio between the numerical value required to reach the failure condition and the one describing the operation level. In structural design, for instance, the yielding stress leading to access to the plastic behavior of material is compared to the maximum stress occurring within the system and their ratio is computed to assure that is suitably higher than one.

It is worth noticing that SE and the methodologies described in this handbook were mainly conceived to prevent the risk of unsafe system operation and management. In many applications, often described as safety critical, the failure modes associated to the risk of unsafe service are the main objectives of requirements and needs. Safety is strictly related to some other abilities as the Reliability, Availability, Maintainability, which all compose the so-called set of RAMS, being a specific target of analysis.

10.6.1 Reliability

The prediction of failures and of their effects is a key target of the reliability analysis. The *failure* is known as "event or inoperable state, in which any item or part of an item does not, or would not, perform as previously specified" (MIL-HDBK-470A). The severity of failure affects the system operation. A *critical* level inhibits to achieve the mission goal, may lead to a permanent *damage* or even to a complete *collapse* of the system or of some component. A *degradation of performance* decreases the quality of service, increases the risk of collapse, but allows continuing the mission, within a certain margin of acceptability.

For a given set of mission profile, use cases, and scenarios, the reliability analysis is performed by calculating some typical values. It is known that systems, after an initial period of time in which the failure rate could be fairly high (λ), are characterized by a decreasing of failure rate, which remains constant for a certain time, quite long with respect to the life, before growing up as soon as aging effects are suffered. This behavior is typical of many industrial products and corresponds to the paradigm of bathtub curve.

Knowing the failure rate, λ, of each component is required to design the whole system. Nevertheless, a probabilistic approach is applied to predict the system reliability, R(t), as a function of time, t, and of the probability of failure in time, P_{fail} as:

$$R(t) = 1 - P_{fail}(t) \tag{8.1}$$

Particularly, if the *failure density in time* is f(t) and is linked to failure rate through an exponential law of time, the system *reliability* is related to the failure probability, F(t), within a given time period $t_0 - t_1$, as follows:

$$R(t) = 1 - F(t) = 1 - \int_{t0}^{t1} f(t)dt = 1 - \int_{t0}^{t1} \lambda e^{-\lambda t} dt \tag{8.2}$$

Reliability allows evaluating the *hazard rate* as "the conditional probability density that the system will continue to perform its mission within specification limits, without failure through the next time increment ($t_1 + \Delta t$)" (Wasson, 2006) as:

$$h(t) = \frac{f(t)}{R(t)} \tag{8.3}$$

in case of exponential law of failure and negative exponent.

The mean failure rate, μ, is calculated to investigate the frequency of system failure. It corresponds to the mean value of the cumulative failure probability density function and is expressed as the reciprocal of the *Mean Time To Failure* (MTTF):

$$\mu = \frac{1}{MTTF} = MTBF - MTTR \tag{8.4}$$

being *MTBF* the *Mean Time Between Failures* and *MTTR* the *Mean Time To Repair*. Those numerical values define the operation condition of the system with respect of the occurrence of failures and are objects of the analysis. Moreover, the overall reliability of a system is calculated by analyzing its architecture and inputting the reliabilities of components, organized into a network of series and parallel connections, by knowing that the equivalent reliability of a series of two components is the product of reliabilities, while the equivalent reliability of a parallel of two components is the difference between the sum of reliabilities and their product.

10.6.2 Maintainability

The maintenance of systems can be *preventive and periodic*, if it is scheduled to avoid that failure occurs or that performance degradation is so severe that might

have an impact on the system behavior. Prevention can be done through a *Condition-Based Maintenance* (CBM), which resorts to a monitoring activity to detect a potential failure. A corrective maintenance is unscheduled, is applied after the failure occurrence and usually requires to remove and replace some components.

To evaluate the maintainability of system, some parameters are calculated. The *Maintenance Down Time* (MDT) describes how long the system will be out of service and is the sum of *Mean Active Maintenance Time* (M*), *Logistic Delay Time* (LDT) and *Administrative Delay Time* (ADT). The *Mean Corrective Maintenance Time* corresponds to the MTTR. Therefore the failure rate can be interpreted even as a *Corrective Maintenance Frequency*.

The *Mean Time Between Maintenance* (MTBM) reveals the maintainability of the system and is related to the frequency of preventive maintenance f_{pt}:

$$MTBM = \frac{1}{f_{pt} + 1/\lambda} \tag{8.5}$$

10.6.3 Availability

The above definitions allow realizing the meaning of availability of the system. Particularly, some reference times are computed to investigate this objective. The system *Operational Availability* is A_0:

$$A_0 = \frac{MTBM}{MTBM + MDT} \tag{8.6}$$

and compares the time between two maintenance actions (MTBM) and the sum of this and the down time (MDT). The real *Achieved Availability,* A_a, is found as:

$$A_a = \frac{MTBM}{MTBM + M^*} \tag{8.7}$$

where the *Mean Active Maintenance Time* is introduced (M*), while the Inherent Availability, A_i, is:

$$A_i = \frac{MTBM}{MTBM + MTTR} \tag{8.8}$$

10.6.4 FMEA and FTA

The RAMS analysis is performed in the industrial product development as a crucial step of the design and V&V activities. In safety critical systems, the benefit introduced by the MBSE is greatly appreciated. According to a fairly traditional

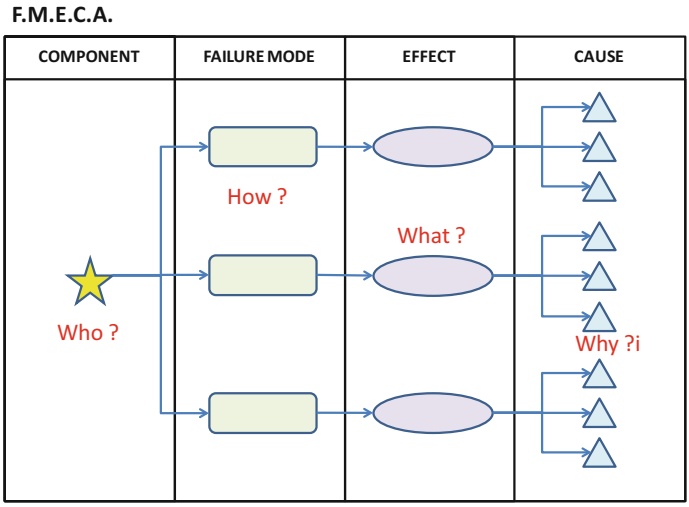


Fig. 10.7 Example of FMEA

approach the RAMS analysis is started in an advanced step of system development. Typical tools are the *Failure Mode and Effect (Critical) Analysis* or FME(C)A and the *Failure Tree Analysis* or *FTA*. The first one investigates through a qualitative analysis, for each element, the failure modes known. Each one is related to its effects and causes, as in Fig. 10.7. A comprehensive overview about failures due to a misuse of the system is provided, but a clear identification of the failure propagation and of system reliability is poorly achievable. The *FTA* is more suitable for this activity. It analyzes the network of components interconnected and allows identifying the probability of occurrence of a certain event, if it is known at local level, for each component and device (Fig. 10.8). A failure path can be even drawn.

The main drawbacks of those tools are that they were not conceived to be connected and integrated with other models, like the functional and numerical ones. They are applied fairly late in the development and provide basically a qualitative output, with some numerical details. This means that they are poorly useful in defining the related requirements, without iterating the whole process.

The MBSE approach helps in developing the FMEA and FTA, once that BBD, IBD and Package diagrams are derived and Behavioral Diagrams are drawn. States, activities and sequences suggest several elements to fill the FMEA tables and to analyze the FTA. In fact, this is just a small contribution.

10.6.5 Dysfunctional Analysis

It is possible to anticipate several analyses concerning the RAMS in an early stage of the system project. This is a consequence of a preliminary analogy which may be

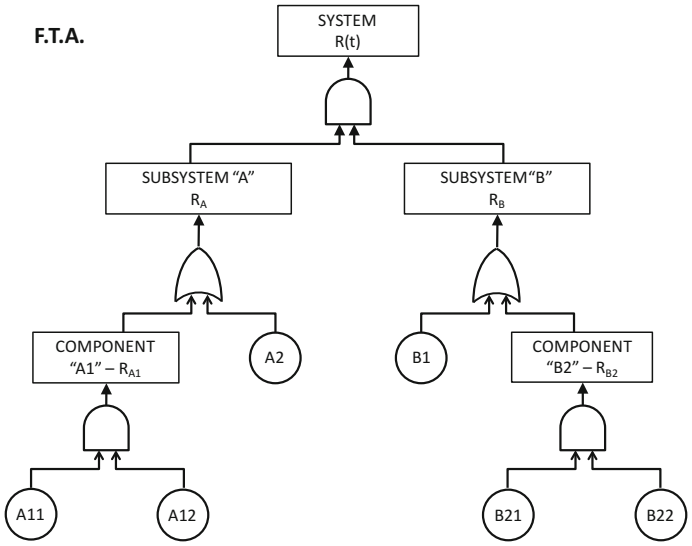


Fig. 10.8 Example of FTA

seen between the product development and the safety analysis. To introduce this analogy a preliminary description of the rationale applied to the Model Based assurance of safety is proposed.

As in the system development the functional analysis is a starting point for the activities of drawing the Functional (FBS), Logical (LBS) and Physical Breakdown Structures (PBS), a non-functional or dysfunctional analysis is performed by Safety Engineers to investigate the failure conditions as dysfunctions. This belongs to the Functional Hazard Analysis (FHA). A second step allocates on the system elements (parts, components, subsystems) some reliability requirements as a target to be achieved. Finally, the system reliability is predicted, once that the overall architecture is defined.

As it may be easily perceived, there is a perfect analogy between the two activities if they are compared each other. The FHA can be promoted early in the system definition, as the functional modeling starts. This helps in performing a screening of critical functions, in terms of safety or RAMS' issues, since the concept design, although the neighborhoods of the system are quite roughly drawn (as in the first picture of Fig. 10.9). As soon as the logical layout is growing up, an immediate correlation with some target reliability can be stated and the system looks more intelligible. Finally, when the Product Breakdown Structure is assessed, several requirements in terms of safety and reliability are already known and the selection of commercial products to compose the system assembly looks already driven. This implies a better system integration, being the RAMS issues defined a priori more than checked only at the end of process. Once that those actions are

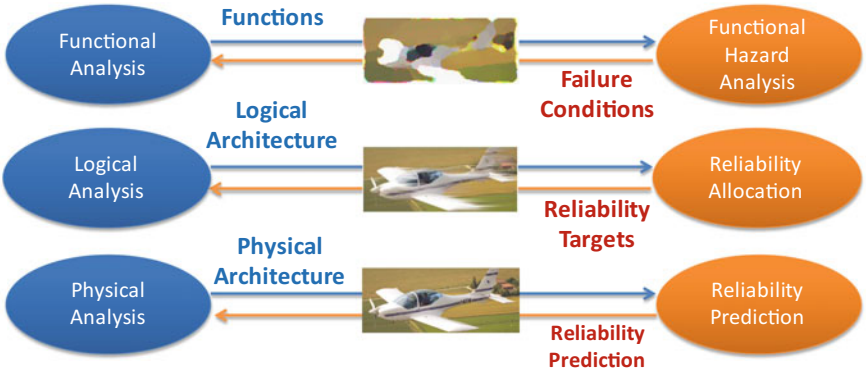


Fig. 10.9 Impression of the analogy between functional and dysfunctional modeling in MBSE

completed a final prediction of the overall reliability of the system is performed. All the information related to that activity concur then to draw the FMEA and FTA and other outputs, according to the Standards applied.

10.6.6 Integration of RAMS and Numerical Simulation

Benefits of the MBSE can be easily appreciated in case of safety and reliability engineering applied to complex systems. Instead of performing a screening of reliability issues, through some tools like the FMEA and FTA, just after having assessed the final layout of the system, it is possible to include the failure modes and even the probability of their occurrence in the numerical simulation of system models, to test the overall system performance. The system nominal behavior is first analyzed and it corresponds to the case in which no failure ideally occurs. Some failures can be then activated, by setting some specific variables, being suitably introduced within the system simulator, to check the transient response of the system in presence of dysfunctions.

This approach allows investigating the role, influence and effectiveness of different failure modes before prototyping the product. As a result, the system integration may be performed with a deeper knowledge of the intrinsic role of each component and it allows defining several requirements for the whole product development. Transition between the *Logical Product Breakdown* and the *Product Breakdown Structure* should be better driven and somehow more effective.

This approach is currently under development, but a process was already proposed and assessed for several applications (Garro & Tundis, 2015), as it was formalized for instance by the so-called *RAMSAS method* (Tundis, Ferretto, Garro, Brusa, & Mühlhäuser, 2017). The workflow is sketched in Fig. 10.10. Four actions

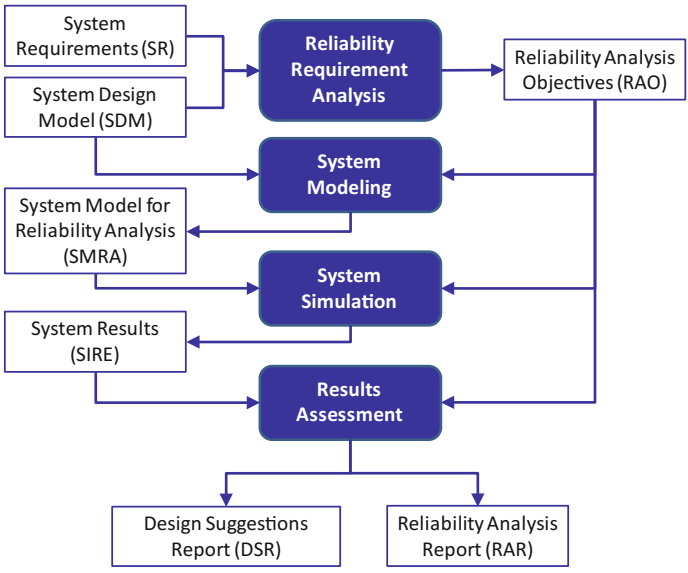


Fig. 10.10 Workflow proposed by the RAMSAS methods to integrate numerical simulation and reliability analysis

are performed in sequence. A preliminary activity concerns the *Reliability Requirement Analysis* and needs as an input the system model and requirements, respectively. It produces a list of *Reliability Objectives*. A real *System Modeling* is then started. It resorts to the system model and includes the reliability objectives as a matter of investigation to create an extended *System Model*, suitable to perform the *Reliability Analysis*. It is used to run a *Simulation*, which includes even the Reliability Objectives and gives some *Simulation Results*. Finally, Simulation Results and Reliability Objectives are *assessed*, to provide some *Design Suggestions* and a *Reliability Analysis*, through some dedicated Reports.

10.7 V&V Peculiarities of the Proposed Test Cases

10.7.1 Didactic Test Case: V&V Issues

A complete implementation of the V&V process for the two proposed test cases looks somehow unpractical and rather difficult. Nevertheless, some remarks will be developed to make easier a straight association of the above described concepts and their practical application. In case of the didactic test it looks interesting investigating the main highlights which were defined by approaching the verification analysis and how the validation of system was preliminarily conceived.

1. The *verification process* could be implemented as the DoD's adaption of the IEEE 1220 Standard suggests. Therefore a preliminary *Design Synthesis* is required as an input. It was provided by the *Analyses* described in previous chapters.

   If one resorts to the block diagrams, basically the system is composed by:

- A *rotor*, being resulting from the assembly of shaft and laying head.
- The *wire rod* running inside and then coiled.
- Two *active magnetic bearings*, with coils, housing and connectors.
- The *motor*, to make rotating the shaft.
- The *platform*, to be grounded (mechanically and electrically).
- The *power amplifier*, the *control system* (Electronic Control Unit—ECU) and several *sensors* (*temperature, position, rod detection, spin speed*).
- An axial magnetic bearing, if required.

Those elements will be verified in terms of *system integrity*, i.e. as a manifestation of *requirements* in components, subsystems and system, through allocation, traceability and completeness.

2. A *verification approach* should be selected and applied. In this case the Standard ASME V&V 10.1 may be suitable, because of the mechatronic nature of the system. It states that a *Design by Analysis* (which resorts to a numerical modeling of the system component) should verify the preliminary *Design by rules* (analytical and parametric approaches).

To proceed with this *verification* tools the *Development Test and Evaluation* should be preliminary defined and selected. They should support then the *validation* activity, being based on the *Operational Test and Evaluation*. In this test case, it is worth realizing that contents of the DT&E and after of the OT&E should be linked to the *Design Objectives*, which will be shortly identified. The activity shall be based on *analyses*, *inspections*, *demonstration* and *tests*, to be described in details.

3. The *Design objectives* related to this application include part of those proposed for a generic industrial product and some more specific for smart systems. Coming back to those previously described, some challenging issues were found for the test case.

   - *Simplicity*: suggests of building up the whole spoiler system as an assembly of modules, *one* is the *rotor* conceived for coiling and a *second* unit should just *store the coil*. This reduces the intricacy of system and makes it more adaptable, since some solutions for the storage or delivery unit could be connected to the rotor, depending on the steelmaking plant where is installed.
   - *Safety*: some issues could be defined in this field. As a rotor the system should avoid operating in regime of critical speed, dynamic instability, high temperature, electromagnetic incompatibility, severe unbalance, plastic behavior of material, severe wear condition or large friction, high noise, condition of rupture and material projection, rubbing (contact between rotor

and stator, prone to nonlinear and unstable dynamic behavior), low electrical insulation or bad grounding, seismic excitation, large aerodynamic drag or induced dynamic instability, high power consumption with high currents, unstable control action, constrained operation (in presence of obstacles to rotate for instance).

- *Security*: communication of operational outputs to the plant control system should be preserved by any unauthorized access as well as commands to the system operation should be authenticated.
- *Testability*: in this case each component should be preliminarily tested, the whole system should be open to be tested and monitored in service.
- *Transportability*: is a key issue. The weight of the structure should be compatible with an easy transportation, therefore modularity of rotor and of stator elements is welcome, moreover some suitable connections should be provided to perform the activity of suspending and moving the system and to install it on the plant.
- *Efficiency*: a suitable compromise between energy consumption and dynamics control is crucial, as well as a favorable ratio between the volume of the whole system and the power required to treat a given amount of wire rod.
- *Availability*: it should assure a continuous service during the whole time period corresponding to a cycle of manufacturing, without a stop, and between two maintenance actions.
- *Maintainability*: all subsystems should be able to be retained and restored to service, by a maintenance action.
- *Reliability*: should be high for all the failure modes which might significantly affect the health assurance of the operators or the overall safety of the steelmaking plant.
- *Survivability*: it is limited to complete the coiling of material in motion from the mill cages towards the storage.

Since this rotor is a *smart system*, several objectives related to the *smartness* should be considered. It is worthy noticing that they could be expressively adapted to this case as follows.

- *Simplicity*: in electromechanical coupling the magnetic field generated by coils assures an effective interaction through the gap between rotor and stator, the architecture of coils is simple and optimized to reduce the flux leakage.
- *Selectivity*: the rotor spin speed should be adapted to the speed of wire rod, imposed by the rolling mill.
- *Self-diagnosis*: the detection of rod presence inside the canned rotor shaft is a task of the diagnosis to be performed as well as the measurement of temperature and unbalance.
- *Self-tuning*: after a preliminary centering the system provides balancing.
- *Sensitivity*: actions of suspension are linearized to allow a simpler control.
- *Shape-ability*: it might be considered the option of a variable shape for the head nozzle.

- *Self-recovery*: if currents are too high the shutdown shall be automatically performed on some bushings.
- *Self-repair*: suspension and rotation will be possible even after a shutdown.
- *Stability*: it applies to the dynamics of controlled rotor.
- *Standby skills*: constant spin speed and standstill configuration are allowed.
- *Survivability*: vibration, instability, severe unbalance, heating and accidental stop of rod delivery are prevented and effects overcome.
- *Switch-ability*: to reduce the consumption and risks associated, the power amplifier work in switching mode condition.

4. According to the process proposed, the next step includes a preliminary definition of *analyses, inspection, demonstration, tests*. The Design Synthesis includes a preliminary model of control rotor, based at least on an analytical approach, with few degrees of freedom as it was described in previous chapters. This model allows predicting the critical speeds, the unbalance response, the dynamic instability and some the relevant structural damage conditions, leading to a material plastic behavior or rupture. This model requires some inputs like the system mass, inertia, stiffness, damping, weight, and unbalance condition. Control system parameters are calibrated to implement a defined strategy. Power amplifier is designed to operate as is required by the *switch-ability* targets. Those issues suggest a list of analyses and tests to be performed in the DT&E.
5. A complete description of the analyses, inspections, demonstrations and tests is intimately related to the different levels of system elements (parts, components, subsystems and system) and to the different tasks of the *verification procedure* (architectural completeness, functional behavior and performance, constraints). A brief overview is herein proposed, although a complete project, as the didactic case might be even considered, should be deployed through a very extensive description.

- Component DT&E.

To understand what kind of tests should be performed, the designer looks at the different analyses. The whole system shall be verified in terms of *static* and *dynamic behaviors*, in different *states*, being described by the State Machine Diagram.

The *lift-off* in case of active magnetic suspension is first accomplished. This requires that a verification of the system weight is performed and of the real effectiveness of the magnetic actuators. The virtual prototype of the rotor shaft and of the laying head should be used to estimate mass, inertia, weight and volume. They could come from the CAD model, but a direct *measurement of weight and inertia* should support this verification task. The first one is usually performed by *weighing the structure*, while a *pendulum* helps in measuring the polar inertia of the components. The *balancing equipment* is then used to detect the real condition of unbalance of components and whole rotor.

Stiffness and structural damping of the rotor shaft can be predicted by computation, but the real thrust of magnetic bearings should be verified by measuring its action on a *dedicated test rig*, built up after assembling of each device. The control

effectiveness on the actuation provided by each bearing is tested before that the whole system is composed.

The *standstill configuration* requires a preliminary *centering of the shaft* to be sure that each bearing suitably measures the error between the target position and that reached in whirling. This activity is associated to a check of the *resolution of position sensors*, which must allow a sufficiently precise location of the rotor shaft within the bearing housing to start the control action effectively. This is a matter of the whole system validation, but a preliminary *test of centering capabilities* should be performed in each bearing.

The *start-up with rotor acceleration* needs a complete knowledge of the electric motor performance as well as of all the numerical inputs of rotor above mentioned. A *full characterization of the electric motor* selected for this application is therefore included among the development tests. Accelerating the rotor might cause a severe condition of stress. This requires a verification of the *stress distribution* in components and connections, which leads to consider the whole system, as it shall be shortly described.

Similarly, in self-centered configuration and *supercritical speed rotation* the risk of yielding and rupture should be checked as the dynamic instability. The last step of *decelerating the rotor*, being usually performed by removing the torque of the electric motor, should be tested and numerically predicted, to avoid rubbing and other phenomena.

- System DT&E.

After a preliminary verification of components, the assembly is then analyzed and tested. In this case a double activity is promoted. An *analysis* based on some numerical methods is usually applied to investigate in detail the overall behavior, while the real response of the system is *tested* in validation. Both lead to some *demonstrations*.

A preliminary demonstration of *feasibility for manufacturing* of the whole system is given by resorting to a *virtual prototype*. It can simulate the assembly operation, by composing the system starting from the separated modules. This action should be completed in the geometrical modeling, before than a numerical simulation could be run. Particularly, the geometrical model is imported, for instance, into a *Finite Element code* to investigate the detailed distribution of stress, strain, displacement and rotations of the structure, when rotating or under a heating condition, or in acceleration. This analysis differs from the design by rules, since the model based on concentrated parameters is usually insufficient to describe the local behavior of materials, which may be investigated only once that a preliminary exploration of the system dynamics is performed and the details of the stem structure are defined. Moreover, a detailed dynamic analysis could be done by resorting to the *multi-body dynamics* modeling. It is suitable to describe the actions exerted by the subsystems, to investigate the system dynamic behavior in time and detect the actions, or the effects of critical speed, unbalance, electromechanical coupling and instability. If the contact between bodies is modeled and friction is considered their effect on the system behavior is also analyzed. Some additional

analyses could include, in this example, the drag of air and eventually its damping effect. To perform this investigation it is required introducing some tool dealing with the *computational fluid dynamics* (CFD) in connection with the other analyses.

The multi-body dynamics code can be exploited as the main tool for a comprehensive *heterogeneous simulation*, to verify the requirements, as, for instance, the capability of shaping the wire rod, within a defined time, provided that the wire rod is preliminarily modeled and even fully characterized by some tests, aimed at identifying its mechanical and physical properties.

- System OT&E.

The final *validation* of the system is usually performed on the prototype. A number of tests is applied to demonstrate the correspondence of each prediction with the real behavior of the system, its components and its materials. Basically, all the states, activities and sequences foreseen by the SE models could be used to define a complete list of tests. In present case all of states are gradually tested and the system is usually operated up to the maximum values of speed, acceleration, and temperature, into a safe and closed environment, to demonstrate its limits.

6. *Inspections* were a little bit neglected in previous description. Nevertheless, they could be crucial, especially when the integration with a main system is required. In the didactic test case some inspections allow realizing some critical issues of design. The main object of observation is the steelmaking plant. A platform connects the system to the main plate. The connection system consists of threaded fasteners. Vibration of the rolling mill in operation is highly suffered by those screws, therefore the risk of *fatigue and fretting fatigue* is high. This suggests to assure a stable connection, by reducing any clearance and fatigue effects.

Grounding might be another problem. It is required to insulate perfectly the housing of magnetic bearing to avoid any short-circuit or even to prevent the risk of fire. This is even important for *connectors*. Actually, some typical connectors used in power electronics are not standard in case of harsh environment exposed to the risk of fire. Inspection looks useful in realizing that, since many other devices exhibit special connectors, preserved against the risk of fire.

Moreover, this inspection reveals that the rotor shaft might be inadequate without a *canned rotor* configuration, in which a inner layer is in contact with the wire rod, but any contact with the coils is inhibited.

A short visit to the plant *control system tower* could reveal that the tools used to monitor the plant and the manufacturing process are based on some signal acquisition which covers a wide range of frequency and alarms are set on quite precise values. This remark, for instance, affects the selection of *sensors resolution* to make compatible the monitoring services and the detection of anomalous behaviors.

To summarize the V&V issues applied to the didactic case a sketch is proposed in Fig. 10.11.

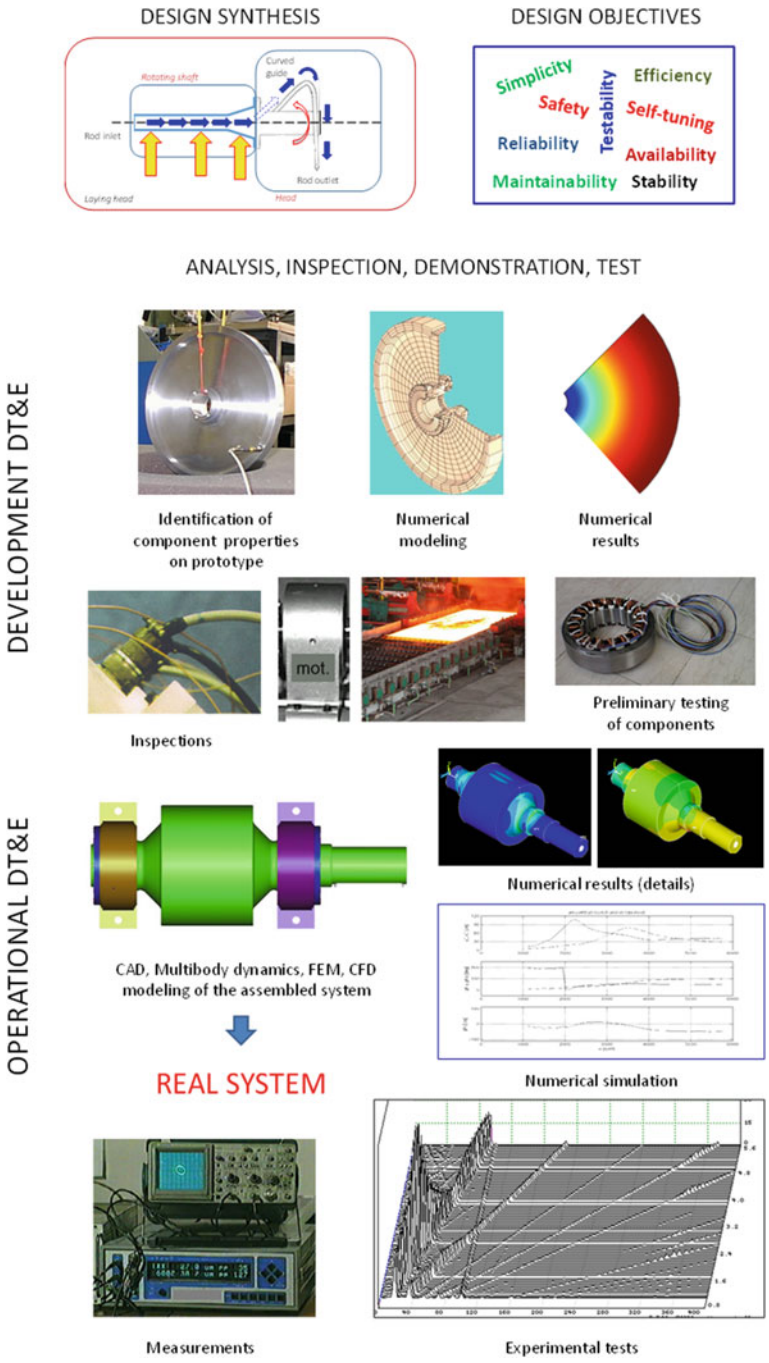


Fig. 10.11 Creative impression of the V&V process applied to the didactic test case

10.7.2 Industrial Test Case: The RAMS Analysis

The industrial test case shall be here briefly introduced to discuss the implementation of the RAMS analysis. This system is critical in relation with the strategy of boots actuation and affects the efficiency of the ice melting. If the ice layers are thin, the inflation of boots may be ineffective, because of the flexibility of ice, while, if the ice is thick, boots may be unable to break the layers. To assess a suitable control, the designer should investigate the system process both in nominal and failure conditions. Basically, the *functional* and *dysfunctional analyses* are required.

According to the process above described, the *Reliability Requirements Analysis* is performed as a first activity. This task requires that the design synthesis is available and the system layout is known. Moreover, some ice accretion profiles are used to verify the system. They are defined by some regulations, as the Certification Specifications of the European Aviation Safety Agency (EASA, 2017). The accretion profiles are directly related to the flight environment. Particularly, the Static Air Temperature (SAT), a Mean Volume Diameter (MVD) of droplets and a Liquid Water Content (LWC) of the clouds are all key parameters. The *Reliability Requirement Analysis* leads to the definition of some *Reliability Analysis Objectives* (RAO). In this case, requirements consist in setting a maximum allowable ice thickness over each protected surface of the aircraft, and a suitable time to perform the boots actuation.

Considering that a limit thickness of ice layers is fixed by requirement, for instance 20 mm, the activity of deicing system is precisely set up, by identifying a maximum time to completely inflate and deflate the boots, being about some seconds. Strategy of inflation is even important. If one boot is inflated at time, a maximum overlap between boots inflations has to be set up. The system performance is defined, when both the overall time to complete the activation of all boots and the maximum power required are fixed.

It is worth noticing that those requirements just cover a *nominal behavior*. However, to investigate the *dysfunctional behavior* a screening of failures causing some anomalous service conditions is added. In the test case, a typical scenario of failures may be described by analyzing the behavior of system valves. They tipically supply pressure for the alternate inflation and deflation of deicing boots, by following a specific sequence, provided by the system controller. Each boot is divided in two chambers, therefore the related valve inflates a compartment and, in the meanwhile, deflates the other one. The mechanical components of this valve are basically spools, springs and retention screws. They assure a fairly high degree of reliability. Their failure conditions are associated to fatigue and wear. Therefore, they should appear in the latest part of life. By converse, the valve actuation is based on electromechanical actions, through a magnetic field generated by some coils, and is imposed by an electronic control system. Those components exhibit a lower reliability. Failure of either the coils or of the control switches is critical in three cases, i.e. when the valve remains completely closed, always open or it opens and closes only partially. In the first two failure modes, the valve is stuck in its

initial position or it reaches the rest configuration, under the effect of springs. The impact on the deicing efficiency is the same. The boots remain either inflated or deflated, therefore no action is applied on the ice layers. When the valve partially works and is never completely closed or open, deicing capabilities are exploited, but their performance is degraded.

Those reliability objectives can be assessed through the numerical simulation (Tundis et al., 2017). The occurrence of failure conditions can be taken into account. The *System Modeling* phase is consequently performed. Nominal and dysfunctional behaviors are represented by resorting to the SysML. An example of the *Block Definition Diagram* (BDD) of the deicing system is given in Fig. 10.12.

Some subsystems are considered and are associated to each couple of protected surfaces distributed over the aircraft profile. Actuators applied to external and internal portions of wings, horizontal and vertical stabilizers and engines inlets are all included. The actuation valves and the sensors to detect the ice complete the system layout.

The integrated design of nominal and dysfunctional behaviors of the valve is described through a *State Machine Diagram* (Fig. 10.13). The left side shows the intended behavior and the regular operation of the valve corresponds to nominal states 'Closed' and 'Full Open'. The right side includes all the states which lead to an anomalous behavior, like 'Fail To Open', 'Fail To Close', 'Partial Open' and 'Partial Closed'.

The system reliability is investigated by implementing the *model* of the deicing system in a numerical simulation environment like the Matlab/Simulink® as is shown in Fig. 10.14. Several modules are created to include all the issues of system behavior previously defined. The ice accretion, as is foreseen by regulation, is a part of the system model, which includes all the physical components aimed to provide the ice protection, such as the pressure regulator valve, the dual distribution valve and the boots, applied to several surfaces of the aircraft. Each wing is equipped with two boots, one external and one internal, closer to the fuselage. Two boots are applied to the horizontal and vertical stabilizers, while each engine is equipped with a single boot.

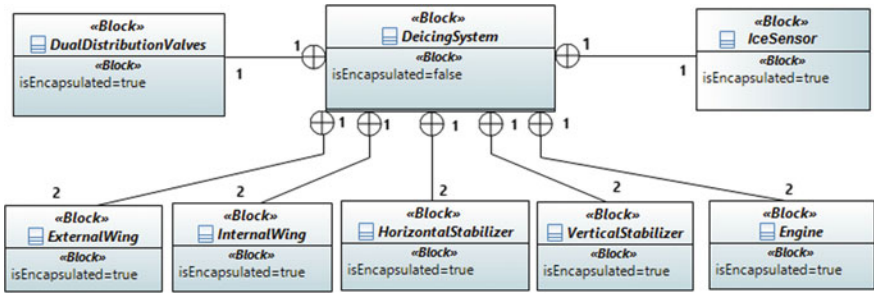


Fig. 10.12 BBD of the deicing system applied in the industrial test case (Tundis et al., 2017)

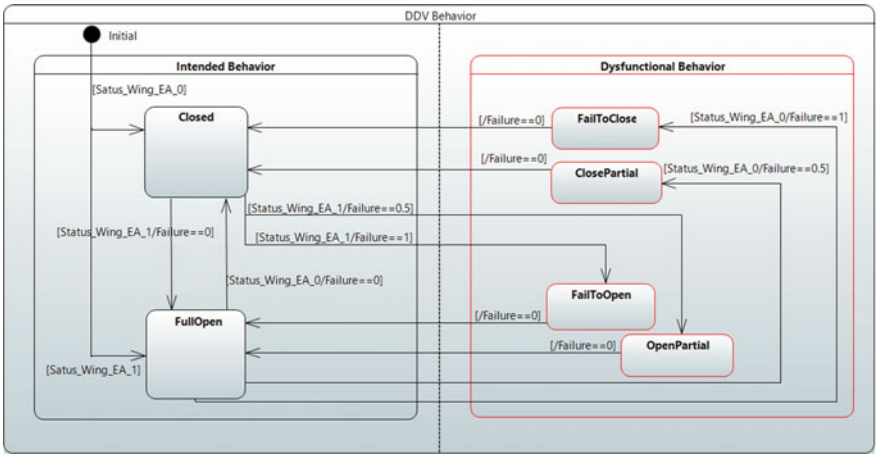


Fig. 10.13 State Machine Diagram of the valve of deicing system (Tundis et al., 2017)

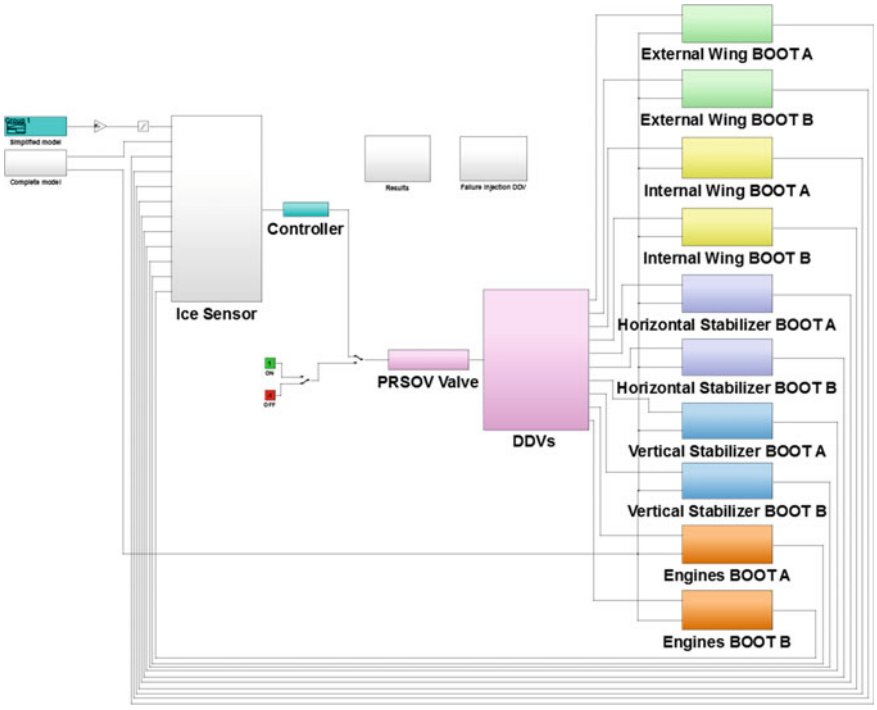


Fig. 10.14 The Simulink® simulation model of the deicing system

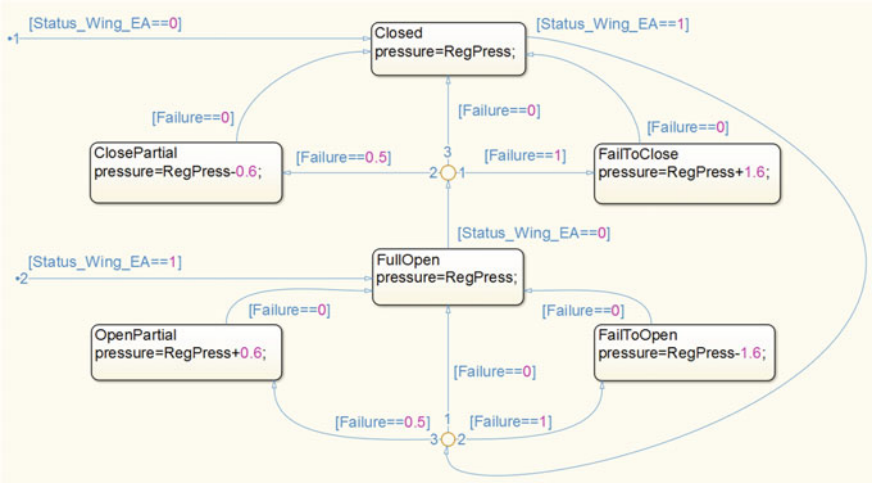


Fig. 10.15 The Simulink® Stateflow of the deicing system

The numerical simulation of the behavior of the valve is performed through a *Simulink Stateflow*, depicted in Fig. 10.15. Main model inputs are the characteristics of valves and boots, respectively, and some parameters of the flight mission profile. Calculations provide the ice thickness per actuator, the ice melting rates and the power consumption. A typical *mission profile*, as, for instance, is described by the regulation CS-25 (EASA, 2017) can be assumed. It includes several phases as take-off, climb, cruise, descent, first holding, first approach, first landing attempt, go-around, second holding, second approach and final landing. Results are therefore related to each phase of the flight mission. The behavior of each component is described by some characteristic equation and related to some design parameters. The regulator valve, for instance, is modeled through a second order transfer function, and the dual distribution valves through a series of signal generators. The behavior of boots is predicted by a second order dynamic system.

The *Results Assessment* is performed by comparing the numerical results computed by assuming nominal and dysfunctional operating conditions. The ice thickness evolution in *nominal conditions* is first predicted, as in Fig. 10.16. It is immediately checked that requirements are fit, i.e. that the ice thickness never exceeds the maximum allowed.

The failure modes previously described are then applied within the system model to investigate the valve behavior, under different *failure conditions*. Since the valve controls the air pressure supplying to the boots, when the valve remains either open or closed, the ice removing capability is lost. As an example the failure is assumed to occur to a wing boot, at take-off and affects the climb phase (Fig. 10.17). For the selected ice accretion profile, the effect is severe. Nevertheless, since the assumption of recovery capability is made, nominal conditions are restored at the top of climb.

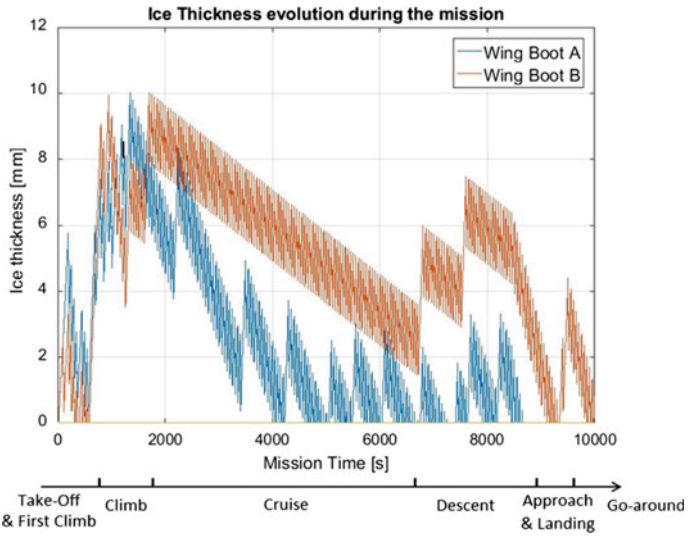


Fig. 10.16 Simulation results for nominal behavior, without failure

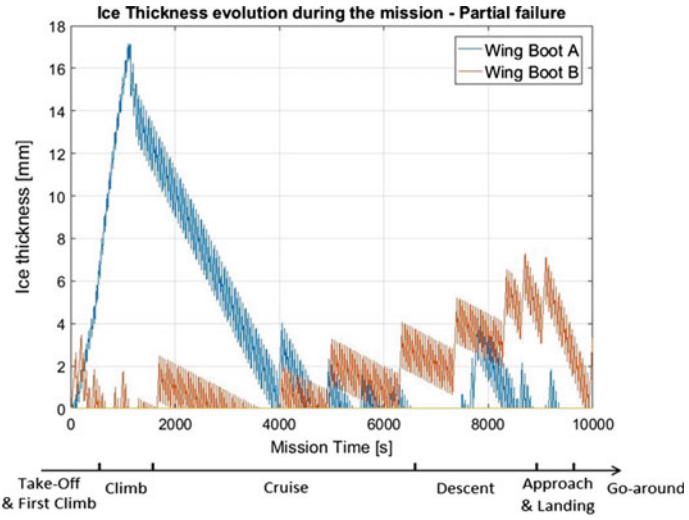


Fig. 10.17 Simulation results for dysfunctional behavior (one boot inactive at take-off, then recovered during the cruise)

As is pointed out by the example, the combined use of functional and numerical modeling techniques allow investigating the performance of the system in terms of reliability through a well driven path, at the early steps of development and quite deeply, if they are compared to some qualitative methods, based on FTA, FMEA.

This approach looks effective, allows a high reuse of simulation models, and helps the designer in decomposing the system complexity. Moreover, the applied approach allows integrating several issues of design, as the regulation requirements concerning the mission profile, the system control and the failure modes.

References

ASME. (2017). *Verification and validation symposium.* Las Vegas, Nevada, USA: ASME.

Debbabi, M., Hassaine, F., Jarraya, Y., Soeanu, A., & Alawneh, L. (2010). *Verification and validation in systems engineering*. Berlin Heidelberg: Springer Verlag.

Defense. (2001). *The DoD systems engineering fundamentals.* Fort Belvoir, VA: Defense Acquisition University Press.

Garro, A., & Tundis, A. (2015). On the reliability analysis of systems and SoS: The RAMSAS method and related extensions. *IEEE Systems Journal, 9*(1), 232–241.

ISO15288. (2015). *Systems and software engineering—System life cycle processes.* ISO/IEC/IEE15288.

Lee, E. A. (2008). Cyber physical systems: Design challenges. In *11th IEEE International symposium on object/component/service-oriented real-time distributed Computing*, Orlando (pp. 440–451).

Shamieh, C. (2012). *Systems engineering for dummies*. Hoboken: Wiley.

Tundis, A., Ferretto, D., Garro, A., Brusa, E., & Mühlhäuser, M. (2017). Dependability assessment of a deicing system through the RAMSAS method. In *2017 IEEE ISSE 3rd International Symposium on Systems Engineering*. Wien, Austria: IEEE.

Walden, D., Roedler, G., Forsberg, K., Hamelin, D, & Shortell, T. (2015). *Systems engineer-ing handbook of INCOSE* (4th ed.). Wiley, Hoboken.

Wasson, C. (2006). *System analysis, design and development*. Hoboken, USA: Wiley.

Chapter 11
Systems Engineering and Product Lifecycle Management (PLM)

Abstract This handbook is mainly aimed at describing the activities related to the ALM, applied to the industrial product coming from a material processing. Nevertheless, a suitable integration of tools and processes with the PLM is obviously required, as is herein briefly described. A Configuration Control Management is strictly required to track the changes applied to the developed product along the lifecycle. This needs to build up a platform which includes the tools to be interoperated. Exploring how this might be done is herein proposed.

11.1 The Big Picture

The MBSE enables the designer to manage the whole product lifecycle and to assess the *configuration* of the system under development. This turns out into a systematic control of the system layout along the sequence of actions performed during its development, by allowing, recording and motivating the *changes* which are progressively introduced. Straightly speaking, the system should be precisely identified step by step in its evolution, to assure that a unique and accessible configuration is shared among the operators. This implies a careful assignment of some *version identifier* items. Changes must be operated by *authorized users*, who are allowed to access to the *tool chain* composing the *platform* for the product development. They should be evaluated and their consistency demonstrated, through several checks. Every change must be recorded, tracked and identified.

This need is behind the main action of integrating all the tools previously described in a unique working environment, managed by some software tool which implements the *Product Lifecycle Management* (PLM). Particularly, an effective integration between the *Application Lifecycle Management* (ALM) and PLM activities is required, and the *Product Data Management* (PDM) needs to be carefully deployed. Moreover, when a SE working *platform* is set up to perform all the activities already described in previous chapters, and several other ones, being related to manufacturing, delivering, service, maintenance and disposal, a *Configuration Management Process* is applied.

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_11

Analyzing the strategy of the *Configuration Control Management* (CCM) allows understanding the criteria driving the design of the platform, allowing the integration of the SE within the PLM. Requirements and specifications for the *tools* composing the chain supporting the platform (or *tool chain*) could be even caught.

The *object* of the CCM is the *configuration baseline* of the system, and of its subsystems. The *actions* of management are basically the reviews and acceptance of requirements, design and product specifications. Particularly, once that some main system elements characterizing the configuration are identified as crucial items to be controlled, the real control consists in performing a continuous action of *evaluation*, *verification* and *validation* of changes proposed and requested by the operators. A goal of the configuration control is avoiding any overlap between different *versions* of the product, when developing, as well as assuring the highest level of *security* in the data object of the PLM. Producing the related *documentation* along the system development is another key issue of this process. This activity is performed by introducing some *audits* in the whole process, being playing the role of *milestones*, and by applying a clear *decision-making* approach.

Those elements become the requirements of the infrastructure to be set up to perform the system integration, through a rational *digitalization*, i.e. the platform of software tools for the PLM. In this chapter, some outlines about the role of the MBSE in the PLM, and of the interoperation of software tools at higher level to accomplish the integration between the ALM and PLM, are proposed. A more detailed analysis of the current methodologies can be found in some other handbooks of this series, which focus more on the system manufacturing and on the PLM itself.

11.2 The Configuration Control Management

Monitoring and controlling the evolution of the system within the PLM process is the main goal of the *Configuration Control Management* (CCM). Policies and procedures have to be stated at the beginning of the implementation, to select and connect the tools. Therefore, a *Configuration Management Plan* is usually written.

A first action performed concerns the identification of the *Configuration Items*. In this phase, the MBSE is significantly effective, since the *Product Breakdown Structure* may be directly used to define, since the beginning of the product design, the hierarchy of system components and elements. They are objects of the configuration control, or *items*.

As the "V" and other diagrams describe, the product development foresees several steps and those items need to be monitored, stabilized and identified in correspondence of each one. This introduces the need for a *Configuration Baseline*. The rationale of the product development is monitored and somehow captured in several steps, to realize and characterize the configuration items. Defining steps, milestones, or simply control points, is the goal of the *Configuration baseline*,

which draws the path followed by the PLM action to make the system evolving until that it could be manufactured and finally delivered.

Once again, a convergence between the tradition of the industrial CCM and the tools of the MBSE could be found in this task. As the MIL–STD–973, for instance, defines the baselines can be distinguished into:

- *Functional baseline*: is related to an approved configuration, whose functional and interface properties are verified
- *Allocated baseline*: is aimed at demonstrating the allocation of functions to configuration items
- *Product baseline*: it is associated to an approved functional and physical configuration of product, defined in its configuration items.

This approach is compliant with the definition of the *FBS*, *LBS* and *PBS* which describe the main milestones of the MBSE.

The *Configuration Control Management* is then specifically performed. It is aimed at assuring that every change introduced is first properly identified and documented, then evaluated in terms of the *impact* applied to the system and finally approved and verified.

To define some *audits*, in case of material product, at least the *functional configuration audit* could be introduced, at the end of functional modeling, to verify the requirement allocation and traceability, and a *physical configuration audit* may verify and validate the correspondence between the system "as is built" against its configuration, when the system looks "as designed".

To define the role of the CCM within the PLM, a sketch is proposed in Fig. 11.1. As the SE describes, the product development in its design and manufacturing issues is deployed from the detection of customer needs to the delivery and service. To each step a configuration is associated. Therefore, the product looks like is "required" first, "conceived" after, and really "designed". As soon as it is manufactured, testing, verification and validation are performed. Consequently, it looks like "planned", "built" and as is finally "maintained" in service.

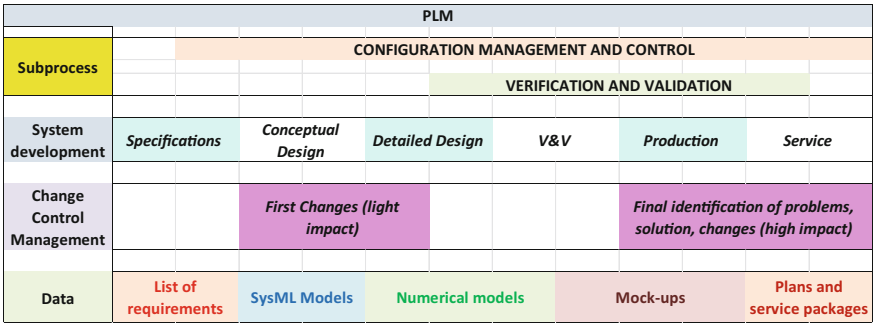


Fig. 11.1 Sketch of the PLM process in a technical domain applied to a material product

The most relevant interaction of the operators with the baselines is observed in the first activity of *design*, where requirements are assessed and the product is modeled, through both the functional and physical modeling techniques described in this handbook. A sort of loop is generated, by resorting to the methodology of the SE. Changes may be even many, but the impact is fairly low in terms of cost and resources, while the system configuration might be significantly modified. By converse, a second intensive change action could be performed when the system is produced, as the quality assurance methodologies are applied or the final validation with the customer is faced, especially in service. Problems are usually first detected, a new solution is analyzed by the manufacturer and then proposed and applied. The impact on cost might be large. Changes may affect quite a lot the baselines, even when they are not so appreciated in terms of configuration, if the appearance of the system does not change so much. If one looks at the whole PLM process, the Configuration Control Management includes several steps, while the V&V process, for instance concerns a few of those.

It is worth realizing that for the above described configurations and development steps, there are some models or configuration items, associated to some data which are managed by the PDM. Requirements are related to the systems as required, but the functional models are strongly supported by the MBSE approach and by its tools, as the SysML language, leading to the definition of a PBS, for instance. Physical models, based on the numerical modeling and simulation, support the final step of design as well as the V&V. Material and digital mock–ups are used in production, while plans and detailed information like the bill of materials (BOM) identify the real parts built in manufacturing. This workflow defines even the needs of the manufacturer and how to exploit a platform to accomplish the goal of delivery the final product.

11.3 The Platform Building

The need for an effective infrastructure is behind the implementation of the CCM process. Basically, building up the platform requires:

- a clear statement about the *workflow* to be implemented, to define when and how activities must be done and through what kind of tool;
- a *unique data source*, to get the required inputs, to store the outputs;
- a set of *secure repositories*, when data are collected and stored, by monitoring the access and the changes;
- a bright *review process*, to effectively perform in development and management;
- a set of suitable criteria, policies, rules and identifiers, to *control the access* and the *data sharing* among the operators;
- a *tool chain* of *interoperated software*.

Those issues should need another textbook to be deeply investigated, but a brief impression of the main topics related to an industrial implementation of the MBSE within the frame of the PLM is herein proposed.

11.3.1 The PLM Collaboration Model

As in the SE, where at least two main models were introduced to represent the process and the system, respectively, in the PLM a *collaboration model* is required too. It defines the network of stakeholders as a number of actors, collaborating through the network, by playing the role of *customer*, *system developer unit* or *supplier*. This model describes, like in the use case diagram, the interactions, interfaces, dependencies and tools applied to share data or to communicate. It is used to understand the connections required into an Internet of Things service, when used, or into the cloud, or simply within a structured network. This is a roadmap for the platform builder, who can trace:

- the number of communicating *actors*
- how *the network* is *distributed*
- how large is the *heterogeneity of tools* used
- the need of *protocols* to be used to interoperate the units
- the amount of *data* produced and to be stored
- the need of *repositories*
- the spatial *location of stakeholders.*

This model, looking quite simple, actually is extremely useful to define a "*virtual co–location*" of operators through the network. It is strictly required to design the strategy of *data replication* (Fig. 11.2).

11.3.2 The PLM Functional View

The first audit above mentioned, concerning the *functional configuration*, may resort to the results of the functional modeling performed by the MBSE, to provide a corresponding *functional view* of the system. It defines both the interpretations "as required", first, and "as conceived" then. This largely benefits of the requirements elicitation and refinement through the MBSE approach, along the *requirements baseline*, as well as of the models described through the SysML, along the *functional baseline*. The *Functional Breakdown Structure* is the reference. Similarly, the *allocation baseline* prepares the second audit, focused on the *physical configuration*, exploiting the *logical analysis*, first, and the *physical analysis* after, by resorting to the *Logical* and *Product Breakdown Structures*.

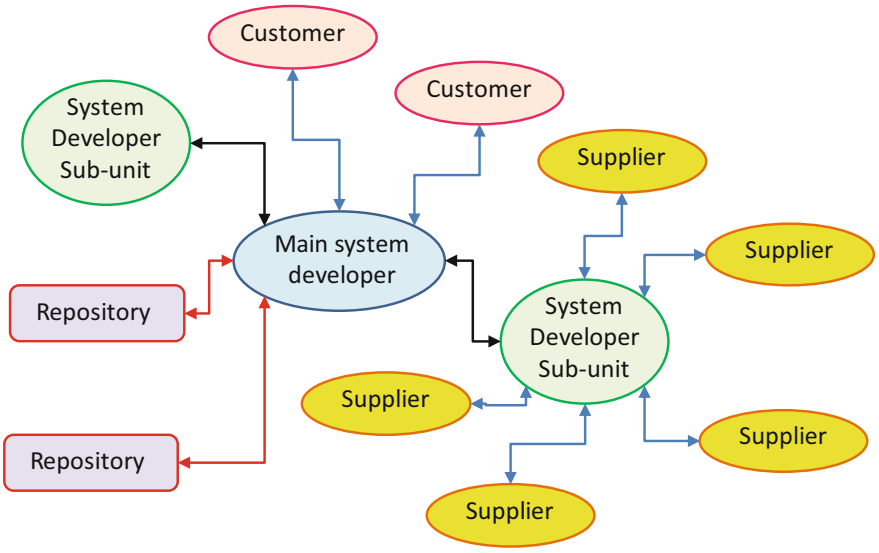


Fig. 11.2 Example of collaboration model in PLM

A linear correspondence between baselines, development steps, configurations and their items is found, when the *traceability of requirements* is investigated, by proceeding forward from needs to product, between product and needs, when the same path is followed backward.

It seems quite obvious, but actually two important consequences are found. A first correlation is detected between the concepts of the PLM process and the digital products provided by the MBSE approach. This greatly enables the system developer to clearly implement the strategy proposed in several PLM references. As it shall be discussed in next sections, there is a similarity between the design analysis workflow and the safety engineering analysis workflow, which can be exploited to apply the same approach and to connect the two analyses within the same tool chain. Items are clearly identified and even the tools to be applied are described, thus driving to a straight investigation about the interoperability of some dedicated software. The whole picture is defined in the same sketch, being representative of the platform enabling this design process (Fig. 11.3). Requirements are specified by functions, and even satisfied, functions are directly allocated to system elements (subsystems, components, …) and implemented by those, elements are physically implemented by parts.

11.3.3 The PLM Data Model Analysis and the Tool Chain

The structure above defined for the complete traceability of requirements suggests of setting up a hierarchy of software tools, where a main system integrator (a PLM

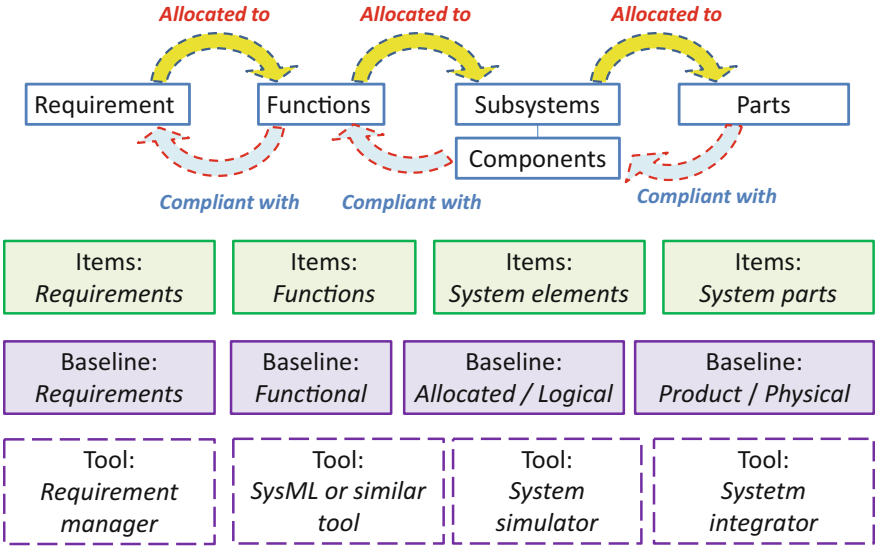


Fig. 11.3 Overview of the traceability of requirements and description of the Configuration Control Management and of its items

dedicated tool) provides a sort of high level interface. All the operators can be connected through it and several elements of the tool chain can be activated and operated, depending on the degree of accessibility allowed to each user. Through an effective interoperability this main tool should allow sharing the data with repositories and other users, and accessing to the network, but even running some dedicated tools for reading and updating the requirements, navigating and eventually modifying the functional models, running the simulations, creating the digital mock–ups and refining them.

As the SE provides the roadmap to implement the whole PLM, this software is like a dashboard, allowing the system developers to monitor all the activities performed. When this tool is requested of exchanging data with another one or directly to activate a function, a suitable connection should be exploited. This means that either a point–to–point connection is applied, when the vendors of the software products already included among the options, or the communication is based on a standard connector, for instance compliant with the *Open Services for Lifecycle Collaboration* or *OSLC*. This enables the interoperation of tools, as it was described to introduce the heterogeneous simulation.

From this point of view, two basic remarks should be added. Designing the *tool chain* is a key feature of the implementation of the SE, and technologies available in computer science may either accelerate or make slower the effective application to the industrial product development. The cost is quite high, at the beginning, as a company sets up the platform, but an immediate reuse in several projects assures that it could be quite fast amortized.

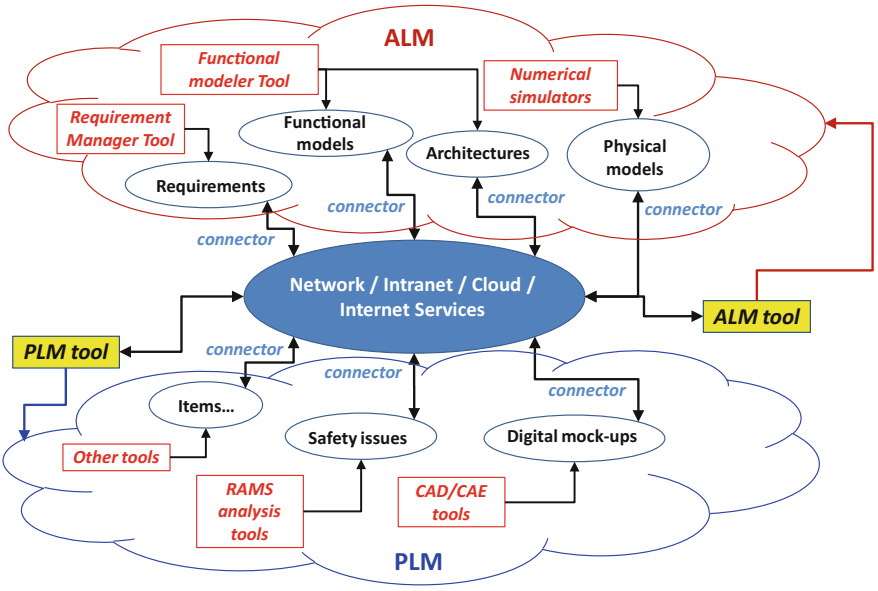


Fig. 11.4 Integration of the ALM and PLM processes and overview on the data, analyses, connections and tools involved

The *model of collaborative work* implemented may either enhance or brake the platform building and affects the overall performance of the methodology applied to a certain technical domain. Therefore, those are crucial issues for the development of the MBSE approach and for its straight implementation.

The tool chain defines the neighborhoods of the *community* involved within the development of a given product. It might be observed that the concept of *virtual square* for collaboration is deployed, as in Fig. 11.4, where a network connection or cloud services could connect each other the operators in a unique working environment. Some main tools manage the two set of activities in ALM and PLM (this one often includes the ALM) while each specific action is performed by some dedicated software, handling a set of data of different type, and are connected to the working environment through some suitable connectors.

To make effective this implementation some issues must be carefully defined:

- The *strategy to integrate the software tools* for the design activity
- The *control action* in the configuration change management
- The technical *interoperation of tools*
- The *integration between analyses*, as for instance, the connection between the functional and dysfunctional design.

Those are the challenging issues of the practical implementation of the MBSE and simultaneously they measure the real performance of the commercially available tools in effectively deploying the SE.

11.4 The Tools Integration and Interoperation

A key focus in platform building is the integration between different tools in terms of compatibility of actions and results exchanged. Moreover, the real possibility of automatically exchange data and information should be assured by suitable connections and a straight interoperability. To make an example of this activity, some details of the integration between tools are herein analyzed.

To implement the *workflow* of actions previously described in the design activity in both the test cases developed in this handbook, it was required to define the selection of tools. The data to be exported and imported in passing from one tool to the next one were then defined. Depending on the tools to be connected, a suitable way to operate the connection was identified and this implied resorting to either proprietary or standard connectors.

The tool chain used within the frame of the project to whom belongs the industrial test case is depicted in Fig. 11.5. The IBM DOORS® was proposed to manage requirements, after a preliminary elicitation made by the user, to convert the information known in terms of customer needs. The requirements listed in the IBM DOORS® are imported, in this case, into the IBM Rational Rhapsody® to perform the functional modeling, by means of the SysML language. This connection is already foreseen by the vendor, therefore it is easily set up. When the *heterogeneous simulation* is run by resorting to the Simulink® simulator, a standard connector is required to import the model and interoperate the tools, as it was seen in previous chapters. Nevertheless, the connection is established. To manage the models of the heterogeneous simulation in connection with other features of the PLM process, a main PLM software tool shall be envisaged. It can be based upon an OSLC compliant connector to communicate with the IBM Rational Rhapsody®,

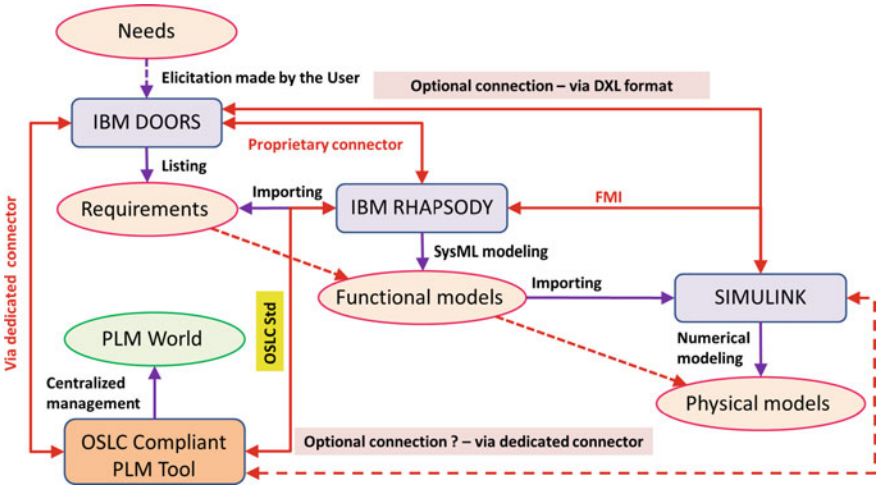


Fig. 11.5 Example of integration with the PLM process of tools used for the industrial test case

while some dedicated connections were developed by vendors and are now available to interact with the IBM DOORS® and the Matlab/Simulink® environment.

The strategy applied in this test case assumed that the IBM Rational Rhapsody® could deserve as a main tool in the ALM activity and exchange data with the PLM main tool This is a layout sometimes implemented by several partners of the project used as an example. The IBM Rhapsody® plays the role of a dashboard for the operators. Some direct and optional connections may be set up to directly read the contents (statements and models) of the requirements manager (DOORS®) and the numerical simulator (Simulink®), which are made already available by vendors. When they are activated, the operator navigates and explores the numerical simulation or the list of requirements, independently on the interaction with the SysML modeler.

The effectiveness of tools integration depends on the performance of the proposed workflow and specifically of connectors. Connection between tools sold by the same vendor obviously makes somehow simpler the integration and faster the interoperation. By converse, the features provided by different tools motivate a careful selection and composition of the tool chain. This is the current limitation to an effective and straight implementation of the MBSE in practice, especially in the industrial product development.

It is true that a wide action is currently promoted by several software vendors to provide a fully integrated tool chain, being able to interoperate the tools through assessed connections. In the same example, the companies involved in the system design tested another layout, being based on the IBM platform Jazz®[1] and the IBM Team Concert®, by resorting to the tool IBM Rhapsody Design Manager® to manage the system integration. This solution, despite of some change of the features provided by the tools, makes possible interoperating some homogeneous tools, as they are sold by the same vendor.

In the didactic test case, this approach was applied by the PTC, who acquired the modules to perform the MBSE as the Artisan Studio® by Atego, to create an integrated tool as the PTC Integrity Modeler®. This action made possible, for instance, configuring the layout described in Fig. 11.6.

The elicitation of requirements was made through the IBM DOORS® as in the simulated industrial environment of the steelmaking technical domain, but a PTC module could be even introduced to make the tool chain complete.[2] However, the interaction between customer, supplier and system developer is even an issue of design of this layout, as it was there tested. The difference with the industrial test case is that the strategy applied by the user favored a proprietary connection between the PLM manager tool and the functional modeler, by developing some partnerships with other vendors or simply some dedicated connectors to integrate the requirement manager, when the proprietary module is not selected, and the numerical simulator.

[1]For further information please visit https://jazz.net/products/rational-team-concert/.

[2]For further information please visit https://www.ptc.com/en/product-lifecycle-management.

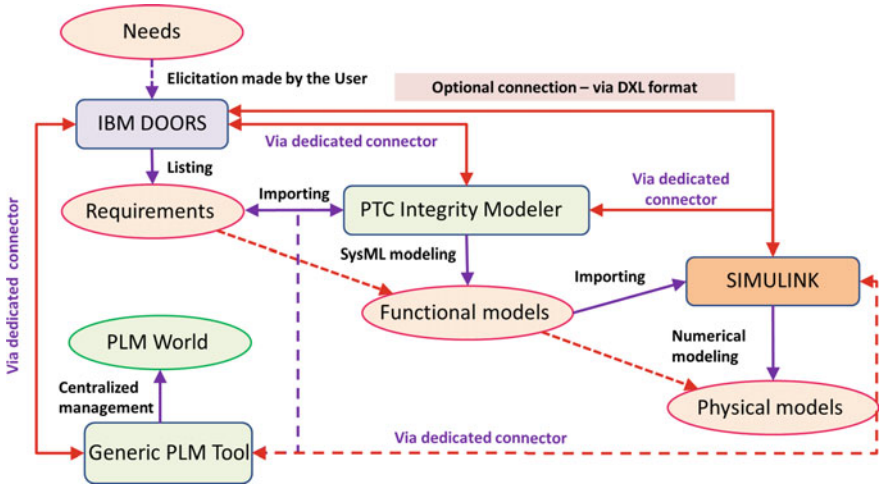


Fig. 11.6 Example of integration with the PLM process of tools used for the didactic test case

As it was mentioned at the beginning of this handbook, some other software vendors already developed alternative solutions, making possible to define other layouts. Those above mentioned were introduced as a picture of the experience performed in the industrial implementation of the MBSE, in recent years and to give an impression of the difficulties associated. From this point of view, they do not reflect the latest up–to–date solutions proposed by the cited vendors, but just the problems that gradually overcome in setting up their latest products, as they could be also identified by the large community of industries, academies and vendors within the frame of the ARTEMIS JU CRYSTAL project ("Acceleration of Safety Critical Systems Engineering" 2013–2016).[3]

11.5 The Configuration Control Action

Once that a tool chain is assessed, a crucial issue concerns the definition of the Configuration Control process through the tools and the steps of product development. Particularly, a suitable sequence of authorized actions is required to assure the overall integrity.

The source of change request may be either the designer or manufacturing teams. In a review of requirements, the system developer might realize the need of changing some detail in the system configuration, but even the production line might detect a difficulty in manufacturing, assembling and integrating parts, components and subsystems, thus requiring a different solution. Provided that a change

[3]Public documents and relevant results are still available on line at www.crystal-artemis.eu.

request is formalized and sent to the system developer team, according to the MBSE, it has to be associated immediately to the function and the requirement which motivated the presence inside the system layout of the part under discussion. Therefore, the request is processed through the requirements manager tool, to identify first the associated requirement, which is allocated to the part to be modified. The impact of change has to be checked. A first action identifies the contents of the requirement to be changed, but to investigate the consequences of change, the ALM tool chain is used. This allows allocating the change, as the requirement was allocated to the corresponding functional and logical blocks, in the system model, without modifying its elements. To check the impact of this change, a functional analysis is performed, by assuming the potential change as active, and eventually, the related numerical simulation is even run. If this step states a preliminary compatibility with the change proposed, it is allocated to the system parts, by acting on those which were initially object of the change request. To check the suitability of this change, the verification and validation process can be applied. If the result is favorable, the change is consolidated and noticed through the PLM tools, as a change note (Fig. 11.7).

As the above described process points out, the configuration control action is fast and effective if a platform is built and the workflow from the needs to the product is clearly defined and based on models to be run within some interoperated software tools. This motivates the integration process design and an optimization of its elements, as well as of the process itself.

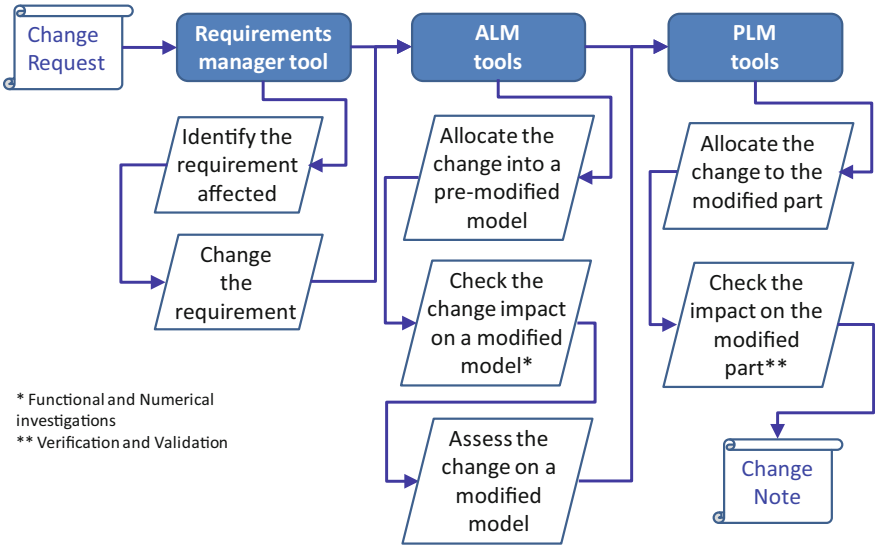


Fig. 11.7 Example of change control implementation

11.6 Integration Between Analyses Within the Tool Chain

11.6.1 Integration Between Design and RAMS

An additional issue, to complete the integration of the system as well as of the tool chain used to design and manufacture it, is the interconnection between different analyses performed within the same workflow and particularly between tools. A typical example is the interaction between the design process, based on a functional modeling and the safety engineering applied to the system integration, which includes a preliminary prediction of the dysfunctional behavior.

This task allows realizing how large is the impact on the product development of the digitalization of activities based on models, as the MBSE promotes. In this case the rationale previously described for the design is somehow replicated, but a different platform is set up. The challenging issue is making the two platforms interoperating to allow, simultaneously, a reduction of the resources required to develop the tool chain and a smart cross check of the system figures and performances in operation through the models developed.

As it was already described in Chap. 10, despite of the tradition of performing an investigation about the degree of reliability and safety of the system, after its final assessment in terms of the PBS, the model based approach may help in performing a suitable refinement of the safety requirements through a gradual investigation, which can be associated to the design operation, step by step. Particularly, the analogy can be found if the functional analysis performed to design the system is associated to the functional hazard analysis (FHA) useful to identify the failure conditions in dysfunctional behavior. As a second step, the logical architecture is then defined in design, like some reliability and safety targets can be identified by proceeding to a reliability and safety requirements allocation. Finally, as the physical architecture is assessed in design, the reliability and safety conditions in operation can be predicted, in the safety engineering process.

The workflow to transform the typical safety and reliability analysis based on some tools like the FMEA and FTA, into a model–based approach is simply described by the above-mentioned analogy. This helps in setting up suitable tool chain, to be carefully connected to the design models.

In the industrial test case it was done, according to some technical standards and best practices applied to the aeronautical domain. The Aircraft and System Development Processes defined by the ARP 4754, for instance, may find a direct correlation through this approach with the Safety Assessment Process Guidelines and Methods provided by the ARP 4761. Assuming that the design activity is performed through the IBM tools, by following the approach above described, the requirements management may be listed by the IBM DOORS® or the new product Doors Next Generation® (DNG), the functional modeling by the IBM Rational Rhapsody® and the subsequent integration by the IBM Jazz® platform, through the IBM Design Manager®. In addition, some typical tools of the safety engineering consist in as toolbox performing the set of RAMS analyses, in cooperation with a

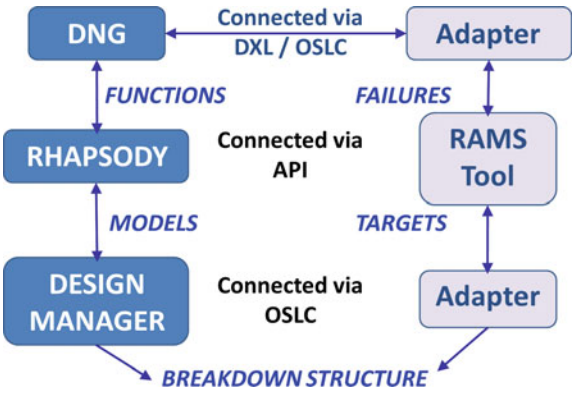


Fig. 11.8 Implementation of the design and safety engineering workflows in parallel as in the test cases

data link connector. In the test case, it was for instance done by resorting to a domain oriented RAMS analysis tool. A sketch of the preliminary tool chain tested is proposed in Fig. 11.8. A connection module could allow joining the RAMS tool and the Design Manager, by resorting to some OSLC compliant connector.

As a result of the construction of a interoperated working environment, the RAMS analyses may be started early in the product development and grow up in parallel with the system design. Moreover, it can be really based on models, more than on some analyses whose results are collected in tables, less manageable than the digital models for a complete allocation of functions and dysfunctions.

The benefit of this integration can be realized if the typical processes performed by safety engineers are analyzed, as is herein briefly proposed.

11.6.2 Integrated Analysis

In case of RAMS analysis, the integration of the two platforms allows proceeding quite simply in subsequent steps.

The functional modeling provides a preliminary *Functional Breakdown Structure*, which can be transformed into a list of functions and assumed as a set of functional blocks by the Reliability Manager tool. This one analyzes the functions and associates a number of potential dysfunctions, by performing the *Functional Hazard Analysis* (FHA) and allows a first updating of requirements and of functions in the design workflow system (Fig. 11.9).

The architecture conceived in terms of *Logical Breakdown Structure* is then transferred from the functional modeling to the reliability workflow and transformed in logical blocks to analyze the *reliability allocation*. This allows a selection of physical products and compiling the architecture of the physical blocks to be transferred back to the design workflow to assess the *Physical Breakdown Structure*, with reliability targets included (Fig. 11.10).

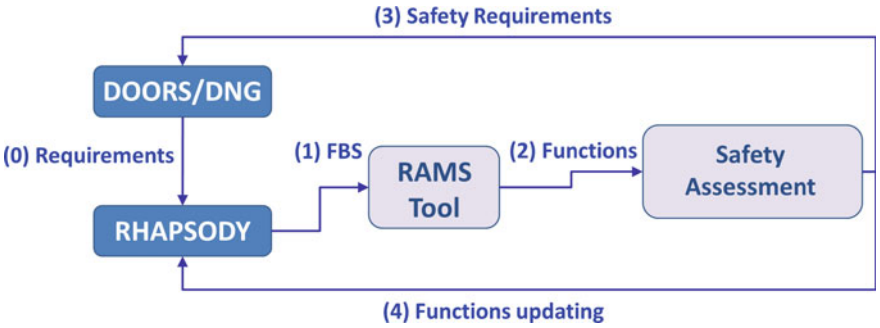


Fig. 11.9 Functional Hazard Analysis loop between design and safety engineering workflows (sequence defined by numbers)

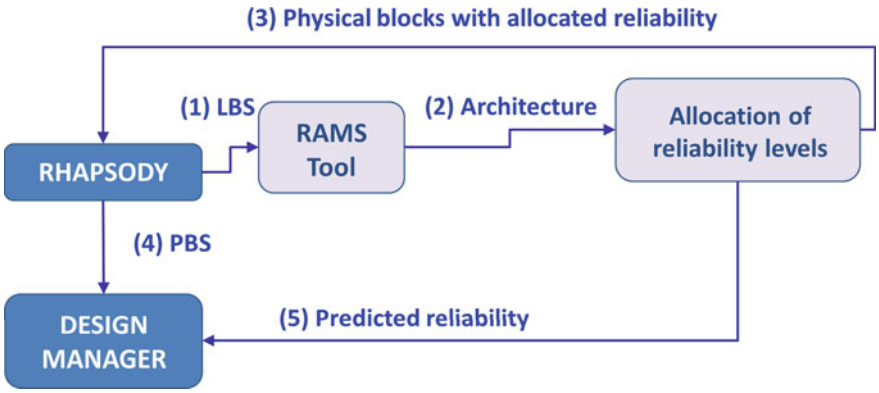


Fig. 11.10 Product configuration assessment loop between design and safety engineering workflows (sequence defined by numbers)

A similar integration is foreseen for other analyses and to include and connect the geometric models of the material product. The methodology applied is even similar, and resorts to several connectors already provided by vendors to include the CAD and CAE tools within the PLM, thus completing the platform which fully enables the implementation of the MBSE.

Chapter 12
Conclusion

Abstract At the end of this path a short resume is proposed, to define the lesson learnt and the next work to be done to improve the expertise within the Systems Engineering.

This handbook provides some preliminary outlines about the methodology, tools and applications of the Systems Engineering, by introducing contents and processes of the Model–Based approach (MBSE), especially when applied to the Application Lifecycle Management (ALM) and to the system design. It focuses on the material product development, as a result of an industrial processing applied to raw materials. As it was deployed along the chapters, the MBSE is implemented by means of the SysML language and of some typical tools, to allow creating some tangible objects to be shared among the system developers, refined and assessed, before that a real prototyping activity is started, with evident benefits of cost and effort reducing. The two test cases analyzed are aimed at simplifying the understanding of the Systems Engineering methodology in its main contents and showing some typical issues of the system design and integration, through some of the most popular software tools exploited in this field. The practical meaning of the full integrity of system development, based on a complete allocation of needs and requirements to the elements of the system architecture is investigated. The traceability of requirements is shown, in practice, as it is currently tracked through the MBSE in several industrial contexts. The tools integration and interoperability within the frame of an industrial platform supporting all the activities related to the Systems Engineering are even described. The tool chain used in the platform allows integrating several analyses and provides a centralized dashboard to the system developer to have a global overview on the whole development, to be then detailed in some specific analyses, performed through some dedicated software tools. Many kinds of models and simulations, having different nature, are exploited to create a heterogeneous working environment. As a main result, the MBSE allows digitalizing the product lifecycle in all of its steps and activities. This enhances the remote control and the distribution of information over the seas and among several

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8_12

operators for a real collaborative work in a real "glocal" context, as the current industrial networks of partners look.

A key issue of this handbook is even clarifying the degree of technological assessment accomplished in recent years in this specific topic, and somehow the depth of the state–of–art. Despite of several approaches proposed in the literature, which offer the best point view to develop the contents of the Systems Engineering, some common statements may be herein summarized and discussed. As it is widely remarked by several authors, the Systems Engineering allows decomposing the complexity of some new products. It is a matter of creating the convergence of several activities, being traditionally performed as separated tasks. A bridge between the detailed design, based on some quantitative analyses and supported by numerical models and data analysis, was built up to provide a model base on the qualitative processes of requirements analysis, concept design and functional modeling. This allows integrating those activities and leaving the documental approach to exploit the digital technologies. It could help in reducing the errors related to the communication among humans. Moreover, the collaboration of different technical competences and operators, used to speak different languages, was enhanced by providing a standard and intuitive language. It supports a suitable integration of the systems, particularly in some multidisciplinary applications, as the mechatronic engineering. In addition, the two activities of design and manufacturing, still quite distinguished within the daily practice of companies, were related by the Systems Engineering approach, because of the pervasive attention to the verification, validation and production issues as a direct expression of the decision making strategy applied in design. The full traceability of the whole decision making path, applied to a product development made available in digital artifacts, somehow overcomes some typical difficulties in transferring the competences between generations, since even some actions like the screening of available technologies or the product concept assessment are now part of the models shared among the operators.

Is everything done? Surely not yet, but few bottlenecks at present still inhibit a complete implementation of the Systems Engineering. A first one is related to the initial cost in terms of time and resources to equip the companies of such tools. As they affect intimately the approach currently used in the whole product development, the people are poorly prone to change their habits, although the improvement in terms of fast return of investment is quite immediate, even thanks to the reusability of models, tools and methodologies. In terms of technology, the assessment of standard approaches is still going on. Even more critical is the real interoperability between software tools. However, the innovation rate of software tools to achieve the required level of integration within the industrial platforms and between tools is continuously growing up. Moreover, in several technical domains the Systems Engineering has been already assumed as the standard approach to face the complexity.

It is really important appreciating that a suitable historical situation for this innovation already came. The transition between some assessed approaches, known in the technical and scientific literature, and the newest ones, to manage the design

and production activities, includes several tasks, as the application of new methodologies, new tools and new standards. It has to be done within a context of people trained in several techniques, previously developed, who look more confident with those than with this new approach, especially when working within the design of safety critical systems. In fact, the current cultural situation helps, since it is characterized by three simultaneous transformations as well as the wide introduction of the lean manufacturing, the fourth industrial revolution and the fast growing up of technologies related to big data.

Actually those are three different items of a coherent and unique technical evolution. The need for a lean manufacturing is pervasively moving the companies to apply a strategic approach to the industrial production based on a systematic reduction of costs and of waste, as for instance the World Class Manufacturing teaches. In this approach some typical goals and tools of the Systems Engineering are reflected. The methodology of the Systems Engineering itself enables the product developer to reduce the cost of system development, as the processes implemented allow to do. The World Class Manufacturing looks like a suitable framework to be applied in production as the architecture frameworks are in the ALM stage.

The strategic initiative of the Factory of the Future, even referred to as Smart Manufacturing or Industry 4.0, practically promotes a straight implementation of some enabling technologies which are tightly connected to the Systems Engineering, which might be properly considered as a tool to apply that approach. It can be easily demonstrated, if some of these nine main technologies are considered.

Use of collaborative robotics in manufacturing, for instance, increases the level of intrinsic complexity of systems, to be decomposed through the MBSE. Resorting to additive manufacturing requires a digital based modelling of product, compatible with the methodology and tools of the Systems Engineering. Augmented reality, which is associated to a deeper diagnosis, prognosis and monitoring of systems in service increases the number of requirements concerning function and structure of the systems and is strongly related to the allocation of requirements and prevention of failures, as the dysfunctional analysis allows to predict. The role of the heterogeneous simulation within the global item of simulation is evident, especially when the context is highly interdisciplinary. The horizontal and vertical integration, aimed at connecting the customer, the supplier and the system developer, as well as different operators within the frame of the system development, is strongly supported by the platform and the tool chain typical of the MBSE implementation. The use of the network and cloud for an extensive collaboration and remote monitoring of the system production and service finds in the Systems Engineering a straight approach for a coherent design of all the services needed and even a frame where they could be developed. The topic of cyber security surely could benefit of a combined functional and physical modeling of systems, to prevent intrusions and embezzlements.

A remark is here expressed, concerning the semantics of the proposed diagrams and, in general, about the overall methodology explained. It is important to clarify that right now neither assessed standard nor a golden rule can be applied to

implement the MBSE design processes. This approach can be slightly customized by the user, depending on the engineering domain, the context of analysis and the tools used. The SysML language is a popular tool. It is implemented through several commercial software that exhibit different characteristics. It is standardized to provide a common representation of the model-based design. Nevertheless, several interpretations have been found in literature, concerning the definition of models and diagrams. This can be understood, if one considers that designers look for a flexible tool that is able to represent straightforward what they need to design. A considerable difference between theory and practice is found, when a real implementation of the SysML to design some complex industrial systems is performed.

The Reader is kindly encouraged to evaluate the application of methods and tools here exposed, rather than of the formal correctness of diagrams, which are surely adapted, to avoid any disclosure of industrial contents, and much more to show a driven implementation in some real case studies, deemed as the most important issue, in this context, by the Authors. They wish that the introduction to the Systems Engineering applied to the industrial product development herein provided could help in discovering the features of this methodology, by identifying the tools currently developed for a straight implementation, and in realizing how they should fulfill the needs of the Reader to enhance the product innovation, in the specific technical domain covered. It should drive to focus better on some specific topic, by resorting to the specialized literature, after a preliminary understanding of the whole picture. Moreover, it should excite the Reader to get acquainted with this approach, by realizing its potential impact and increasing application to several technical fields, and mastering its main and useful contents.

Index

A

Ability, 303–305, 315, 316
Action, 11, 12, 28, 31, 48, 58, 61, 63, 71, 74, 75, 78–81, 92, 115, 127, 150, 165–167, 190, 195, 206, 209, 233–236, 239, 241, 242, 247, 250, 256, 265, 283, 290, 292, 295, 298, 300, 301, 304, 315–317, 321, 327–330, 334, 336, 338
Activity, 1, 3, 4, 8, 10–12, 17, 18, 20, 21, 25, 26, 31, 32, 34–39, 41, 43, 44, 46, 47, 49, 50, 52, 54–56, 58, 60, 63, 65, 70, 72, 73, 77, 81–87, 90–92, 103–105, 111, 115, 123, 126, 150, 151, 165, 168, 186, 190, 193, 196, 203, 205, 206, 221, 223, 225, 226, 228, 229, 233, 237, 239, 243, 252, 255, 271–273, 289–291, 293, 294, 296, 297, 299–301, 303, 305, 309, 310, 312–315, 317, 320, 328, 330, 335, 336, 339, 343
Activity diagram, 63, 150, 151, 163–165, 196, 206, 207
Actor, 28, 61, 63, 65, 116, 117, 119, 123, 128, 129, 134, 138, 150, 158, 167
Aerospace Engineering, 15, 184
Agile Systems Engineering, 18
Allocation, 5, 8, 11, 27, 40, 83, 86, 89, 97, 103, 106, 155, 166, 184, 186, 190, 191, 193–196, 199, 200, 202, 204–206, 208, 210, 211, 214, 219, 222, 226, 229, 254, 255, 258, 260, 268, 269, 286, 298, 301, 314, 329, 331, 339, 340, 343, 345
Alt, 124, 125, 134
Anti icing, 70, 72, 81, 245, 246, 248, 250, 251
Application Lifecycle Management, 5, 31, 327, 343
Approach, 3, 7, 8, 10–21, 27–31, 33, 34, 38, 42, 43, 45–50, 52, 55–57, 64, 70, 72, 74–76, 85, 87, 89, 92, 97, 103, 104, 119, 123, 126, 131, 132, 135, 139, 140, 143, 148, 149, 151, 155, 156, 158, 159, 166, 167, 184, 186–188, 190, 193, 195, 200, 204–206, 211, 214, 215, 222, 223, 225, 226, 229, 230, 235, 239, 243, 260, 268, 269, 271, 272, 274, 275, 277, 279, 283, 285, 286, 293, 296, 298–301, 307, 308, 310, 312, 314, 316, 323, 325, 328–332, 334, 336, 339, 343–346
Architecture, 1, 10, 11, 13, 20, 25, 26, 28, 29, 32, 34, 35, 37–44, 48, 49, 52, 55, 57, 59, 64, 74–76, 87, 90, 103, 105, 116, 119, 121, 126, 128, 140, 144, 147–149, 151, 152, 155–159, 179, 188–190, 193–195, 197, 199–201, 204–206, 219, 222, 227, 229, 245, 251, 254–256, 261, 269, 278, 283, 286, 300–302, 308, 311, 315, 340, 343, 345
Architecture framework, 29, 34–36, 38, 39, 45, 52
Assessment, 9, 11–13, 15, 19, 21, 33, 34, 42, 50, 51, 76, 90, 97, 131, 184, 191, 194, 228, 244, 292, 294, 298, 301, 302, 323, 339, 341, 344
Association, 19, 45, 58, 117, 123, 128, 131, 160, 313
Atomic action, 165
Attribute, 92, 106, 218
Automatic control, 14, 304
Automation Modeling Language, 56
Availability, 4, 10, 41, 80, 85, 271, 296, 304, 306, 307, 309, 315

B

Behavior diagram, 151, 171, 197, 206, 222
Bill of Materials (BOM), 45, 330

E. Brusa et al., *Systems Engineering and Its Application to Industrial Product Development*, Studies in Systems, Decision and Control 134,
https://doi.org/10.1007/978-3-319-71837-8

Black box, 115, 135, 150, 159, 196, 274, 276
Block, 28, 30, 32, 39, 40, 56, 58, 60, 63–66, 119, 140, 152–154, 156, 159–161, 166, 179, 184, 186, 194, 197, 202, 203, 210, 211, 214, 217, 218, 255–257, 260, 262, 280, 338, 340
Block Definition Diagram, 64, 116, 152–154, 159, 184, 197, 199, 321
Boundary, 47, 74, 117, 128, 214, 228, 258
Business model, 34, 47, 75, 300

C
Capability, 8, 26, 30, 35, 83, 218, 271, 296, 303, 304, 318, 323
Class, 51, 64, 83, 89, 92, 97–102, 179, 228, 244, 345
Cohesion, 296
Coiler, 79, 120, 123
Completeness, 90, 131, 138, 139, 144, 205, 219, 298–300, 314, 316
Complexity, 1, 2, 8–11, 16, 27, 49, 55, 70, 74, 121, 126, 131, 135, 138, 190, 218, 251, 294–296, 325, 344, 345
Component, 1, 3, 11, 13, 27, 28, 32, 40, 103, 106, 159, 166, 186, 194, 198, 202–204, 206, 209, 222, 232, 236, 250, 269, 277, 293, 307, 308, 310, 312, 314–316, 323
Concept of system, 13, 19, 45
Configuration, 2, 5, 10, 27, 32, 34, 40, 45, 51, 53, 54, 56, 70, 74, 79, 81, 82, 84, 91, 143, 223, 236, 239, 252, 261, 286, 290, 298, 301, 305, 316–318, 321, 327–332, 334, 337, 338
Configuration control management, 329, 330, 333
Connectivity, 32, 36, 41, 76, 87, 304
Connector, 273, 333, 335, 340
Consistency, 53, 76, 90, 104, 123, 134, 155, 157, 158, 167, 186–188, 190, 205, 219, 226, 229, 254, 257, 258, 273, 290, 294, 299, 300, 327
Constraint, 47, 66, 82, 239
Context, 14, 27, 36, 49, 57, 67, 76, 78, 84, 90, 92, 105, 115–117, 119, 120, 123, 126, 128, 130–132, 140, 144, 147, 149, 153–155, 159, 179, 233, 269, 271, 279, 292, 344–346
Co-simulation, 277–279
Coupling, 228, 230, 233, 236, 241, 296, 304, 305, 315, 317
Customer needs, 1, 8, 14, 31, 37, 44, 45, 74, 76, 77, 79, 81, 92–94, 103, 104, 121, 124, 139, 144, 293, 297, 301, 304, 329, 335
Customer satisfaction, 7, 300, 302
Cyber security, 12, 294, 345

D
Data, 2, 9, 11, 12, 19, 22, 31, 36, 37, 43, 53, 64, 67, 74, 78, 81, 87, 106, 116, 118, 127, 129, 133, 134, 138–140, 143, 144, 148, 150, 155, 157, 158, 161, 167, 169, 171, 174, 175, 179, 182, 184, 186–188, 195–197, 200, 203, 206, 210, 214, 219, 223, 226–229, 244, 246, 251, 254–258, 260, 269, 271–274, 277–279, 281, 286, 297, 298, 327, 328, 330, 331, 333–336, 340, 344, 345
De-icing, 138, 167, 179, 182, 187, 209, 245, 246, 248, 250
Dependability, 17, 41, 155, 304
Dependency, 124, 131, 154, 156, 158, 163, 169, 203
Design synthesis, 16, 29, 41, 47, 73, 238, 290, 297, 298, 314, 316, 320
Diagram, 30–33, 45, 46, 51, 52, 58–67, 74, 82, 116–126, 128, 130, 132, 133, 135, 140, 145, 148, 150–153, 156–158, 160, 163–165, 167, 168, 171, 174–179, 182, 186, 187, 190, 197, 198, 200, 203, 204, 206, 209, 219, 220, 233, 241, 255–257, 260–262, 290, 291, 316, 321, 322, 331
Digital model, 59, 239
Documentation, 3, 8, 9, 20, 299, 300, 328
Dysfunctional, 34, 43, 55, 78, 105, 139, 184, 191, 311, 312, 320, 321, 323, 324, 334, 339, 345

E
Efficiency, 2, 9, 80, 254, 258, 260, 280, 282, 304, 315, 320, 321
Engineering method, 54
Environment, 5, 12, 14, 35, 48, 51, 54, 55, 58, 59, 64, 80, 87, 90, 105, 115, 116, 118, 123, 127, 128, 130, 148, 152, 153, 196, 201, 228, 243, 245, 246, 251, 254, 255, 258, 260, 269, 272–274, 277, 279, 280, 285, 286, 293, 294, 297, 298, 318, 320, 321, 327, 334, 336, 340, 343
Evaluation, 19, 32, 41, 43, 46, 63, 225, 292, 294, 301–303, 305, 306, 314, 328
Event, 36, 37, 60, 61, 106, 134, 150, 163, 167, 281, 307, 310
Extend, 117, 118, 131

External traceability, 155

F
Feasibility, 31, 41, 52, 76, 82, 294, 317
Feedback, 14, 78, 119, 149, 280, 298
Flexibility, 87, 131, 145, 190, 239, 250, 275, 276, 320
Flow, 15, 37, 39–41, 61, 63–65, 81, 83, 84, 118, 123, 133, 134, 148, 150, 151, 161, 165, 167, 168, 186, 187, 196, 200, 206, 219, 231, 265, 279, 295, 300
Flow port, 64, 256
Function, 1, 3, 7, 8, 10, 12, 21, 26–28, 31, 32, 35, 37, 39, 40, 43, 45, 49, 57, 60, 61, 65, 73, 74, 78, 80, 82, 86, 90, 97, 105, 111, 115, 116, 119, 121, 126, 143, 144, 147–149, 154, 155, 157–161, 163, 184, 186–189, 193, 195, 199–204, 210, 211, 221, 222, 228, 231, 241, 248, 250, 252, 265, 279, 300, 304–306, 308, 311, 312, 323, 329, 332, 333, 338, 340
Functional, 2, 4, 5, 10, 16, 21, 25, 39, 40, 42, 43, 46, 49, 55, 56, 65, 73, 74, 81, 82, 85, 87, 91, 97, 103, 139, 140, 147–150, 152–159, 166, 169, 184, 186–188, 190, 191, 193–195, 199–201, 203, 205, 219, 221–223, 226, 229–231, 252, 258, 269, 272, 294, 296, 298, 301, 310, 311, 316, 320, 324, 329–331, 334, 336, 338, 340, 345
Functional analysis, 33, 34, 40, 47, 66, 86, 91, 97, 104, 118, 123, 132, 143, 145, 147–159, 163, 166, 167, 169, 171, 173, 182, 184, 186, 189–191, 193–196, 198–200, 203–206, 211, 214, 219, 222, 229, 230, 281, 311, 338, 339
Functional block, 28
Functional breakdown structure, 184, 201, 331, 340
Functional Mock-up Interface, 22, 277
Functional Mock-up Unit, 277
Functional modeling, 5, 10–12, 32, 43, 56, 193, 194, 199, 230, 255, 272, 311, 329, 331, 335, 339, 340, 344

G
Goal, 1–3, 8, 12, 14, 20, 21, 26, 43, 46, 52, 56, 57, 138, 150, 194, 219, 227, 229, 230, 244, 269, 271, 296, 305, 307, 328, 330
Granularity, 83, 92, 166, 186, 188
Guard condition, 151, 166

H
Hardware, 2, 8, 12, 37, 49, 64, 293, 294, 296, 298, 299, 303
Heterogeneous simulation, 4, 5, 35, 46, 53, 55, 56, 67, 223, 227, 269, 272, 279, 283–285, 292, 318, 333, 335, 345
Hierarchy, 4, 14, 31, 59, 64, 65, 111, 148, 152, 156, 159, 195, 196, 328, 332
Holistic, 10, 14, 16, 21, 31, 48

I
Ice protection system, 4, 72, 80, 105, 126, 244, 279
Include, 26, 34, 42, 44, 45, 54, 59, 60, 64, 78, 83, 89, 91, 92, 105, 106, 117, 118, 125, 131, 132, 144, 152, 157, 171, 176, 179, 184, 187, 195, 196, 201, 203, 205, 228, 231, 268, 279, 290, 292, 303, 304, 312, 314, 318, 321, 341
Information, 4, 9, 12, 14–17, 20, 28, 34–38, 41, 45–48, 51, 54, 56, 59, 67, 70, 79, 83, 87, 89–92, 97, 103, 105, 106, 111, 116, 123, 125–128, 133, 134, 138, 139, 143, 144, 149, 150, 155, 157, 159, 161, 167, 169, 179, 184, 194, 195, 197, 200, 205, 206, 209, 211, 227, 261, 274, 281, 312, 330, 335, 336, 343
Innovation target, 77
Input, 54, 55, 87, 106, 118, 127–129, 131, 134, 138, 144, 150, 158, 165, 167, 175, 196, 210, 233, 235, 243, 244, 246, 262, 279–282, 298, 300, 313, 314
Integration, 2, 4, 5, 10, 14, 16, 26, 27, 31, 32, 41, 42, 45, 52, 66, 73, 74, 105, 119, 126, 179, 195, 237, 238, 269, 271–274, 279, 283, 286, 296, 297, 300–302, 311, 312, 318, 327, 328, 334–341, 343–345
Interaction, 7, 26, 32, 50, 60, 61, 73, 75, 78, 80, 121, 128, 129, 131, 133–135, 138, 139, 144, 173, 201, 228, 239, 294, 297, 304, 315, 330, 336, 339
Interdisciplinary Modeling Language, 56
Interface, 36, 78, 84, 89, 106, 138, 153, 158, 179, 197, 210, 251, 272, 329, 333
Internal Block Diagram, 64, 65, 152, 153, 179, 197, 198, 200, 203
Internal traceability, 103, 104, 111

International Council on Systems Engineering, 17
Interoperability, 3–5, 12, 22, 35, 43, 46, 55, 56, 87, 223, 227, 229, 255, 256, 269, 271, 272, 274, 276, 277, 286, 306, 332, 333, 335, 343, 344
Iteration, 148, 222, 223

L

Language, 2, 3, 11, 12, 18, 22, 29, 39, 43, 46, 48, 49, 53–58, 65, 74, 106, 116, 117, 123, 128, 131, 149, 179, 194, 233, 294, 330, 335, 343, 344, 346
Laying head, 71, 79, 80, 91, 103, 119–126, 159–162, 166, 201, 203, 204, 231, 239, 243, 314, 316
Layout, 12, 27, 28, 32, 41, 43, 46, 76, 111, 200, 201, 214, 233, 236, 244, 261, 283, 296, 302, 303, 305, 311, 312, 320, 321, 327, 336, 338
Lifecycle, 2, 3, 8, 9, 20, 22, 26, 30, 33, 34, 37, 46–48, 52, 76, 82, 87, 218, 286, 289, 327, 333, 343
Lifecycle Modeling Language, 18, 56
Lifeline, 61
Limitation, 2, 30, 32, 336
Link module, 111, 190
Logical architecture, 40, 46, 57, 194, 195, 197, 201, 205, 222, 255, 269, 339
Logical block, 28, 40, 166, 258
Loop, 41, 63, 134, 139, 165, 175, 205, 222, 235, 242, 330, 341

M

Maintainability, 1, 79, 80, 85, 105, 106, 218, 219, 294, 296, 304, 307, 309, 315
Maintenance, 1, 8, 9, 19, 32, 41, 52, 76, 80, 91, 217, 218, 304, 308, 309, 315, 327
Mathematical model, 43, 239, 273, 299
Matrix, 19, 29, 36–38, 75, 103, 104, 111, 140, 142, 157, 186, 214, 219, 220, 241, 242
Mechatronic, 2, 4, 8, 70, 73, 91, 92, 121, 303, 304, 314, 344
Meta-model, 57
Method, 3–5, 11–14, 16, 18, 19, 25, 29, 42, 46–48, 52–57, 73, 149, 188, 195, 211, 227, 243, 292, 301, 307, 313, 317, 324, 339, 346
Methodology, 3, 11–13, 19, 29, 45, 47–49, 52, 66, 74, 119, 148, 155, 158, 184, 200, 230, 272, 291, 299, 330, 334, 341, 343, 345, 346
Mission, 15, 26, 39, 46, 49, 76, 105, 115, 123, 127, 133, 134, 147, 159, 201, 279–285, 301, 304, 306–308, 323, 325
Mock-up, 44, 45, 299, 330, 333
Model-Based Systems Engineering, vii–ix, 11
Model exchange, 271, 275, 277, 279
Models, 2–5, 8–12, 16, 28–30, 37, 39, 43–45, 48, 53, 55, 56, 66, 72, 73, 81, 90, 115, 119, 130, 148, 159, 193, 200, 225–230, 238, 239, 251, 254, 258, 267, 269, 272, 273, 277, 279, 280, 282, 283, 286, 290–295, 297, 298, 301, 310, 312, 318, 325, 330, 331, 333, 335, 336, 338–341, 343, 344, 346

N

Need, 3, 8, 13, 14, 18, 19, 26–29, 31, 32, 41–43, 49, 55, 76–80, 82, 83, 92, 97, 105, 111, 115, 116, 121, 123, 143, 147, 148, 161, 187, 202, 219, 227–230, 232, 233, 236, 251, 256, 271, 273, 274, 304, 327, 328, 330, 331, 337, 345, 346
Node, 36, 159, 165, 167

O

Object, 2, 7, 11, 16, 28, 29, 34, 47, 48, 51, 52, 54, 56–58, 73, 89, 97, 106, 111, 121, 128, 130, 134, 135, 139, 140, 144, 145, 189, 227, 228, 268, 275, 276, 285, 286, 294, 297, 298, 308, 318, 328, 338, 343
Object heading, 106
Object text, 106
Ontology, 3, 90
Open system, 14
Operation, 2, 3, 7–9, 12–14, 21, 26, 27, 29, 31, 32, 34, 41, 50, 53, 57, 60, 61, 63–65, 70, 71, 74, 78–81, 85, 105, 111, 115, 118, 119, 121, 122, 127, 131–135, 138–145, 147–153, 155, 157, 158, 161, 163, 165, 166, 169, 171, 184, 188, 195, 196, 204–206, 209–211, 214, 222, 225, 228, 238, 239, 301, 302, 305–308, 315, 317, 318, 321, 339
Operational, 5, 10, 16, 36, 37, 49, 72, 73, 82, 84, 85, 87, 91, 97, 103, 105, 115, 116, 118–121, 126, 128–133, 138–141, 143–145, 148, 149, 152, 155, 156, 158–160, 169, 171, 186, 188, 190, 194–196, 201, 203, 217, 222, 226, 246, 247, 269, 272, 293, 294, 299–301, 303, 305, 306, 309, 314, 315
Operator, 4, 106, 120, 122–125, 130, 134, 159–161, 294, 336

Opt, 134
Output, 55, 65, 87, 118, 134, 138, 144, 157, 165, 166, 176, 179, 196, 210, 257, 259, 280–282, 290, 291, 293, 310

P
Package diagram, 64–66
Parallel, 34, 43, 53, 63, 133, 134, 151, 167, 175, 219, 278, 299, 308, 340
Parametric diagram, 66
Part, 8, 10, 20, 27, 28, 32, 33, 44, 45, 61, 65, 80, 81, 93, 94, 98–103, 106–110, 112, 131, 135–137, 148, 152, 154, 160, 174, 179, 184, 194, 199, 232, 238, 241, 248, 292, 293, 306, 307, 314, 320, 321, 338, 344
Physical architecture, 40, 46, 166, 194, 297, 298, 339
Physical modeling, 5, 11, 32, 33, 39, 42, 43, 55, 223, 255, 256, 269, 272, 299, 330, 345
Platform, 10, 12, 22, 50, 52, 53, 87, 106, 120, 123, 126, 239, 314, 318, 327, 328, 330–336, 338, 339, 341, 343, 345
Port, 36
Process, 2, 3, 8, 10, 11, 14, 16, 17, 20, 21, 26, 28–35, 37, 39, 41–43, 46–56, 59, 65, 72–74, 78, 79, 82, 86, 87, 90, 91, 103, 111, 119–125, 131, 133–135, 138–140, 142–144, 147, 148, 151, 155, 157, 158, 160–162, 167, 184, 186, 190, 191, 193–195, 199, 200, 204–206, 209, 210, 214, 216, 219, 221–223, 225–227, 229, 230, 244, 250, 254, 256, 258, 269, 271, 273–278, 286, 290, 292–301, 305, 310–314, 316, 318–320, 327–332, 335, 337–339
Product, 1–4, 7–15, 17, 19, 21, 26–32, 34, 35, 37, 41–43, 46–48, 50, 51, 54–56, 69, 73–77, 79–83, 85, 86, 91, 111, 119, 128, 148, 155, 156, 166, 194, 195, 206, 231, 278, 286, 289, 290, 292–294, 300, 301, 303, 304, 307, 308, 311, 312, 328–330, 333, 337, 339, 340, 343–346
Product Breakdown Structure, 40, 166, 194, 201, 202, 298, 311, 312, 328
Product Data Management, viii, 327
Product Lifecycle Development, 8, 17, 32, 41, 45, 53, 105, 230, 239
Product Lifecycle Management, 5, 32, 291, 293, 327
Project management, 10, 13, 20, 51
Project planning, 51, 82
Pseudostate, 163
Purpose, 12, 14, 26, 47, 57, 58, 82, 86, 92, 131, 157, 167, 193, 227, 228, 292

Q
Quality, 1, 2, 8, 10, 41, 49, 51, 54, 56, 76, 82, 85, 86, 90, 184, 189, 221, 274, 286, 290, 292, 296, 298, 300, 303, 304, 307, 330

R
Ref, 16, 38, 52, 80, 133, 186, 206, 217, 218, 255, 277, 281, 286, 293, 296, 345
Reference Technology Platform, 52
Refine, 19, 27, 117, 123, 124, 131, 144
Relationship, 117, 123, 124, 131, 144, 148, 153, 160, 163, 184
Reliability, 1, 10, 34, 41, 43, 53, 74, 79, 80, 85, 87, 218, 219, 222, 239, 245, 250, 251, 273, 275, 304, 307, 308, 310–313, 315, 320, 321, 324, 339, 340
Requirement, 1, 2, 5, 8, 10, 11, 14, 15, 17, 20, 21, 26–28, 31–34, 37–39, 43, 44, 46–49, 51, 54, 58, 59, 66, 70, 72–74, 77, 79, 81–87, 89–92, 97, 103–106, 111, 115, 119, 121, 123, 126, 133, 138–141, 143, 144, 148, 155, 156, 158, 163, 184, 186–191, 193, 199, 200, 203, 204, 206, 219, 226, 229, 231, 233, 239, 252, 254, 258–260, 268, 269, 285, 286, 289, 292–295, 297, 298, 300, 301, 304–306, 310, 311, 313, 314, 320, 328, 330, 332, 333, 335–340, 343, 345
Requirements diagram, 59, 60, 124, 140, 156, 186, 204
Reusability, 3, 8, 46, 294, 296, 344
Risk analysis and management, 13
Robustness, 87

S
Safety, 1, 4, 10, 21, 34, 43, 74, 78, 82, 85, 92, 103, 105, 106, 184, 298, 304, 305, 307, 309, 311, 315, 320, 339
Safety engineering, 19, 184, 243, 307, 332, 339–341
Satisfaction, 7, 188, 199, 204, 205, 219, 220, 222, 229, 258, 269, 298–300
Scenario, 26, 46, 79, 105, 119, 124, 126, 127, 132–138, 143–145, 147, 150, 152, 153, 163, 167, 190, 195, 228, 277–280, 282, 285, 286, 320
Security, 9, 41–43, 53, 76, 85, 87, 303, 307, 315, 328
Sequence, 1, 8, 11, 27, 29, 36, 37, 52, 60, 63, 97, 118, 121, 123, 126, 131, 132, 134, 135, 138, 139, 144, 147–149, 167, 187,

196, 206, 208, 230, 248, 249, 252, 259, 269, 281–283, 285, 301, 310, 313, 320, 327, 337
Sequence diagram, 61, 62, 116, 118, 119, 123–126, 132, 157, 163, 197
Simplicity, 1, 87, 131, 138, 147, 251, 278, 296, 303, 305, 314, 315
Simulation, 5, 10, 15, 38, 39, 44, 53, 55, 56, 59, 149, 150, 152, 153, 157, 167, 173, 179, 182, 184, 193–195, 200, 216, 223, 225–227, 229, 230, 236–238, 243–246, 252, 256, 257, 260, 265, 267, 269, 276–280, 282, 283, 285, 286, 290, 293, 294, 297, 298, 306, 312, 313, 317, 321–325, 330, 336, 338, 345
Smartness, 4, 8, 41, 70, 89, 92, 96, 304, 315
Software, 2–4, 12, 13, 15, 18–22, 26, 28, 34, 37, 42, 43, 47–50, 56, 57, 64, 73, 74, 78, 86, 90, 119, 130, 140, 143, 205, 223, 225, 227, 229, 239, 244, 259, 269, 271, 272, 274, 278, 279, 286, 293–297, 302, 303, 327, 328, 330, 332–338, 343, 344, 346
Solution, 18, 27, 32, 34, 40–42, 47, 50, 51, 70, 73, 83, 134, 149, 158, 193–195, 198–200, 205, 206, 208, 209, 214, 216, 217, 219, 222, 226, 232, 233, 235, 241, 245, 249, 256, 259, 273, 275, 284, 301, 314, 330, 336, 337
Solution-oriented, 21, 26
Stability, 159, 162, 201, 228, 233, 242, 273, 274, 296, 305, 316
Stakeholder, 51, 75, 78, 80, 83, 126, 147–149, 293
Standard, 2, 9, 12, 13, 15, 16, 18–22, 25, 26, 29, 32, 34, 35, 37, 39, 42, 43, 45, 49–52, 56–58, 74, 77, 85, 90, 105, 106, 147, 148, 184, 223, 227, 233, 256, 271, 273, 274, 276–279, 286, 292, 294, 295, 298, 299, 303, 306, 312, 314, 333, 335, 339, 344, 345
Standardization, 3, 12, 13, 19, 20, 49, 50, 52, 56, 226, 292
State, 3, 15, 21, 25, 27, 29, 35–38, 48, 55, 60, 61, 73, 76, 77, 85, 86, 116, 117, 123, 131, 141, 143, 147, 149–152, 157–159, 163, 164, 166, 174–179, 182, 186, 197, 230, 289, 293, 300, 307, 310, 311, 314, 318, 321, 328, 338
State machine diagram, 60, 150–152, 163, 165, 171, 197, 316, 321, 322
Strategy, 32, 40, 49, 53, 54, 87, 165, 167, 174, 179, 190, 195, 239, 248, 272, 274, 275, 277, 279, 285, 286, 301, 316, 320, 328, 331, 332, 334, 336, 344
Structure diagram, 65, 190
Sub-state, 61, 151, 163, 174
Survivability, 303–305, 315, 316
Sustainability, 17, 41, 76, 80
Syntax, 83, 85, 86, 103, 116
System, 1–5, 7–22, 25–29, 31, 32, 34–53, 55–58, 60, 63–65, 69, 70, 72, 75, 76, 78–81, 83, 84, 86, 87, 90–92, 97, 103, 105, 106, 111, 115, 116, 118–129, 132, 135, 138, 140, 141, 143, 144, 148–151, 155, 157–161, 163, 165–167, 174, 184, 186–188, 190, 191, 193–196, 199–204, 208, 209, 211, 214, 216, 218, 219, 222, 225, 228–232, 235–239, 241–248, 250–252, 255, 256, 260, 262, 268, 269, 279, 283, 285, 286, 289–318, 320–322, 324, 327, 328, 330–332, 336–340, 343–345
System behavior, 5, 12, 14, 27, 28, 32, 44, 48, 59, 60, 67, 80, 129, 132, 144, 148–151, 158, 163, 164, 166–168, 171, 184, 187, 190, 195, 197, 200, 206, 214, 228, 230, 279, 284, 291, 298, 309, 317, 321
System integration, 14, 27, 45, 73, 301, 311, 312, 328, 336, 339
System Modeling Language, 12, 18, 56, 58
Systems Engineering, 2–4, 7, 11, 13, 15–18, 20, 21, 27, 31–34, 47, 49–52, 56, 58, 81, 83, 85, 91, 103, 226, 227, 337, 343–346

T

Task, 9, 11, 12, 29, 39, 43, 46, 50, 53, 75, 83, 149, 150, 155, 187, 199, 204, 226, 227, 292, 300, 301, 306, 315, 316, 320, 329, 339
Technical requirement, 1, 15
Technical standard, 37
Technology, 1, 2, 8, 14, 17, 19, 27, 32, 37, 40–43, 48, 73, 81, 126, 209, 214, 233, 287, 305, 344
Test, 3, 9, 14, 31, 45, 55, 56, 59, 69, 70, 78, 87, 119, 122, 140, 145, 150, 158, 163, 190, 226, 229, 250, 269, 290, 293, 294, 297–299, 301, 303, 305, 306, 312–314, 316–318, 335, 343
Testability, 105, 296, 303, 315
Test case, 4, 18, 70, 73, 79, 86, 91, 93–102, 104–110, 112, 113, 120–123, 126, 129–131, 133, 134, 140–142, 144, 159, 163, 184, 231–233, 238–240, 243, 244,

256, 260, 272, 306, 314, 318–321, 335–337, 339, 340
Testing, 1, 18, 30–32, 34, 73, 167, 190, 225, 230, 292, 294, 301, 303, 329
Theory of systems, 13, 14
Tool, 12, 18, 22, 38, 42, 47, 49, 52–54, 56, 57, 59, 74, 87, 89, 90, 92, 103, 104, 106, 130, 139, 140, 150, 157, 195, 223, 227, 229, 243, 255, 256, 259, 268, 271, 273, 274, 276–279, 297, 318, 327, 328, 330, 332–340, 343, 345, 346
Trace, 36, 37, 87, 111, 131, 139, 140, 155, 156, 188, 189, 331
Traceability, 3, 5, 8, 10, 11, 17, 27, 28, 37, 74, 87, 111, 119, 123, 128, 130, 133, 139, 141, 144, 145, 155–159, 169, 186, 190, 191, 195, 199, 203–205, 219, 221–223, 226, 229, 230, 258, 268, 269, 271, 285, 286, 294, 298, 301, 314, 329, 332, 333, 343, 344
Trade-off, 82, 128, 194, 205, 222
Transition, 37, 52, 60, 151, 163, 165, 179, 200, 294, 312, 344
Transportability, 303, 306, 315
Trigger, 119, 128, 129, 133, 150, 163, 174, 280, 281

U

Unified Architecture Framework, 35, 38
Unified Modeling Language, 3, 12, 18, 56
Use case, 27, 32, 39, 49, 60, 61, 115–119, 121, 123–134, 138–145, 150, 153, 157–162, 164–175, 177, 179, 181, 183, 187, 195, 197, 206–209, 281, 307
Use case diagram, 60, 61, 116, 117, 121, 122, 128, 160, 331
User, 8, 9, 12, 15, 19, 38, 48, 49, 57, 81, 87, 89, 90, 103, 106, 111, 115, 123, 138, 157, 158, 163, 179, 188, 210, 252, 271, 272, 286, 300, 301, 304, 306, 333, 335, 336, 346

V

Validation, 1, 5, 8, 14, 19, 32–34, 41–46, 48, 49, 52, 73, 76, 82, 91, 145, 167, 228, 230, 258, 275, 277, 283, 290, 291, 293–295, 297–303, 305, 306, 313, 314, 317, 318, 328–330, 338, 344
V-diagram, 140
Verification, 1, 5, 19, 32, 41–46, 48, 52–54, 56, 58, 73, 82, 86, 131, 134, 140, 145, 149, 157, 158, 188, 199, 228–230, 237, 252, 254, 255, 258, 269, 275, 290–293, 295, 297–303, 313, 314, 316, 317, 328, 329, 338, 344
View, 10, 13–16, 19, 21, 28, 29, 35, 37, 39, 45–47, 50, 52, 58, 65, 103, 106, 111, 113, 116, 119, 130, 132, 138, 140, 145, 149–151, 157, 158, 184, 186, 189, 190, 196, 200, 204, 214, 220, 232, 252, 256, 262, 269, 283, 286, 292, 299–301, 303, 331, 333, 337
Viewpoint, 36, 37
Virtual platform, 10

W

Waterfall diagram, 30
White box, 119, 150, 195, 196, 200, 205, 206