DELIVERABLE D 1.3

PROCESS DEFINITION A4F FOR APPLICATION DEFINITION

Author(s): DLR
WorkPackage N°: WP1
Due date of deliverable: 31.05.2020
Actual submission date: 13.04.2021
Document ID: D1.3 - Process definition A4F for use case definition - v08_submitted.docx

Grant Agreement number: 815122
Project acronym: AGILE 4.0
Project title: Towards cyber-physical collaborative aircraft development
Start date of the project: 01/09/2019
Duration: 36 months

Project coordinator name & organisation:
Pier Davide Ciampa, DLR – System Architectures in Aeronautics | Aircraft Design and System Integration
Tel: +49 40 2489641-322
E-mail: pier-davide.ciampa@dlr.de

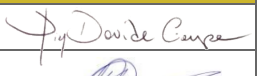
Project website address: www.agile4.eu

THIS PROJECT HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S HORIZON 2020 RESEARCH AND INNOVATION FRAMEWORK PROGRAMME UNDER GRANT AGREEMENT NO 815122

DOCUMENT INFORMATION

Document ID	D1.3 - Process definition A4F for use case definition.docx		
Version	1.0		
Version Date	13.04.2021		
Author	Luca Boggero (DLR)		
Dissemination level	PU		

APPROVALS

	Name	Company	Date	Visa
Coordinator	Pier Davide Ciampa	DLR	13.04.2021	
WP Leader	Jos Vankan	NLR	18.09.2020	
Reviewer	Susan Liscouët-Hanke	CONU	28.08.2020	
Reviewer	Vincent Saluzzi	BOM	09.09.2020	

DOCUMENTS HISTORY

Version	Date	Modification	Authors
0.1	21.04.2020	Document initialization	Luca Boggero (DLR)
0.2	28.04.2020	Added Chapter “Nomenclature”	Luca Boggero (DLR)
0.3	29.06.2020	Added Chapters “Process”, “Application” and “Conclusion”	Luca Boggero (DLR)
0.4	02.07.2020	Initialized Chapter “Viewpoints”	Luca Boggero (DLR)
0.5	14.07.2020	Revised Chapters “Ontology” and “Viewpoints”, added Chapter “MBSE Framework”, general revision	Luca Boggero (DLR)
0.6	21.07.2020	Version submitted to Concordia University for review	Luca Boggero (DLR)
0.7	18.09.2020	Version submitted to DLR for final review	Luca Boggero (DLR)
0.8	08.04.2021	Added section of AGILE 4.0 Requirement Ontology; changed “Use Case” with “Application Case”	Luca Boggero (DLR)

LIST OF AUTHORS

Full Name	Organisation
Luca Boggero, Pier Davide Ciampa	DLR

DISTRIBUTION LIST

Full Name	Organisation
AGILE 4.0 Consortium	-
Externals	-

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

GLOSSARY

Acronym	Signification
A4F	AGILE 4.0 Architectural Framework
ConOps	Concept of Operations
HTP	Horizontal Tail Plane
INCOSE	International Council on Systems Engineering
MBSE	Model Based Systems Engineering
MDAO	Multidisciplinary Design Analysis and Optimization
OEM	Original Equipment Manufacturer
SysML	System Modeling Language
WP	Work Package

1 EXECUTIVE SUMMARY

1.1 Introduction

The present deliverable is in the form of a model, and it addresses the first stages of a Systems Engineering Product Development process, which aims at designing and optimizing a system starting from the identification of stakeholders and needs and including other activities as requirements engineering. More specifically, D1.3 focuses on the Application Case Definition, which encompasses the identification of stakeholders involved during the whole life cycle of the system, collection of their needs and validation through high-level scenarios and transformation of needs into requirements. The deliverable presents the guideline and the process defined in AGILE 4.0 project for the Application Case Definition. The deliverable D1.3 is in the form of a model publicly available, through the AGILE 4.0 Cloud[1] and the AGILE 4.0 Zenodo page[2]. The present supporting report describes the developed model.

1.2 Brief description of the work performed and results achieved

The research activities that have brought to the final version of the present deliverable started from a literature survey on the concepts of *Stakeholder*, *Needs*, *Scenarios* and *Requirements*. Additional investigation has been the carried out addressing Systems Engineering processes and Architectural Framework.

A first version of process and guideline for Application Case Definition developed in the project has been presented to the whole Consortium at M05 during a physical workshop. The process and guideline have been applied by the partners for the development of the seven Application Cases, and feedback has been collected. The process and guideline have been revised and at then modeled and described in the report. Additional virtual workshops have been organized and have contributed to refine and improve the work, bringing eventually to the results collected in the current version of the deliverable.

1.3 Deviation from the original objectives

1.3.1 Description of the deviation

The activities in preparation of the present deliverable were originally planned to start in December, 2019. The actual start took place in preparation of the M05 meeting at Lausanne early February, 2020. Therefore, the activities were slightly delayed. In addition, due to the COVID-19 crisis, planned physical meetings had to be turned into web meetings, negatively impacting the progress.

1.3.2 Corrective actions

Apart from turning physical meetings into web meetings and shifting the deadline of actions and the deliverable D1.3 forward in time, no corrective actions were required.

[1] https://www.agile4.eu/cloud/index.php/s/pD2qtG9TyJX4ie3
[2] https://zenodo.org/record/4671896

2 INTRODUCTION

2.1 Aim of the deliverable and organization of the report

The deliverable D1.3 is presented in the form of a model, which represents the process and guideline for the definition of Application Cases. Within the context of the AGILE 4.0 project, *Application Case Definition* is meant as identification of four elements that have a certain relationship with the system (e.g. an aircraft, a component, a part) being developed. These four elements are: *stakeholder*, *need*, *scenario* and *requirement*. Aim of this deliverable is to provide the definition of these four elements, to expresses their mutual relationships and with the system, and to propose a process for their development.

The Application Case Definition represents the first part of the Model Based Systems Engineering (MBSE) process being set up within the AGILE 4.0 project for the development of complex systems of interest (e.g. aircraft). The MBSE process also includes system architecting and Multidisciplinary Design Analysis and Optimization (MDAO) of systems. This process is schematized in Fig. 1 (based on Ref. [1]).

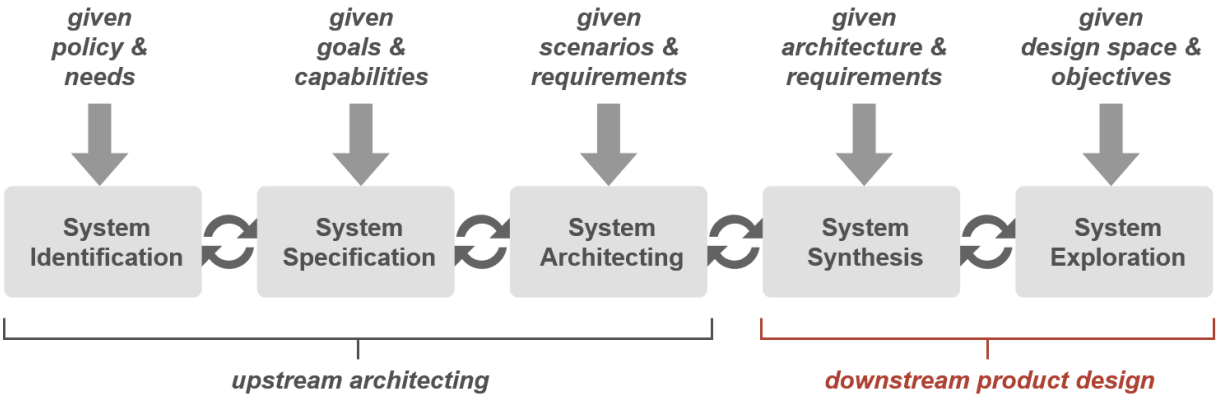


Fig. 1: Schema of the AGILE4.0 Development process: a bridge between upstream architecting phase (e.g. a MBSE approach) and the downstream product design phase (e.g. a MDAO process) [1].

This process starts with the identification phase of system stakeholders and collection of their needs. Then, in the specification phase the Concepts of Operations (ConOps) are elaborated to describe through scenarios how the system will operate during its life cycle and therefore to refine and validate stakeholder needs. Validated needs are afterwards transformed into requirements, which drive the system architecting and its development. In the architecting phase, multiple alternative solutions are generated. The architecture definition includes the functional description (all the functions to be satisfied), the logical descriptions (mapping functions to logical components fulfilling the functions), and the instantiation of physical architectures (allocation of physical components to the logical ones). For every selected architecture, and the given set of requirements, the integration and the synthesis of the system is performed. Such activity delivers the sizing and the verification of the design parameters of the physical architecture of the system under development. For a given physical architecture and defined design space the system is explored, analyzed, and\or optimized through MDAO processes. This deliverable focuses on the first two steps of the proposed MBSE process: *Stakeholders & Needs (System Identification)* and *ConOps & Requirements (System Specification)*.

The present report describes the model of the process for definition of Application Cases. It is organized as follows. Chapter 2 introduces the topic of the deliverable, and highlights the relations between D1.3 and the other research activities of the project. Chapter 3 presents the MBSE Architectural Framework being developed in AGILE 4.0, providing the definitions of *Ontology*, *Viewpoint* and *Process*. The *Ontology* that includes stakeholders, needs, scenarios and requirements is described in Chapter 4. Chapter 5 collects and describes the *Viewpoints* identified for the Application Case Definition. The *Process* for the definition of the four elements is instead covered in Chapter 6. Chapter 7 presents an example of application of the proposed process based on one of the Application Cases of the AGILE 4.0 project. The Application Case definition ontology described in this deliverable is made open access, as explained in Chapter 8. Finally, the report presents conclusions and further remarks in Chapter 9.

The model of deliverable D1.3 is made publicly available, and it can be downloaded from the AGILE 4.0 Cloud[3] and the AGILE 4.0 Zenodo page[4].

2.2 Deliverable in the context of AGILE 4.0 and organization of the research tasks

Fig. 2 shows the main relations between the present deliverable and the other activities of the AGILE 4.0 project.

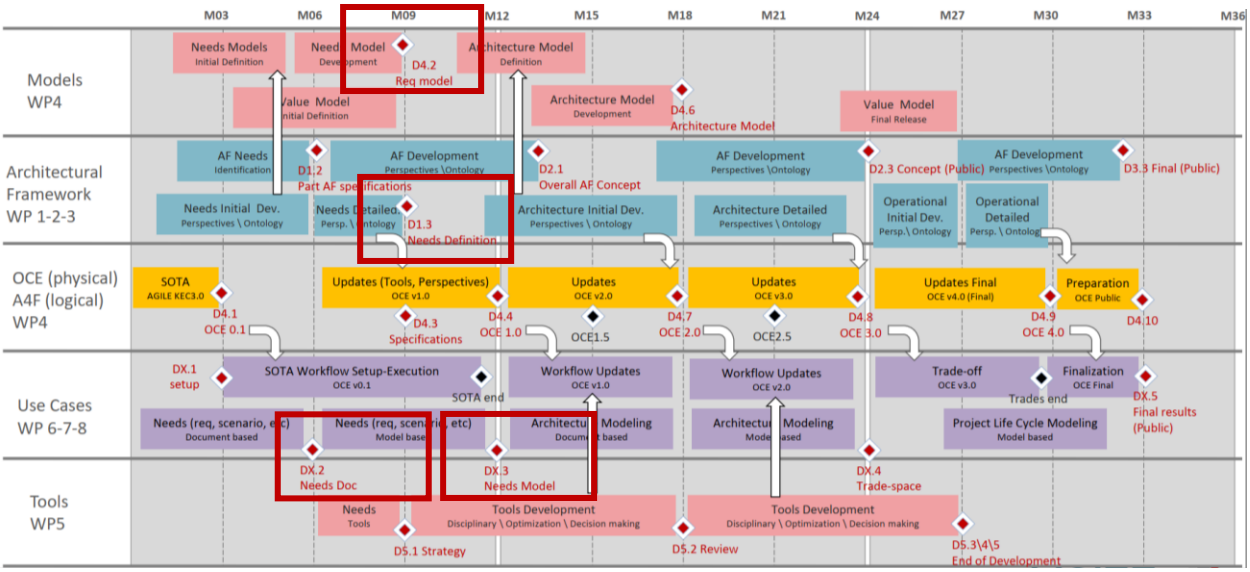


Fig. 2: Main deliverables and tasks of AGILE 4.0 project.

The research activity carried on for deliverable D1.3 began from a literature review to initially define the concepts of stakeholder, need, scenario and requirement, to clarify their relationships and to derive a process for their identification and development. The activities of deliverable D1.3 were indeed driven by some needs identified in deliverable D1.2 [2], more specifically stating the lack in the project of a guideline for the definition of product requirements. First results of this literature review have been presented to the whole Consortium in M05, during a physical workshop. The workshop aimed at applying the proposed process for the definition of the stakeholders, needs, scenarios and requirements of the seven Application Cases of the AGILE 4.0 project. Two were the outcomes derived from the workshop and from further refinements. First, these activities were used in preparation of deliverables D6.2 [3], D7.2 [4] and D8.2 [5], which cover the definition of the seven Application Cases by adopting a document based approach. Secondly and at the same time (around M06), feedback was collected from Work Packages WP 6, 7 and 8. This feedback was furtherly elaborated and a new guideline has been defined for D1.3 and used in preparation of deliverable D4.2 [6] of WP4, which is about a model-based approach for definition of Application Cases. In other words, D4.2 presents the types of model that should be used in AGILE 4.0 for the representation of stakeholders, needs, scenarios and requirements. These types of models proposed in deliverable D4.2, together with the updated Application Case Definition process described in the present report, will be used in WP 6, 7 and 8 for the preparation of deliverables D6.3 [7], D7.3 [8] and D8.3 [9], which again will address the definition of the seven Application Cases, but this time through a model based approach.

[3] https://www.agile4.eu/cloud/index.php/s/pD2qtG9TyJX4ie3
[4] https://zenodo.org/record/4671896

3 AGILE 4.0 MBSE ARCHITECTURAL FRAMEWORK

One of the goals of the AGILE 4.0 project is the definition of a new MBSE Architectural Framework. The ISO/IEC/IEEE 42010 standard defines an architectural framework as a set of:

conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders [10]

In other words, an architectural framework is a guideline that can be used to represent system architectures, where an architecture is a formal description of a system, of its behavior and of the relationships among all the entities composing the system.

The MBSE Architectural Framework developed in AGILE 4.0 is structured in four layers, as shown in Fig. 3.

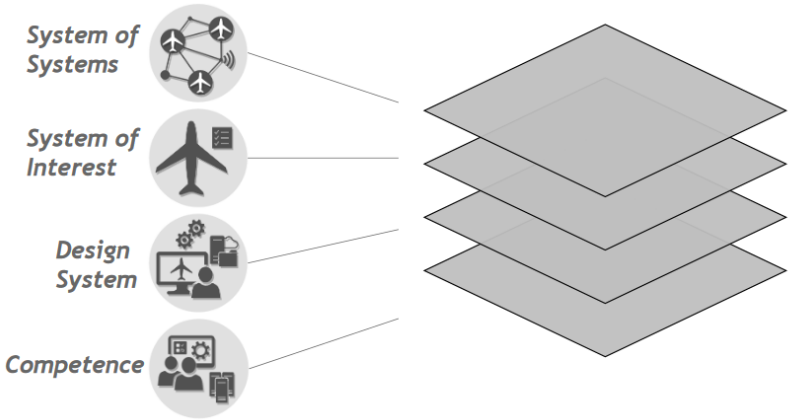


Fig. 3: Four-layer structure of the MBSE Architectural Framework addressed in AGILE 4.0 project.

One layer focuses on the *System of Interest*, which is the system being designed and optimized. The System of Interest can be for example an aircraft, a subsystem (e.g. ATA 24 – Electrical System) or a component (e.g. an electric generator), but also an enabling system, like a Supply Chain or a production system. The MBSE approach schematized in Fig. 1 is employed for the development of the System of Interest. In AGILE 4.0 project, seven Systems Of Interest are developed in WP6-7-8.

The System of Interest can be developed through a *Design System*, which is represented by a different layer. This layer encompasses all the Systems Engineering and MDAO processes required to develop the System of Interest. Again, an MBSE approach can be employed for the definition, architecting and development of this layer.

The development of a System of Interest needs several competences, which can be for example formalized through disciplinary modules. Therefore, the lower layer *Competences* is part of the MBSE Architectural Framework.

As mentioned before, the ultimate aim of the MBSE Architectural Framework is the development of Systems of Interest. However, the same framework can be employed for the development of multiple systems operating together according to the concept of *System of Systems*, which defines the fourth layer, the highest one.

The present deliverable focuses on the *Design System* layer, although the proposed guideline is used to represent the architectures of a *System of Interest*.

Several architectural frameworks are available in literature (e.g. Zachman's Framework [11], DoDAF [12], MODAF [13], NAF [14] and TOGAF [15]). They are all characterized by the following elements:

- Ontology: definition of the concepts composing the architecture and their relationships.
- Viewpoint: convention for the construction, interpretation and use of architectures from the perspective of specific system concerns.
- Process: logical sequence of tasks performed to achieve a particular objective [16].

As explained in Section 2.1, the focus of this deliverable is on the first two steps of the Systems Engineering approach being adopted in the AGILE 4.0 project, namely *Stakeholders & Needs* and *ConOps & Requirements*. Therefore, the ontology presented in this deliverable is about the following concepts: *stakeholder*, *need*, *scenario* and *requirement*. Analogously, the viewpoints presented are focusing on the *Stakeholders* and *Requirements* perspectives. Finally, the process starts with the identification of stakeholders, collection of their needs, analysis of scenarios for the validation of needs and development and management of requirements.

It should be finally noted that an MBSE architectural framework is employed in the frame of the present deliverable to address the definition of Systems Of Interest. However, an MBSE architectural framework can be similarly adopted to address all the layers depicted in Fig. 3, therefore targeting also the development of the Design System.

3.1 Model of the MBSE Architectural Framework: A4F

One of the main innovations brought through the AGILE 4.0 project is the realization of the MBSE Architectural Framework in the form of model itself. This model of Architectural Framework is named AGILE 4.0 Architectural Framework, or A4F. The System Modelling Language (SysML) [17] is adopted as standard modelling language since it is recommended by the International Council on Systems Engineering (INCOSE) for MBSE activities [18] and it fits the purpose in the project. A Microsoft Visio file depicting the SysML diagrams for the representation of the A4F for the Application Case Definition is publicly available in: https://www.agile4.eu/cloud/index.php/s/pD2qtG9TyJX4ie3.

The SysML profile is here extended to model all the elements and attributes presented in [19]. A new profile named *AGILE4Profile* is defined, as represented by the SysML Package Diagram of Fig. 4.

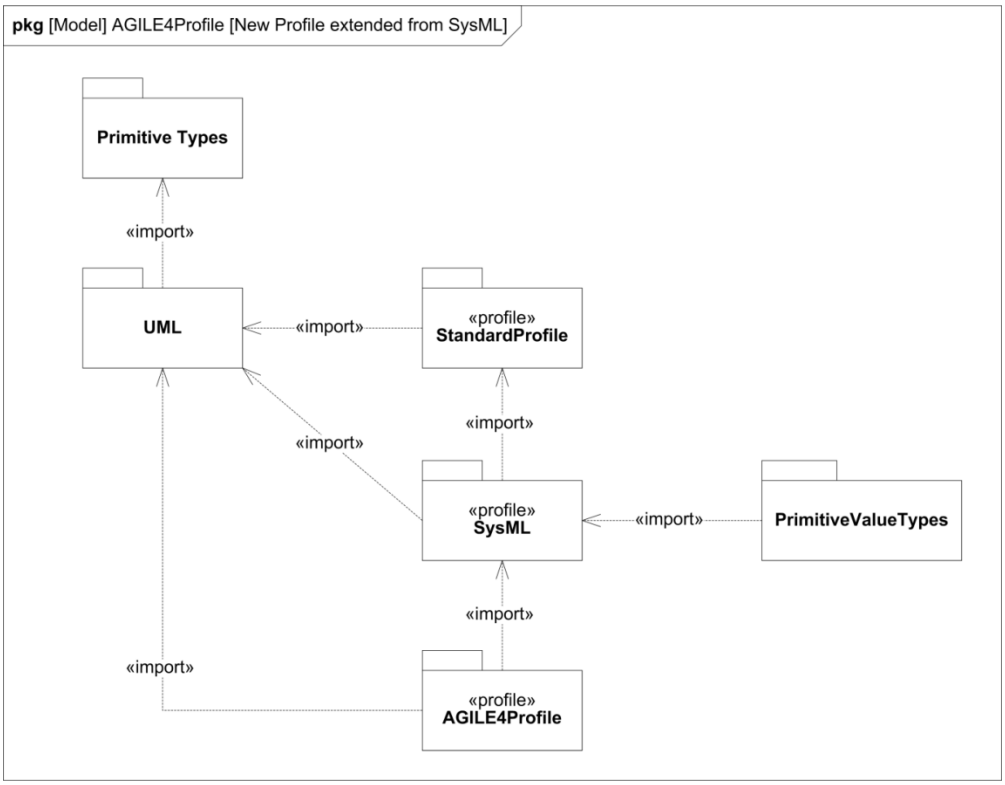


Fig. 4: Extension of the SysML profile: AGILE4Profile.

The new stereotypes introduced with the *AGILE4Profile are collected in the SysML Package Diagram* of Fig. 5.

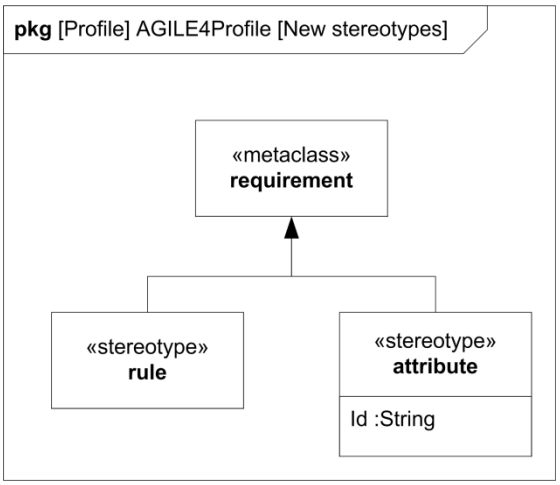


Fig. 5: New stereotypes of the AGILE4Profile.

The general metaclass <<*requirement*>> is extended to generate the new stereotype <<*rule*>>, to represent each specific rule the requirements should comply with, as described in Section 4.4. The stereotype <<*attribute*>> is generated from the metaclass <<*requirement*>>, too. This new element will be described in Section 4.5.

The models of ontology, viewpoints and process for the Application Case Definition are described in the following Chapters.

4 ONTOLOGY

The definitions of stakeholder, need, scenario and requirement are given in the present Chapter. These definitions are collected from the literature, and relevant references are collected at the end of the report. Section 4.2 illustrates the relationships among the four concepts, while more information about requirements is collected in Sections 4.3, 4.4 and 4.5.

4.1 Definitions of stakeholder, need, scenario and requirement

Stakeholder

- "*group or individual that is affected by or has a stake in the product or project*" [20]
- "*individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations*" [21]

Examples of stakeholders encompass: airlines, passengers, maintainers, Original Equipment Manufacturers (OEMs), supplier, regulation authorities, ground crews, cabin crews, air traffic controllers.

Need

- "*informal expression of something that has to be provided, ensured or avoided by a system or the development project of this system, from the viewpoint of one or several stakeholders*" [22]

In other words, needs represent wishes, necessities and desires that are expected from the system by the different stakeholders. Needs are generally unstructured and expressed in fuzzy or general, ambiguous terms.

Examples of needs are: "maximization of profit" (stakeholder: airline); "safe and comfortable flight" (stakeholder: passenger); "good accessibility to equipment to be maintained" (stakeholder: maintainer).

Scenario

- "*Operational scenarios are a step-by-step description of how the system should operate and interact with its users and its external interfaces (e.g., other systems).* […]. *Operational scenarios should be described for all operational modes, mission phases (e.g., installation, startup, typical examples of normal and contingency operations, shutdown, and maintenance), and critical sequences of activities for all classes of users identified. Each scenario should include events, actions, stimuli, information, and interactions as appropriate to provide a comprehensive understanding of the operational aspects of the system*" [20]
- "*Description of an imagined sequence of events that includes the interaction of the product or service with its environment and users, as well as interaction among its product or service components*" [22]
- "*Scenarios are […] useful for the acquirer and the supplier to verify that the system design will satisfy the stakeholders' needs and expectations*" [22]

Summarizing and combining the two provided definitions, scenarios are step-by-step descriptions of how the system should operate and interact with its users and external interfaces. Scenarios represent both nominal situations - as aircraft operations (e.g. maintenance, flight) and other life-cycle stages (e.g. production, certification) – and off-nominal situations, e.g. emergencies. High-level scenarios (i.e. focusing on the whole System of Interest and not on its lower-level components) can be used for the validation of the stakeholder needs, as it will be presented in Section 4.2. Scenarios include different types of activities, some of them are performed by the system, others are inputs and outputs exchanged between system and users. An example of scenario can be found in Chapter 7.

Requirement

- *"a statement which translates or expresses a need and its associated constraints and conditions"* [22]

Contrary to stakeholder needs that are ambiguous and unstructured, requirements have to follow precise structures and rules, as it will be described in Sections 4.3, 4.4 and 4.5. Requirements should indeed assure characteristics as unambiguity, completeness, feasibility, verifiability and correctness.

Chapter 7 will provide some examples of requirements.

4.2 Relationships between stakeholder, need, scenario and requirement

A model representing the relationships among the four elements previously defined – namely stakeholder, need, scenario and requirement is developed and depicted in Fig. 6. It should be noted the "Requirements" which should be more properly defined as "Requirement expressions", since they are not only composed the text but also additional attributes required for their management.

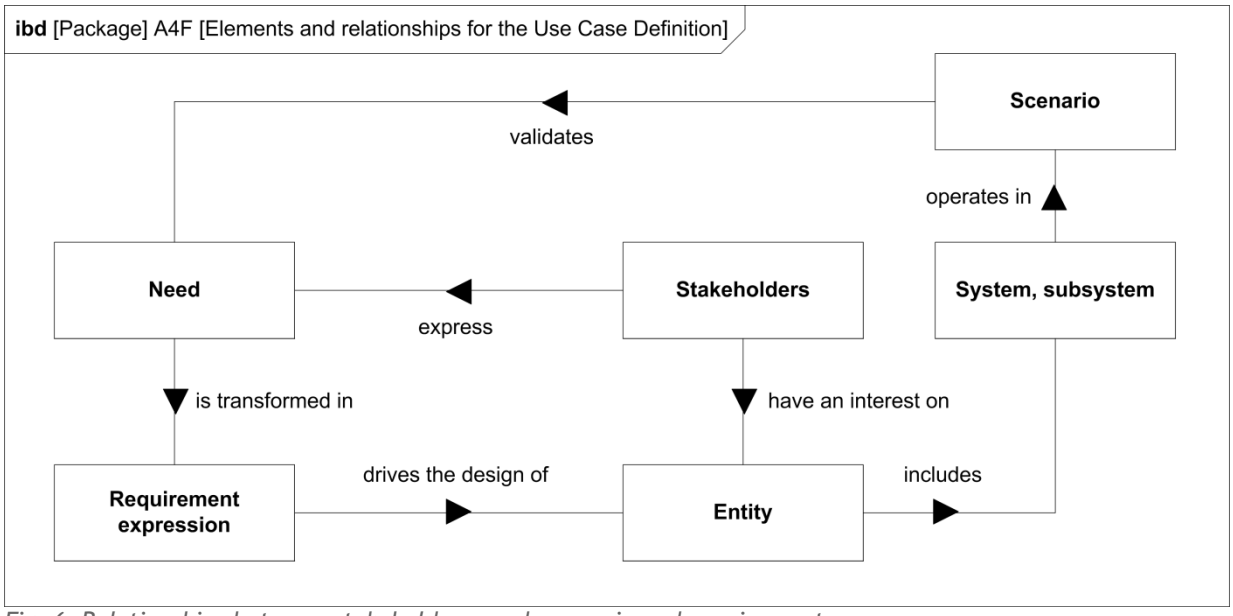


Fig. 6: Relationships between stakeholder, need, scenario and requirement.

As explained before, stakeholders are individuals, groups of people or organizations who have an interest on a product or a service, which are referred in the present model as an entity. Since the development process being set up in the AGILE 4.0 project focuses on aeronautical products – e.g. aircraft of lower level components – the "entity" of the present model includes systems and subsystems, which are operated in different scenarios during the various stages of their life-cycles.

All the stakeholders expect different needs from the entity. A developed product can be successful only if the needs of the stakeholders are reflected in the solution. In other words, the product is validated if it is compliant with all the needs. Please note that the term "validation" has a different meaning from the term "verification". As it will in more details described later, a product is verified when it is compliant with all its requirements. But a verified product might not be how demanded by the stakeholders, and therefore it could be not validated. The difference between the words "verification" and "validation" can be also clearly underlined by the questions ([23]) "*Are we building the system right?*" (verification – i.e. in the correct way?) and "*Are we building the right system?*" (validation – i.e. the product expected by the stakeholders?). High-level scenarios are therefore helpful means to check whether the suppliers (i.e. who designs and produces the System of Interest) fully understand and agree on what the stakeholders expect from the entity. However, it should be noted that not all stakeholder needs can be validated through Scenarios. An example of validation through the analysis of scenarios will be provided in Chapter 7.

As written before, stakeholder needs are generally expressed in a very informal way, without following structures or rules, and often including ambiguous terms. Therefore, all the needs should be transformed into requirements, in order to assure characteristics as unambiguity, completeness, feasibility, verifiability and correctness. Otherwise, the failure of a project or product may be very likely. Therefore, guidelines for the correct development and organization of requirements are introduced by INCOSE and slightly adapted by the project Consortium. These guidelines include for instance rules and structures for the correct writing of requirements and attributes for the management of the requirements during the whole system development phase. The view in Fig. 7 depicts the elements composing these guidelines and their relationships.

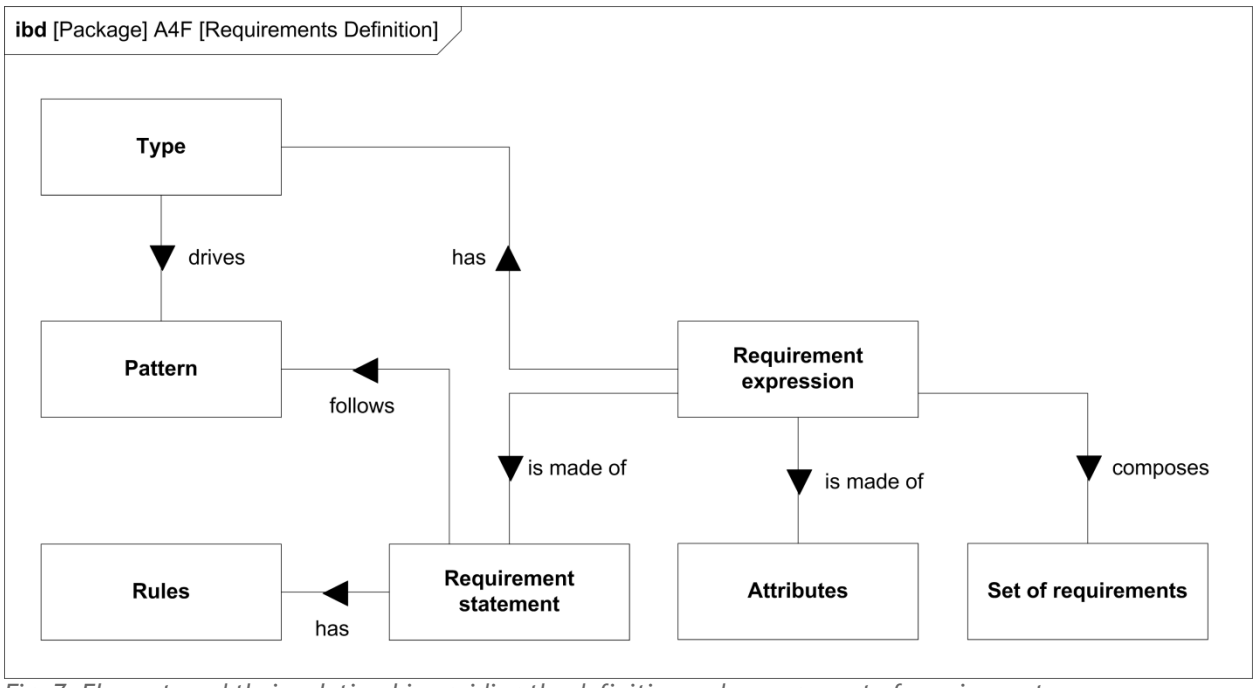


Fig. 7: Elements and their relationships guiding the definition and management of requirements.

Requirements are collected in groups called "Requirement sets" and drive the design of the product. As mentioned before, requirement expressions are composed by two elements. The first one is the text – or statement – while the latter is a set of attributes that are needed for the management of the requirement expression. Section 4.5 will list and explain all the attributes considered in the AGILE 4.0 project.

Each statement must follow a predefine structure, or pattern, which depends on the type of requirement. The pattern specifies which elements (e.g. subjects, functions, parameters) must be or could be included in the requirement statement. More details are provided in Section 4.3.

Five requirement types are considered in AGILE 4.0, although different types are also proposed in literature:

1. **Functional requirements**: define what functions need to be performed to accomplish the objectives [20]

   e.g. "The aircraft shall accelerate the pre-flight operations

2. **Performance requirements**: define how well the system needs to perform the functions [20]

   e.g. "The aircraft shall fly at Mach 0.8 during cruise"

3. **Design constraint requirements**: limit the options open to a designer of a solution by imposing immovable boundaries and limits [24]

   e.g. "The avionic system shall be supplied with standard voltage of 28V"

4. **Environmental requirements**: define which characteristics the system should exhibit when exposed in specific environments (e.g. acoustic/thermal loads, atmospheric conditions) [24]

   e.g. "The aircraft shall be maneuverable in case of icing conditions"

5. **Suitability requirements**: include a number of the "-ilities" in requirements to include, e.g. transportability, survivability, flexibility, portability, reusability, reliability, maintainability, and security [24]

   e.g. "The aircraft shall have a failure rate < 50 failures/1000 FH during the entire aircraft life (reliability)"

Finally, requirement statements have to follow all the rules described in Section 4.4, in order to be unambiguous, complete, feasible, verifiable and correct.

4.3 Requirement patterns

A requirement pattern is a set of mandatory and optional elements included in the requirement text that assure the correct syntax of the requirement statement. Different patterns identified according to the type of the requirements. In other words, the text of a performance requirement must include some element that might not be required in the text of a functional requirement.

The specification ISO/IEC/IEEE 29148:2011 [22] demands that a requirement text – although expressed in natural language – follows a predefine syntax including a subject, a verb and a complement. However, the most interesting proposal about requirement patterns is provided by Carson in [24]. Within the context of the AGILE 4.0 project, new patterns are derived from Carson's proposal.These patterns together with examples are presented below. Elements in square brackets are optional.

- **Functional requirement**

The SYSTEM shall [exhibit] FUNCTION [while in CONDITION]

e.g. "The aircraft shall provide propulsive power [during the entire mission]"

- **Performance requirement**

The SYSTEM shall FUNCTION with PERFORMANCE [and TIMING upon EVENT TRIGGER] while in CONDITION

e.g. "The aircraft shall fly at min Mach 0.8 during cruise"

- **Design constraint requirement**

The SYSTEM shall [exhibit] DESIGN CONSTRAINTS [in accordance with PERFORMANCE while in CONDITION]

e.g. "The aircraft shall have technologies with maturity TRL 9"

- **Environmental requirement**

The SYSTEM shall [exhibit] CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION]

e.g. "The aircraft shall be maneuverable during exposure to ice conditions [for the entire flight]"

- **Suitability requirement**

The SYSTEM shall exhibit CHARACTERISTIC with PERFORMANCE while CONDITION [for CONDITION DURATION]

e.g. "The aircraft shall exhibit a steady gradient of climb of minimum 2.4% while condition of one-engine-inoperative"

The five kinds of pattern have been represented in the delivered model. As an example, the diagram depicted in Fig. 8 shows the pattern of a performance requirement. The requirement statement is composed by different elements, some of them mandatory (i.e. system, function, performance and condition), other optional (i.e. timing and event trigger). The multiplicity number shown closed to the element blocks define whether each element is mandatory (multiplicity = 1) or optional (multiplicity = 0,1).

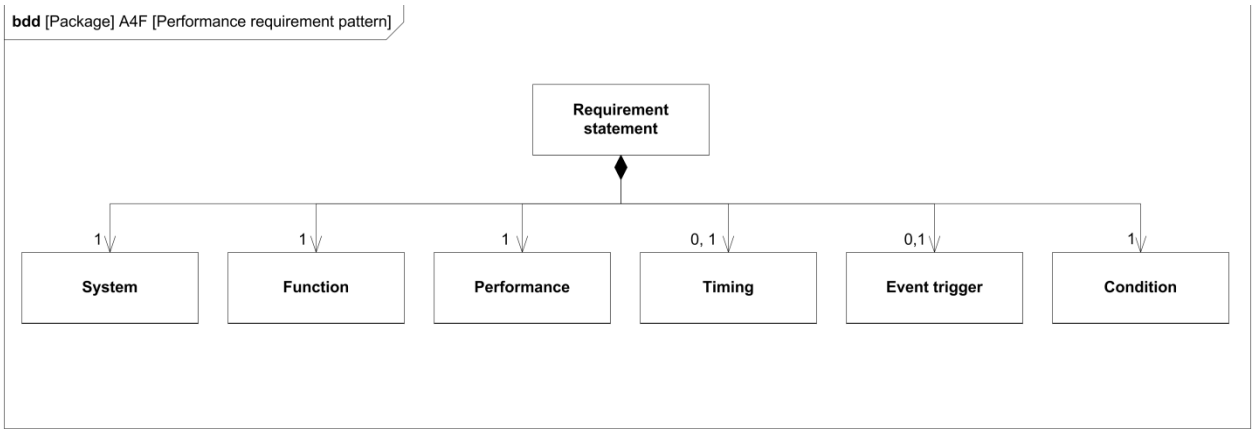


Fig. 8: Pattern of the performance requirement.

4.4 Requirement rules

INCOSE states that requirements statements and requirement must assure the following characteristics: necessity, appropriateness, unambiguity, completeness, singularity, feasibility, verifiability, correctness, conformity, consistency, comprehensibility and ability to be validated. For more details, please refer to [25].

With the aim of deriving requirements compliant with all these characteristics, INCOSE proposes 44 rules that should be followed during the writing of the requirement statements. A subset of 17 rules is selected in AGILE 4.0. The 17 rules are part of the model of the present deliverable, as represented in Fig. 9.

The explanation of each rule together with some examples is extracted from [25] and reported in appendix.

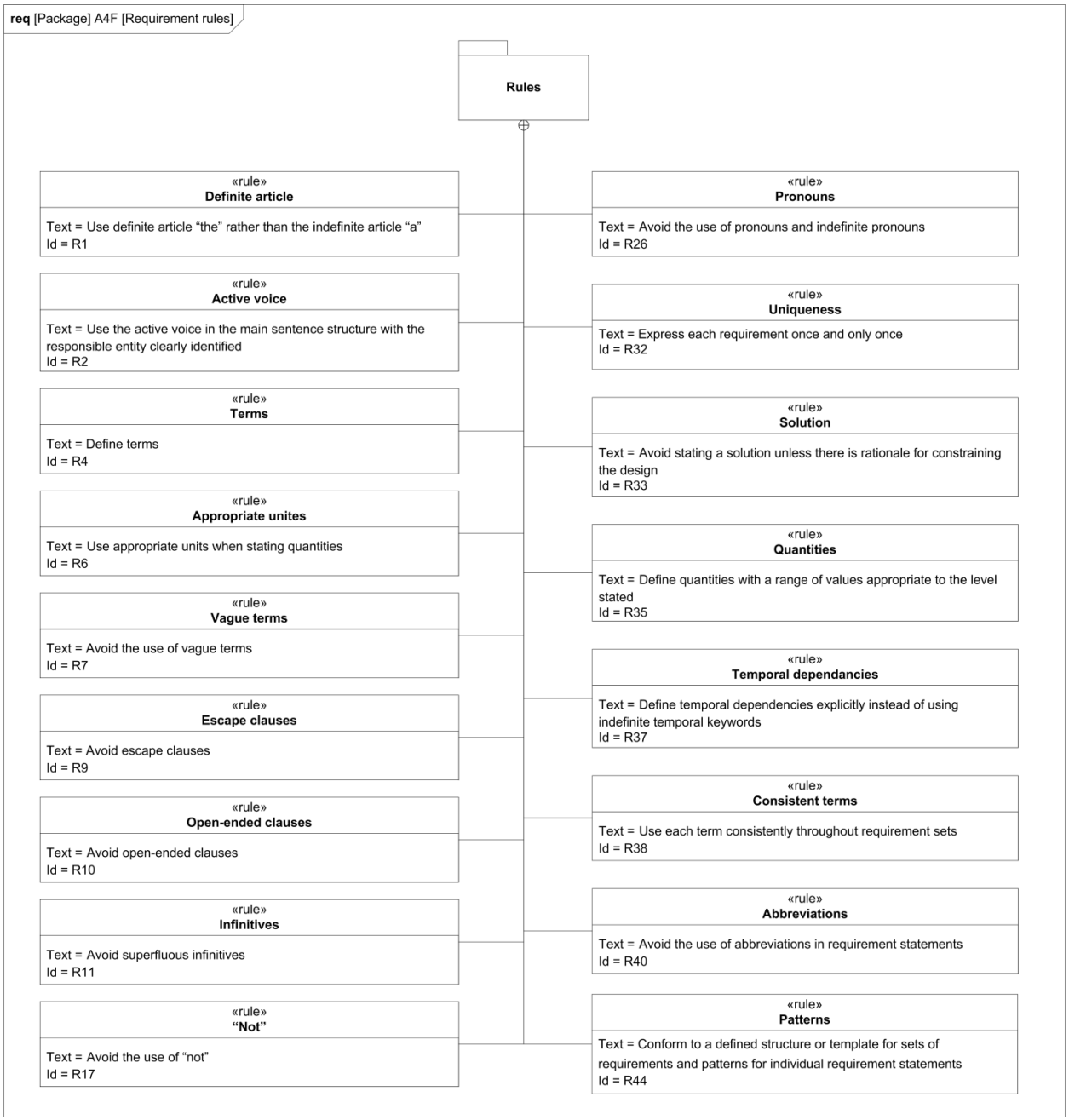


Fig. 9: Subset of INCOSE’s rules for the writing of requirements.

4.5 Requirement attributes

Other than rules for the writing of requirement statements, INCOSE suggests a set of attributes for the management of the requirement expressions. During the research activities of the project, additional attributes have been identified. The diagram in Fig. 10 represents part of proposed model, showing all the identified attributes. Their description is provided in the remaining part of the Section.

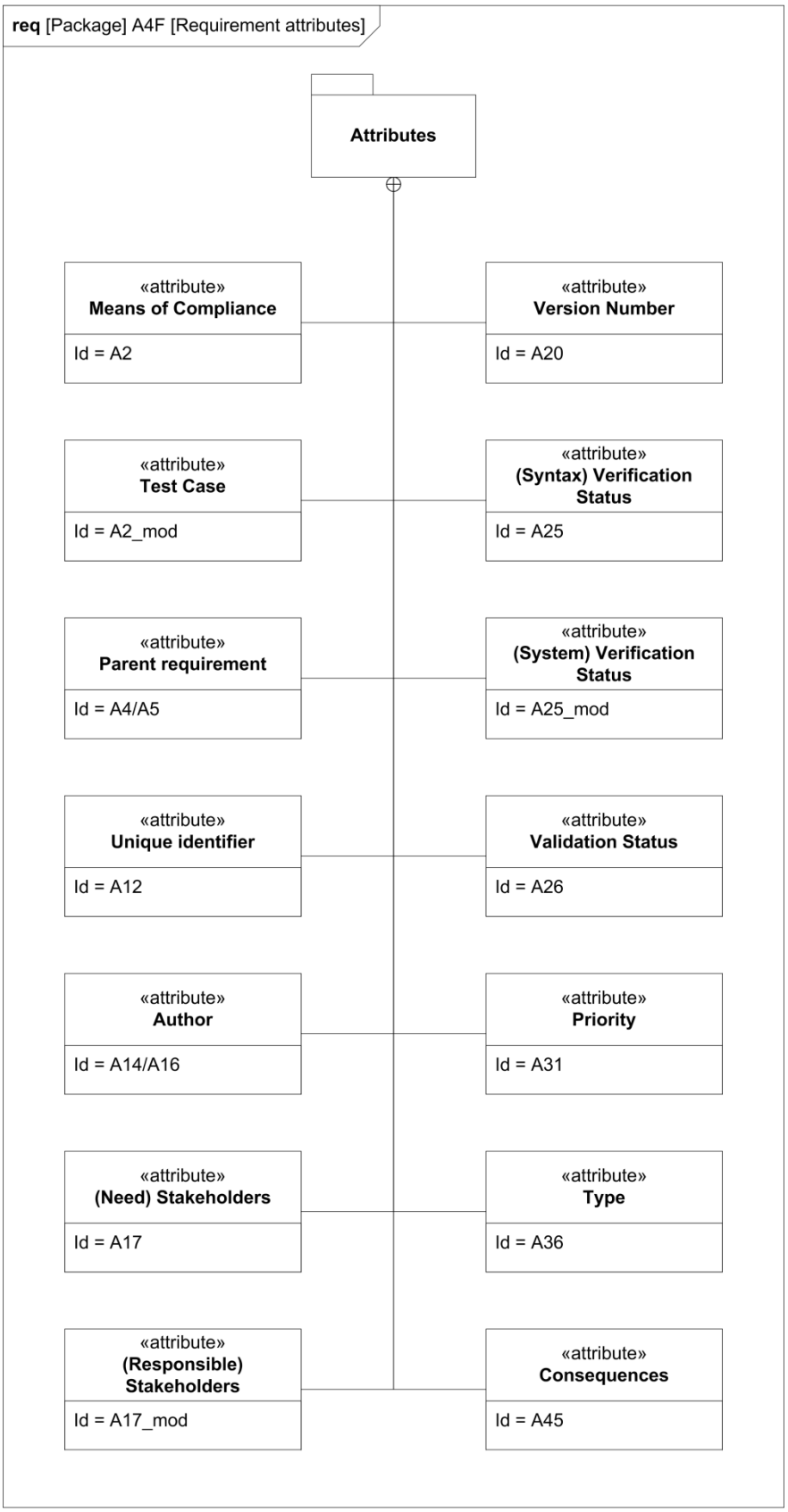


Fig. 10: Attributes for the management of requirement expressions.

A2 - System Of Interest (SOI) primary verification method (Means of Compliance)

“Means of Compliance” is an agreement between the “Needs Stakeholder” and the “Responsible Stakeholder” about how compliance will be demonstrated for a requirement. A means of compliance states for example the methods that shall be used to proof compliance. Finally a means of compliance will result in a conclusive verdict based on the output of a “Test Case”.

A2_mod – Test Case

A “Test Case” describes a demonstration of compliance to a requirement. A “Test Case” can be for example a physical test, a numeric simulation using *x* design competence tools, an empirical analysis or expert judgement. It is possible that a test case is used to demonstrate compliance to multiple requirements.

This attribute is not proposed by INCOSE in [25], but it has been added by the Project Consortium.

A4/A5 - Trace to parent requirements (or source)

Parent requirements at one level are implemented at the next level of the system architecture via allocation. A child requirement is one that has been derived or decomposed from the allocated parent; the achievement of each of the children requirements will lead to the achievement of the parent requirement. Each of the children requirements must be able to be traced to its parent requirement.

Some requirements can be traced to their source (such as document, trade study, workshop, or interview). The source could be a stakeholder need or some form of model.

A12 - Unique identifier

A unique identifier can be either a number or mixture of characters and numbers used to refer to the specific requirement.

A14/A16 - Originator/Author (/Owner)

The Originator/Author is the person submitting the requirement. The Owner is the person or element of the organization that maintains the requirement, who has the right to say something about this requirement, approves changes to the requirement, and reports the status of the requirement.

A17 – (Need) stakeholders

List of key stakeholders that have a stake in the implementation of the requirement and who will be involved in the review and approval of the requirement as well as any proposed changes.

A17_mod – (Responsible) stakeholders

The attribute “Responsible Stakeholder” indicates who is responsible for complying with a requirement by means of a test case.

This attribute is not proposed by INCOSE in [25], but it has been added by the Project Consortium.

A20 - Version number

An indication of the version of the requirement. This is to make sure that the correct version of the requirement is being implemented as well as to provide an indication of the volatility of the requirement.

A25 – (Syntax) verification status

An indication that the requirement has been verified. Requirement verification is the process of ensuring the requirement meets the rules and characteristics (necessary, singular, conforming, appropriate, correct,

unambiguous, complete, feasible, and verifiable) for a well-formed requirement as defined in this Guide or a comparable guide or checklist developed by the organization. Possible values could include: true/false, yes/no, or not started, in work, complete.

A25_mod – (System) verification status

The attribute "System Verification" ensures that the designed/produced system is compliant with the requirement. In other words, it ensures that the requirement has been met during the development of the system.

This attribute is not proposed by INCOSE in [25], but it has been added by the Project Consortium.

A26 - Requirement validation status

An indication that the requirement has been validated. Requirement validation is the confirmation the requirement is an agreed-to transformation that clearly communicates the needs of the stakeholder expectations in a language understood by the developers. Requirement validation activities need to involve the customers and users of the system being developed. Possible values could include: true/false, yes/no, not started, in work, complete.

A31 – Priority

This is how important the requirement is to the stakeholders. Priority may be characterized in terms of a number (1,2,3) or label (high, medium, low). Assuming each child requirement is necessary and sufficient, their priority may be inherited from their parent requirement. High priority requirements must always be met for the project to be successful; lower priority requirements may be traded off when conflicts occur or when there are budget or schedule issues resulting in a de-scope effort[5].

A36 - Type/Category

Each organization will define types or categories to which a requirement fits, based on how they may wish to organize their requirements. The Type/Category field is most useful because it allows the requirements database to be viewed by a large number of designers and stakeholders for a wide range of uses.

A45 - Consequences

When a requirement is not met there are consequences. For example looking at manufacturability requirements, if they are not met it can mean scrap or extra cost because of the need for concession to be written or the need for a repair. In real life aircraft design not all requirements can be met. Therefore to make an informed decision on which requirements to bed or soften information is needed on the consequences of not meeting a requirement. This could be quantifiable feedback, for example extra cost or lead time. Or it could be qualitative feedback.

This attribute is not proposed by INCOSE in [25], but it has been added by the Project Consortium.

[5] It should be noted from this definition given by INCOSE that the priority attributed is assigned by stakeholders to their requirements (through needs). Therefore, an objective priority attribute considering all the requirements with the same yardstick independently from stakeholders might be needed. In this regards, the reader can refer to the method proposed in [25].

5 VIEWPOINTS

The viewpoints identified in AGILE 4.0 for the representation of the Application Case Definition are grouped in two categories to address two specific concerns: Stakeholders Viewpoints and Requirements Viewpoints.

5.1 Stakeholders Viewpoints

The Stakeholders Viewpoints represent Stakeholders involved during the life cycle of the system, their Needs and Scenarios in which the system is operated during its life cycle.

Stakeholders hierarchy Viewpoint

AIM
This Viewpoint is used to list all the system stakeholders. Stakeholders can be represented in a hierarchy
CONCERNS ADDRESSED
Stakeholders involved during the life cycle of the system
REPRESENTATION
SysML Block Definition Diagram (see D4.2 [6])

Needs Viewpoint

AIM
This Viewpoint is used to represent the needs and which stakeholder these needs are derived from. This Viewpoint should report per each need the following properties: - Need text - Need ID - Associated stakeholder
CONCERNS ADDRESSED
Needs and their ownership to stakeholders
REPRESENTATION
SysML Requirement Diagram (see D4.2 [6])

Scenario Viewpoint

AIM
This Viewpoint is used to represent the high-level scenarios in which the system is operated during its life cycle. Scenarios are used to validate the needs. This Viewpoint should report the following information: - System of Interest - Stakeholders interacting with the system - Activities performed by the system and interactions with stakeholders
CONCERNS ADDRESSED
Scenarios for needs validation
REPRESENTATION
SysML Sequence Diagram (see D4.2 [6])

5.2 Requirements Viewpoints

The Requirements Viewpoints represent system Requirements.

Requirement sets Viewpoint

AIM
This Viewpoint is used to represent the different requirement sets that group all the requirements.
CONCERNS ADDRESSED
Requirement sets
REPRESENTATION
SysML Package Diagram (see D4.2 [6])

Requirement list Viewpoint

AIM
This Viewpoint is used to list all the requirements belonging to each requirement set. This Viewpoint should report per each requirement the following properties: - Requirement text - Requirement ID - Requirement type - Requirement author - Requirement version
CONCERNS ADDRESSED
List of requirement belonging to a requirement set
REPRESENTATION
SysML Requirement Diagram (see D4.2 [6])

Requirement pattern Viewpoint

AIM
This Viewpoint is used to list all the textual elements composing a requirement statement. This Viewpoint should represent the following textual elements: - System [mandatory for all types of requirement] - Function [mandatory for functional and performance requirements] - Condition [optional for functional and design requirements; mandatory for performance and suitability requirements] - Condition duration [optional for suitability requirements] - Performance [optional for design requirements; mandatory for performance and suitability requirements] - Timing [optional for performance requirements] - Event trigger [optional for performance requirements] - Design constraints [mandatory for design requirements] - Characteristic [mandatory for environmental requirements] - Environment [mandatory for environmental and suitability requirements] - Exposure duration [optional for environmental requirements] This Viewpoint should report per each requirement the following properties: - Requirement text - Requirement ID - Status of the syntax verification
CONCERNS ADDRESSED
Pattern of requirements

REPRESENTATION
SysML Requirement Diagram (see D4.2 [6])

Glossary Viewpoint

AIM
This Viewpoint is used to collect and define acronyms and specific terms and concepts for which a definition is required
CONCERNS ADDRESSED
Definitions of used acronyms, terms and concepts
REPRESENTATION
SysML Requirement Diagram (see D4.2 [6])

Requirements verification Viewpoint

AIM
This Viewpoint is used for the verification of each requirement. This Viewpoint should report per each requirement the following properties: - Requirement text - Status of system verification - Status of validation - Means of Compliance - Responsible stakeholder The Viewpoint should also represent the test case selected for the verification of each requirement. This Viewpoint should report the following properties of the test case: - Test case ID - Reference to the view describing the test case
CONCERNS ADDRESSED
Verification of requirements
REPRESENTATION
SysML Requirement Diagram (see D4.2 [6])

Means Of Compliance Viewpoint

AIM
This Viewpoint is used to collect all the available test cases grouped per Means Of Compliance
CONCERNS ADDRESSED
Collection of test cases grouped per Means Of Compliance
REPRESENTATION
SysML Package Diagram (see D4.2 [6])

Test Case Viewpoint

AIM
This Viewpoint is used to describe each test case selected to verify a requirement. This description should include: - Specific element (e.g. a disciplinary tool) employed to verify the requirement - Need stakeholder from which the requirement has been derived - Responsible stakeholder who performs the test - Actions and interactions done during the test case
CONCERNS ADDRESSED

Description of the test case for requirement verification
REPRESENTATION
SysML Sequence Diagram (see D4.2 [6])

Traceability Viewpoint

AIM
This Viewpoint is used to represent the “derivation” relationship between needs, requirements and consequences. The represented requirements should include the following properties: - Requirement text - Requirement ID - Responsible stakeholder - Requirement priority
CONCERNS ADDRESSED
Traceability between needs, requirements and consequences
REPRESENTATION
SysML Requirement Diagram (see D4.2 [6])

Fig. 11represents the SysML Package Diagram showing all the viewpoints just described.

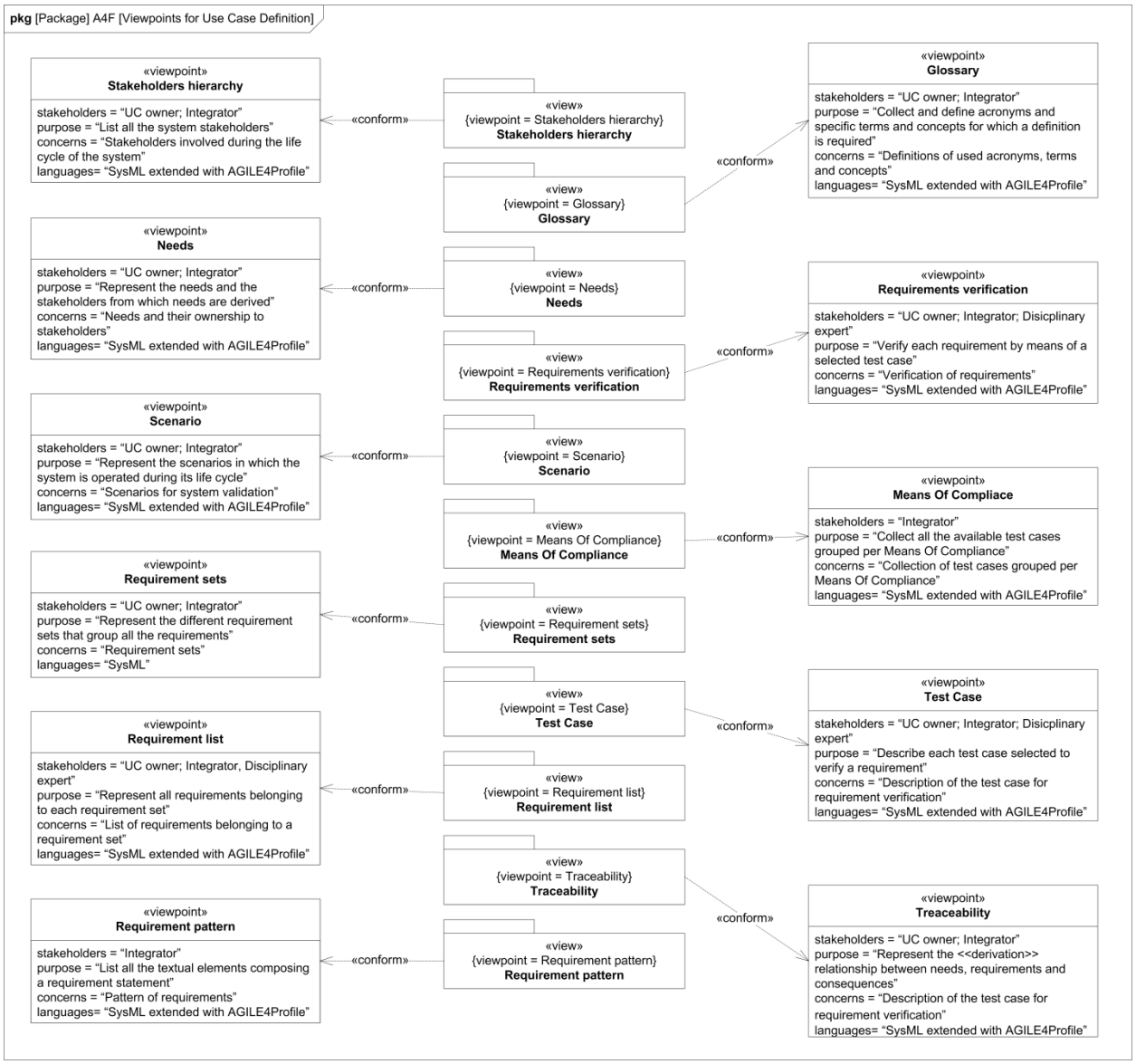


Fig. 11: List of viewpoints for Application Case Definition.

6 PROCESS

Fig. 12 represents the process set up in WP1 for the definition of the Application Case in terms of stakeholders, needs, scenarios and requirements.

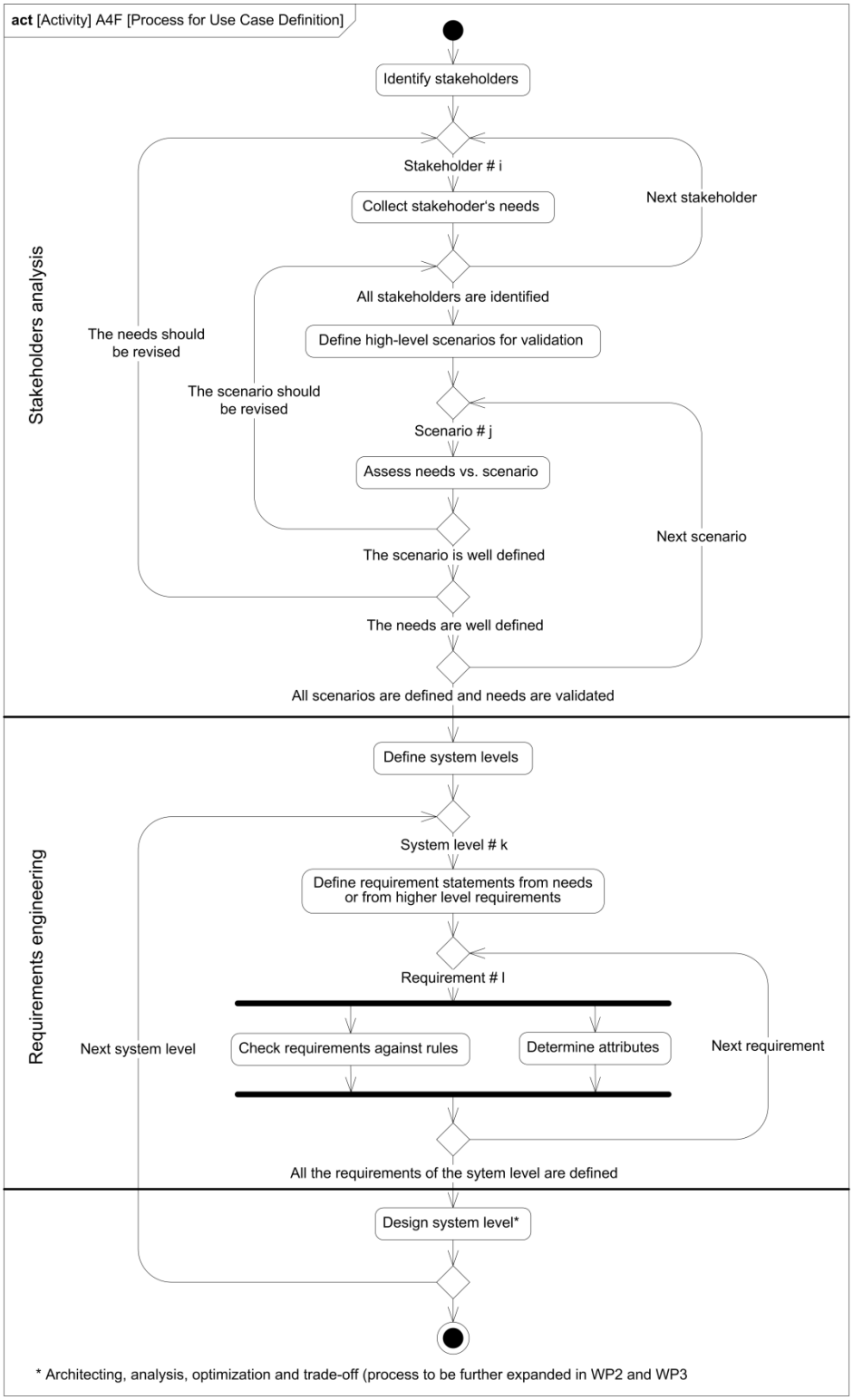


Fig. 12: Process for Application Case Definition.

A *Process* is defined as a logical sequence of tasks performed to achieve a particular objective. A process defines "WHAT" is to be done, without specifying "HOW" each task is performed. The "HOW" is defined by the *Method*, which consists of techniques for performing a task by means of certain *Tools* [16]. More specifically, the process described in the present Chapter is part of a *Systems Engineering Process*, which can be defined as a collection of interdisciplinary tasks that are required throughout a system's life cycle in order to transform the customer needs into a system solution.

The process identified in AGILE 4.0 for the Application Case Definition is organized in two parts: *Stakeholders analysis* and *Requirements engineering*.

6.1 Stakeholders analysis

The first part of the proposed process starts with the identification of all the stakeholders who have an interest on the system being developed and produced during the whole life-cycle of the system.

Subsequently, all the needs expressed by each stakeholder are collected. This collection can be done through interviews, workshops and market surveys.

It is mandatory for the success of the system to clearly identify all the stakeholders involved and to capture all their expectations. Otherwise, the produced system wouldn't be validated and consequently it would not be compliant with what demanded by the stakeholders. Therefore, the company who designs and builds the system has to define several high-level scenarios to represent and demonstrate to all the stakeholders how the system will operate and interact with users during its entire life cycle. In other words, scenarios are used to reach an agreement between the suppliers (i.e. who designs and produces the System of Interest) and the stakeholders about how the system will be compliant with the stakeholder needs. Scenarios are helpful to identify unclear needs that might potentially bring to wrong solutions. Each scenario is therefore assessed together with a subset of needs, and modifications of the scenarios are made in case of misunderstandings with the stakeholders. In addition, unclear needs can be modified, bringing eventually to a validated list of needs, well-conceived by the suppliers, which can be then transformed into requirements.

6.2 Requirements engineering

It can be noted from the schema of Fig. 12 that this part of the process is iterative, since the definition of requirements is also influenced by other design tasks, as architecting, analysis, optimization and trade-off studies. A System of Interest (e.g. *aircraft*) is composed by different levels, like *engine*, *compressor* and *blade*. In other words, the specifications of requirements at a certain level are determined by the results of designs performed at a higher level. This is for instance the case of requirements specifying the maximum mass of a component. This value of mass is indeed evaluated from the analysis of the entire system that includes the component.

Per each level of the System of Interest, the *Requirements engineering* part of the process targets the following tasks. First, the requirement statements are derived from needs or from other requirements (belonging to higher or same level). Their definition follows the guidelines addressed in Chapter 4. The requirement statements are built according to the requirement types and following the pre-defined requirement patterns described in Section 4.3. Then, the requirements are checked against all the rules reported in Section 4.4 and corrections are made. Finally, all the attributes of Section 4.5 are specified and included to complete all the requirement statements. These tasks are repeated until the requirements definition of that specific level is complete. Then, design activities are carried out to specify requirements at lower level. This process proceeds until all the system level have been addressed.

7 APPLICATION

The guideline presented in the previous Chapters is now used for the definition of stakeholders, needs, scenarios and requirements of an application case. The AGILE 4.0 Application Case 2 tackled in WP6.2 is selected as application case. The Application Case 2 deals with the design and production of a Horizontal Tail Plane (HTP) for a 90-passenger regional jet aircraft [26]. Application Case 2 investigates different production supply chains and production technologies. Therefore, the Systems of Interest of the present Application Case are two: aircraft and HTP.

7.1 Stakeholders analysis – example

Several stakeholders can be identified for the proposed Application Case. However, this example only mentions three stakeholders. The first stakeholder is the *Airline*, which is also considered as the customer of the product. The second stakeholder is the OEM, who manufactures the aircraft by design and producing some parts of it and outsourcing the design and production of other parts to other companies. These other companies are the third type of stakeholder considered in this example, i.e. *Tier Supplier*. The components produced by the OEM and all the suppliers are finally assembled together by the OEM.

The *Stakeholders analysis* phase proceeds with the collection of stakeholder needs. Tab. 1 lists some of the needs owned by the three stakeholders considered in this example.

Stakeholder	Need	ID
Airline	Aircraft able to take off/land in as many airports as possible	N_01
	Medium-range aircraft	N_02
OEM	The aircraft has to be competitive in the market	N_03
	To sell a large volume of aircraft	N_04
	Produce a safe and reliable aircraft	N_05
	To receive from supplier a well designed HTP	N_06
	To receive from supplier the htp on time	N_07
Tier Supplier	Receive clear htp interface requirements	N_08

Tab. 1: Stakeholders and Needs of the present example of application

It can be noted that the collected needs are vague and not structured. For instance, need N_01 is generic and it should be transformed into requirements stating specific metrics and targets, as take-off and landing field lengths. Moreover, need N_02 represents an example of un-structured statement, since not verbs or additional complements are specified. Section 7.2 addresses how these needs are transformed into structured, unambiguous, complete, feasible, verifiable and correct requirements. However, it is first important to check whether the collected needs are effectively those requested by the stakeholders. The validation of the system is therefore performed through the identification and assessment of scenarios.

In the present example, a scenario is defined to validate needs N_06, N_07 and N_08. The scenario focuses on the design of the production of the HTP by the Tier Supplier as demanded by the OEM. The scenario is used to check whether the OEM and the Tier Supplier have a common understanding on the identified subset of needs.

SCENARIO: Design and production of the HTP

- The Tier Supplier receives HTP interface requirements from the OEM
- The Tier Supplier designs the HTP according to the OEM's requirements
- The Tier Supplier checks raw materials availability, their capacity and capability
- The Tier Supplier produces the HTP
- The Tier Supplier verifies that the produced HTP is in line with the allowed tolerance specified by the OEM
- The Tier Supplier provides the produced HTP to the OEM on time

The scenario validates needs N_07 and N_08, since it includes the receiving of HTP requirements and the delivery on time of the completed product. However, the need N_06 is not validated because it is misunderstood by the Tier Supplier. According to this stakeholder, a "well designed" HTP is in line with the allowed tolerance specified by the OEM. However, the OEM might have additional metrics to categorize an HTP as "well designed". This metric is the surface roughness of the HTP, which has to be within a pre-defined limit. Therefore, the scenario should be modified to include the new activity:

The Tier Supplier verifies that the surface roughness of the HTP is within a pre-defined limit

In addition, need 52 was unclearly expressed, and it should be modified in:

To receive from the Tier Supplier an HTP with surface roughness of TBD μm - N_06

The validated set of needs can therefore be transformed into requirements.

7.2 Requirements engineering - example

The stakeholder needs listed in Section 7.1 are now transformed into requirements. Two system-levels are here considered. The former is the whole aircraft, while the latter is the HTP.

Tab. 2 collects some requirements at aircraft level. In the present application, examples of Functional, *Performance* and *Design* requirements are provided. The requirement statements are built following the patterns described in Section 4.3. All the requirement statements are checked against the rules provided in Section 4.4.

ID	Requirement Statement	Type	Parent
R_01	The aircraft shall have the sale price of maximum TBD $ in the market	Design	N_03; N_04
R_02	The aircraft shall take off with the Take Off Field Length of maximum 1500 m during the take-off condition flight	Performance	N_01
R_03	The aircraft shall land with the Landing Field Length of maximum 1400 m during the landing condition flight	Performance	N_01
R_04	The aircraft shall have the design range of minimum 3500 km	Design	N_02
R_05	The aircraft shall have the passengers number of maximum 90 in every flight	Design	N_02
R_06	The aircraft shall be compliant with CS - 25 regulation	Functional	N_05

Tab. 2: Subset of requirements at aircraft level of the present example of application

Additional requirement attributes are collected in Tab. 3 and Tab. 4. More specifically, the attributes of Tab. 4 regard the verification of the requirements.

ID	Author	Version	Priority	Syntax Verification	(Need) Stakeholder	Validation
R_01	Luca	0.1	Medium	Yes	OEM	Yes
R_02	Luca	0.1	High	Yes	Airline	Yes
R_03	Luca	0.1	Medium	Yes	Airline	Yes
R_04	Luca	0.1	Medium	Yes	Airline	Yes
R_05	Luca	0.1	Medium	Yes	Airline	Yes
R_06	Luca	0.1	High	Yes	OEM	Yes

Tab. 3: Additional requirement attributes of the present example of application

ID	MoC	Test case	(Responsible) Stakeholder	System Verification	Consequence
R_01	Costs analysis	PRICE	OEM	Not started	Loss of competitiveness in the market
R_02	OAD analysis	OpenAD	OEM	Not started	The aircraft could be not able to take off in some airports
R_03	OAD analysis	OpenAD	OEM	Not started	The aircraft could be not able to land in some airports
R_04	OAD analysis	OpenAD	OEM	Not started	Loss of competitiveness in the market
R_05	OAD analysis	OpenAD	OEM	Not started	The aircraft will not flight in the design conditions; waste of money, profit loss; financial penalty
R_06	Flight Test	Specific flight test	OEM	Not started	Aircraft is not allowed to fly

Tab. 4: Requirement attributes of the present example of application regarding the verification of the requirements

To conclude this Chapter, an example of HTP-level requirements is provided in Tab. 5. Again, all the statements are written following the patterns and rules presented in Chapter 4, and all the attributes are specified, although only a subset is provided in this example. It should be note that the requirement R_11 is derived from the modified need N_06.

ID	Requirement Statement	Type	Parent
R_10	The HTP shall have the sale price of maximum TBD $ in the market	Design	R_01
R_11	The HTP shall have a surface roughness of maximum TBD µm	Design	N_06

Tab. 5: Subset of requirements at HTP level of the present example of application

8 PUBLIC AVAILABILITY OF THE AGILE 4.0 MBSE ONTOLOGY

Stakeholders, Needs, Scenarios and Requirements are some of the elements that are part of the AGILE 4.0 MBSE Ontology being developed within the context of the project. This AGILE ontology will be expanded, revised and maintained during the project. More importantly, the ontology is being published as open access, therefore it can be re-used by any user inside and outside the project Consortium.

The AGILE 4.0 MBSE Ontology has been uploaded onto the Community page of the project on the Zenodo website[6]. Being characterized by a D.O.I., the ontology can be effectively cited, as [27]. The available files represent the meta-models, rendered by OWL, supporting the development of complex systems.

At the moment, the part of the ontology supporting the modelling of system requirements is available on Zenodo. Fig. 13 provides an example of representation of the ontology file.

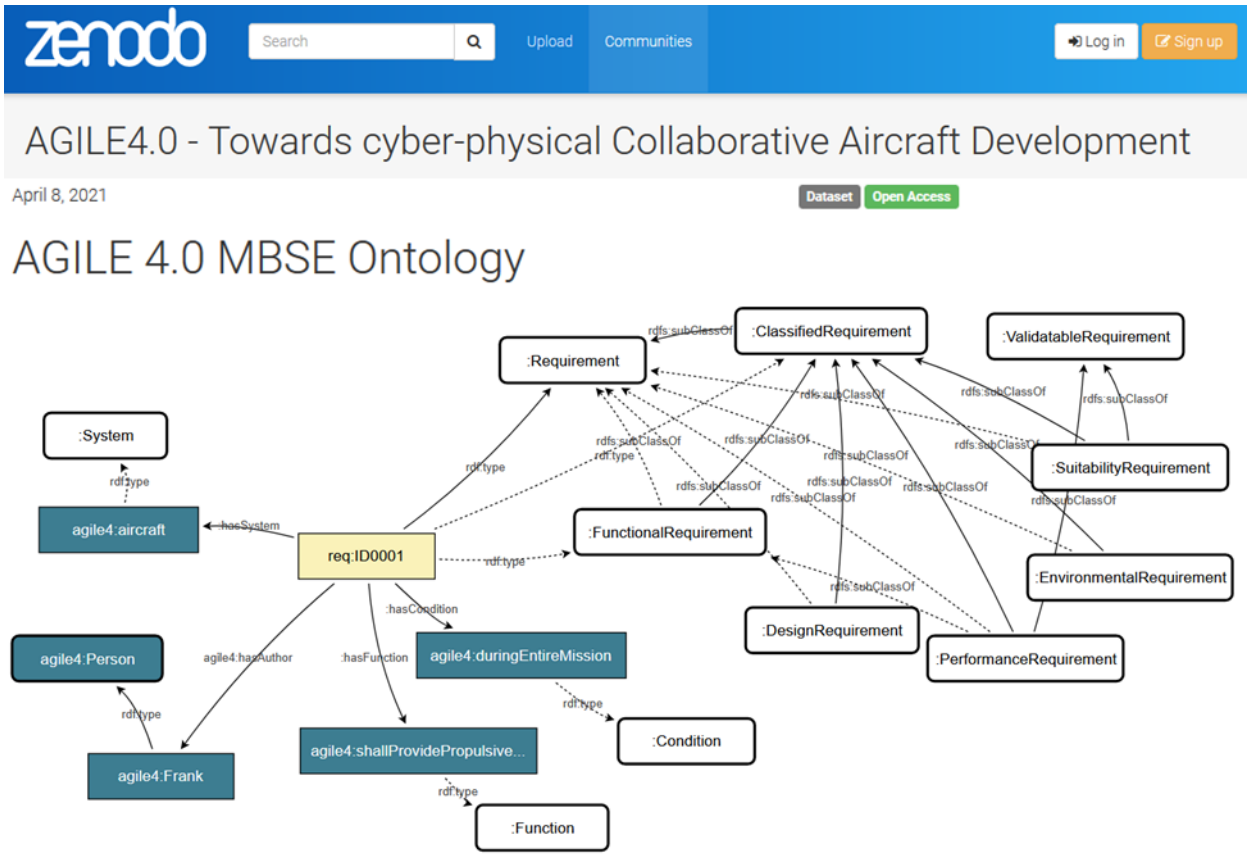


Fig. 13: Example of representation of the AGILE 4.0 MBSE Ontology (specifically for requirements modelling) openly accessible through Zenodo [27].

Finally, it is worth saying that the AGILE 4.0 MBSE Ontology has been developed to address the development of complex aeronautical systems investigated in the project, but it can be used to address any other kind of system.

[6] Link of the AGILE 4.0 Community on Zenodo: https://zenodo.org/communities/agile4/?page=1&size=20
Link of the AGILE 4.0 MBSE Ontology on Zenodo: https://zenodo.org/record/4671896

9 CONCLUSIONS AND FUTURE DIRECTIVES

This deliverable presents the process and guideline for the definition of Application Cases, meaning the identification of stakeholders, collection of their needs, analysis of scenarios for the validation of needs and development and management of requirements. The process and guideline for the Application Case Definition are followed to develop an example of application, and the main results are provided.

The process for Application Case definition has been iteratively developed in close collaboration with the envisaged stakeholders of guidelines: the Application Case developers in AGILE 4.0 WPs 6, 7 and 8. During the development of the process, it became clear that awareness and close involvement of stakeholders were important elements for acceptance, validation and are an essential basis for successful application of the method.

The process and guideline addressed in the deliverable can be applied in both document and model-based approaches.

REFERENCES

[1] P. D. Ciampa, G. La Rocca and B. Nagel, "A MBSE Approach to MDAO Systems for the Development of Complex Products," in *AIAA Aviation Forum*, Virtual, 2020.

[2] AGILE 4.0, "D1.2 Requirements & Specifications for AGILE 4.0 enabling technologies," AGILE 4.0 project (H2020-815122), 2020.

[3] AGILE 4.0, "D6.2 Production driven stream use case – document based," AGILE 4.0 project (H2020-815122), 2020.

[4] AGILE 4.0, "D7.2 Certification driven stream use case – document based," AGILE 4.0 project (H2020-815122), 2020.

[5] AGILE 4.0, "D8.2 Upgrade driven stream use case – document based," AGILE 4.0 project (H2020-815122), 2020.

[6] AGILE 4.0, "D4.2 Models for use case definition," AGILE 4.0 project (H2020-815122), 2020.

[7] AGILE 4.0, "D6.3 Production driven stream use cases – model based definition," AGILE 4.0 project (H2020-815122), 2021.

[8] AGILE 4.0, "D7.3 Certification driven stream use cases – model based definition," AGILE 4.0 project (H2020-815122), 2021.

[9] AGILE 4.0, "D8.3 Upgrade driven stream use cases – model based definition," AGILE 4.0 project (H2020-815122), 2021.

[10] International Organization for Standardization, "ISO/IEC/IEEE 42010 - Systems and software engineering - Architecture description," 2011.

[11] J. A. Zachman, "A framework for information systems architecture," *IBM systems journal*, vol. 26, no. 3, pp. 276-292, 1987.

[12] U.S. DoD, "The DoDAF Architecture Framework Version 2.02," 2010. [Online]. Available: https://dodcio.defense.gov/Library/DoD-Architecture-Framework/.

[13] UK Ministry of Defence, "MOD Architecture Framework," 2012. [Online]. Available: https://www.gov.uk/guidance/mod-architecture-framework.

[14] NATO, "NATO Architecture Framework Version 4," 2018.

[15] The Open Group, "The TOGAF® Standard, Version 9.2," 2018. [Online]. Available: https://www.opengroup.org/togaf.

[16] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," 2008.

[17] Object Management Group (OMG), "System Modeling Language (SysML)," [Online]. Available: https://www.omg.org/spec/SysML/About-SysML/.

[18] INCOSE, Systems Engineering Handbook v.3, 2006.

[19] AGILE 4.0, "D1.3 Process definition A4F for use case definition," AGILE 4.0 project (H2020-815122), 2020.

[20] NASA, Systems Engineering Handbook Rev 2, 2016.

[21] International Organization for Standardization, "ISO/IEC 15288 - Systems and Software Engineering - Software Life Cycle Processes," 2002.

[22] International Organization for Standardization, "ISO/IEC 29148 FDIS Systems and software engineering - Life cycle processes - Requirements engineering," 2011.

[23] SAE International, "ARP4754A - Guidelines for Development of Civil Aircraft and Systems," 2010.

[24] R. Carson, "Implementing structured requirements to improve requirements quality," in *INCOSE International Symposium*, Seattle (WA), 2015.

[25] INCOSE, "Guide for Writing Requirements, INCOSE-TP-2010-006-01," 2012.

[26] AGILE 4.0, "D6.1 SOTA AGILE workflows," AGILE 4.0 project (H2020-815122), 2020.

[27] *Boggero, Luca, Ciampa, Pier Davide, & Jepsen, Jonas. (2021). AGILE 4.0 MBSE Ontology [Data set]. Zenodo. http://doi.org/10.5281/zenodo.4671896.*

[28] D. Firesmith, "Prioritizing requirements," *The Journal of Object Technology*, vol. 3, no. 8, pp. 35-48, 2004.

APPENDIX – REQUIREMENT RULES

The present Appendix reports a subset of requirement rules identified in [25] and adopted in AGILE 4.0 project.

R1 - Use definite article "the" rather than the indefinite article "a"

Elaboration:
The definite article is "the"; the indefinite article is "a." When referring to entities, use of the indefinite article can lead to ambiguity. For example, if the requirement refers to "a user" it is unclear whether it means any user or one of the defined users for which the system has been designed. This causes further confusion in verification—for example, babies are arguably users of baby food, but the system would fail if the test agency sought to verify that a baby could order, receive, open, and serve (and even independently consume) baby food. On the other hand, if the requirement refers to "the User", the reference is explicitly to the nature of the user defined in the glossary—in the baby food example, the "User" is presumably the adult responsible for feeding the baby.

Examples:
Unacceptable: The system shall provide a time display. *{This is unacceptable because it is ambiguous—will any time display do? Is a one-off display of the time satisfactory? The writer's intention was most likely that they wanted the system to display continuously the current time, yet if the developer provided a constant display of "10:00 am" (or even a one-off display of any time), they could argue (albeit unreasonably) that they have met the requirement; yet they would have clearly failed to meet the customer's need and intent.}* *Acceptable*: The system shall display the Current_Time. *{Note that "Current_Time" must be defined in the glossary since there are a number of possible meanings and formats of the more-general term "current time."}*

Exceptions and relationships:
The purpose of this rule is to avoid the ambiguity that arises because "a" or "an" is tantamount to saying "any one of". In some cases, however, the use of an indefinite article is not misleading. For instance, "… with an accuracy of less than 1 second" allows the phrase to read more naturally and there is no ambiguity because of the accuracy quoted.

Characteristics that are established by this rule:
C3 – Unambiguous
C7 - Verifiable

R2 - Use the active voice in the main sentence structure with the responsible entity clearly identified

Elaboration:
The active voice requires that the entity performing the action is the subject of the sentence, since the onus for satisfying the requirement is on the subject, not the object of the sentence. If the entity responsible for the action is not identified explicitly, it is unclear who or what should perform the action making verification of that requirement very difficult. Including the entity in the subject also helps ensure the requirement refers to the appropriate level consistent with the entity name (see R3 /Accuracy/Subject). Often when the phrase "shall be" is used, the statement is in the passive voice.

Examples:
Unacceptable: The Identity of the Customer shall be confirmed. *{This is unacceptable because it does not identify who/what is responsible/accountable for confirming the identity.}* *Acceptable*: The Accounting_System shall confirm the Customer_Indentity. *{Note that" Accounting_System", "confirm", and "Customer_Indentity" must be defined in the glossary since there are a number of possible interpretations of these terms.}*

Characteristics that are established by this rule:
C2 - Appropriate
C3 - Unambiguous
C7 - Verifiable

R4 - Define terms

Elaboration:
Most languages are rich with words having several synonyms, each with a subtly different meaning. In requirements, shades of meaning will most likely lead to ambiguity and to difficulty in verification. Define the term in some form of ontology, including a glossary, data dictionary, or similar artifact that allows the reader of a requirement to know exactly what the writer meant when the word was chosen. The meaning of a term should be the same every time the word is used. A standard should be agreed upon to make the use of glossary terms identifiable in the requirements text; for example, glossary items may be capitalized and multiple words in single terms joined by an underscore (e.g. "Current_Time"). This is essential for consistency to avoid using the word with its general meaning. Definitions need to be agreed to throughout the project.

Examples:
Unacceptable: The system shall display the current time. *{This is unacceptable because it is ambiguous--what is 'current', in which time zone, to what degree of accuracy, in what format?}* *Acceptable*: The system shall display the Current_Time. *{Note that 'Current_Time' must then be defined in the glossary in terms of accuracy, format, and time zone.}*

Characteristics that are established by this rule:
C3 - Unambiguous
C7 - Verifiable
C11 - Consistent
C13 - Comprehensible
C14 – Able to be Validated

R6 - Use appropriate units when stating quantities

Elaboration:
All numbers should have units of measure explicitly stated. Temperatures must be stated in terms of the measurement system used—Celsius, Fahrenheit, or Kelvin, etc.).

Examples:
Unacceptable: The Circuit_Board shall have a storage temperature less than 30 degrees. *{This is unacceptable because the units used are incomplete. What system of measuring temperature?}* *Acceptable*: The Circuit_Board shall have a storage temperature of less than 30 degrees Celsius. Unacceptable: The system shall establish connection to at least 4 in less than or equal to 10 seconds. *{This is unacceptable because the units used are incomplete. "4" of what?}* *Acceptable*: The system shall establish connection to greater than 4 satellites in less than or equal to 10 seconds. *{Note that the term "connection" is acceptable at the business level, but would need elaboration at lower levels so that the requirement is decomposed and verifiable.}*

Characteristics that are established by this rule:
C3 - Unambiguous
C4 - Complete
C7 - Verifiable
C8 - Correct

R7 - Avoid the use of vague terms

Elaboration:
Avoid words that provide vague quantification, such as "some", "any", "allowable", "several", "many", "a lot of", "a few", "almost always", "very nearly", "nearly", "about", "close to", "almost", and "approximate". Avoid vague adjectives such as "ancillary", "relevant", "routine", "common", "generic", "significant", "flexible", "expandable", "typical", "sufficient", "adequate", "appropriate", "efficient", "effective", "proficient", "reasonable" and "customary." Vague adjectives can lead to ambiguous, unverifiable requirements that do not reflect accurately the stakeholder expectations. Adverbs qualify actions in some way and are particularly troublesome. Avoid vague adverbs, such as "usually", "approximately", "sufficiently", and "typically". Vague adverbs can lead to ambiguous, unverifiable requirements that do not reflect accurately the stakeholder expectations. As a general rule, words that end in "-ly" often result in ambiguity.

Examples:
Unacceptable: The Flight_Information_System shall usually be on line. *{This is unacceptable because "usually" is ambiguous - is availability what is meant?}* *Acceptable*: The Flight_Information_System shall have an Availability of greater than xx% over a period of greater than yy hours. *{Note that "Availability" must be defined in the glossary since there are a number of possible ways of calculating that measure.}* *Unacceptable*: The Flight_Information_System shall display the Tracking_Information for relevant aircraft. *{This is unacceptable because it does not make explicit which aircraft are relevant. Additionally, the statement allows the developer to decide what is relevant; such decisions are in the province of the customer, who should make the requirement explicit.}* *Acceptable*: The Flight_Information_System shall display the Tracking_Information of each Aircraft located less than or equal to 20 kilometers from the Airfield. *{Now it is clear for which aircraft the information needs to be displayed. Note that "Aircraft", "Tracking_Information", and "Airfield" must be defined in the glossary.}*

Exceptions and relationships:
R3 points out that the use of a verb such as "safe", may be acceptable at the business level as long as it is unambiguous at that level, decomposed at the lower levels, and is verifiable at the level stated. Similarly, some vague adjectives may be allowable at the business level, providing they are not ambiguous at that level. For example, it may be appropriate for the business to state "The system shall provide a user-friendly interface." Although '"user-friendly" will need to be decomposed and verified at lower levels before it is useful, it is arguably sufficiently clear at the business level.

Characteristics that are established by this rule:
C3 - Unambiguous
C4 - Complete
C7 - Verifiable

R9 - Avoid escape clauses

Elaboration:
Escape clauses give an excuse to the developer of the system at lower levels to not bother with a requirement. They provide vague conditions or possibilities, using phrases such as "so far as is possible", "as little as possible", "where possible", "as much as possible", "if it should prove necessary", "if necessary", "to the extent necessary", "as appropriate", "as required", "to the extent practical", and "if practicable." Escape clauses can lead to ambiguous, unverifiable requirements that are open to interpretation and that do not reflect accurately the stakeholder expectations. From a contracting standpoint, requirements with these phrases could be interpreted as being optional.

Examples:
Unacceptable: The GPS shall, where there is sufficient space, display the User_Location. *{This is unacceptable because whether there is sufficient space is vague, ambiguous, and unverifiable. The requirement is clearer without the escape clause.}*

Acceptable: The GPS shall display the User_Location. *{Note that "GPS" and "User_Location" must be defined in the glossary. Specific performance requirements need to be defined as well such as within what time, format, and accuracy.}*

Unacceptable: The Television shall, to the extent necessary, conform to the requirements in Section a.1 of Standard XYZ. *{This is unacceptable because it is unclear who is to determine the meaning of 'necessary'. The requirement is clearer without the escape clause.}*

Acceptable: The Television shall conform to the requirements in Section a.1 of Standard XYZ. *{Note that this form of requirement means that each requirement of Standard XYZ is to be met—if only a subset are required, that subset must be listed explicitly.}*

Characteristics that are established by this rule:
C3 - Unambiguous
C4 - Complete
C7 - Verifiable

R10 - Avoid open-ended clauses

Elaboration:
Open-ended clauses say that there is more required without stating exactly what. Avoid non-specific phrases such as "including but not limited to", "etc." and" and so on." Open-ended clauses can lead to ambiguous, unverifiable requirements that do not reflect accurately the writer's expectations and create ambiguity in the mind of the reader. Use of open-ended clauses also violates the one thought rule and singular characteristic. If more cases are required, then include additional requirements that explicitly state those cases. Depending on the contract type (fixed price vs level of effort) opened ended requirements can lead to serious interpretation problems concerning what is in or out of scope of the contract.

Examples:
Unacceptable: The ATM shall display the Customer Account_Number, Account_Balance, and so on. *{This is unacceptable because it contains an opened list of what is to be displayed.}* *Acceptable: {Split into as many requirements as necessary to be complete. Note that the types of customer information to be displayed needs to be defined in the glossary.}*: The ATM shall display the Customer Account_Number. The ATM shall display the Customer Account_Balance. The ATM shall display the Customer Account_Type. The ATM shall display the Customer Account_Overdraft_Limit.

Characteristics that are established by this rule:
C3 – Unambiguous
C4 - Complete
C5 - Singular
C7 - Verifiable

R11 - Avoid superfluous infinitives

Elaboration:
We sometimes see a requirement that contains more verbs than necessary to describe a basic action, such as “The system shall be designed to be able to ...” or “The system shall be designed to be capable of ...” rather than simply “The system shall ...” Think ahead to verification. If the system is able to, or is capable of, doing something one time but fails 99 times, the system has not met the requirement. Note that at the enterprise and business levels, requirements for an entity to “provide a capability” are acceptable. Where capability is made up of people, processes, and products; these requirements will be decomposed to address the people aspects (skill set, training, roles, etc.), processes (procedures, work instructions, etc.); and products (hardware and software systems). The parent requirement will be allocated to the people, processes, and products, as appropriate. When the resulting requirement sets are implemented for all three areas, the capability will exist to meet the entity needs.

Examples:
Unacceptable: The Weapon_Subsystem shall be able to store the location of each Ordnance. *{This is unacceptable because it contains the superfluous infinitive “be able to”.}* *Acceptable*: The Weapon_Subsystem shall store the location of each Ordnance. *{Note that the terms “Weapon_Subsystem” and “Ordnance” must be defined in the glossary.}*

Characteristics that are established by this rule:
C3 – Unambiguous
C7 - Verifiable

R17 - Avoid the use of “not”

Elaboration:
The presence of “not” in a requirement implies “not ever”, which is impossible to verify in a finite time. Rewriting the requirement to avoid the use of “not” results in a requirement that is clearer and is verifiable.

Examples:
Unacceptable: The <system> shall not fail. *{This is unacceptable because verification of the requirement would require infinite time.}* *Acceptable*: The <system> shall have an Availability of greater than or equal to 95%. *Unacceptable*: The <system> shall not contain mercury. *{This is unacceptable because verification of the requirement would require infinite precision. In addition, the real requirement may not be stated, e.g., the real concern may be the use of toxic materials, not just mercury. If that is the case, it may be best to reference a standard from a governmental agency concerning allowable exposures to a list of common toxic materials.}* *Acceptable*: The <system> shall limit metallic mercury exposures to less than or equal 0.025 mg/m3 over an 8-hour workday.

Exceptions and relationships:
It may be reasonable to include “not” in a requirement when the logical “NOT” is implied—for example when using not [X or Y]. In that case, however, in accordance with Rule 16, it may be better to capitalize the “NOT” to make the logical condition explicit: NOT [X or Y]. There may be other cases such as “The <system> shall not be red in color.” Which is stating a constraint and is verifiable, as long as the range of shades of red is stated (RBG rr,bb,gg range or a “name” of red in some standard).

Characteristics that are established by this rule:
C3 - Unambiguous
C7 - Verifiable

R26 - Avoid the use of pronouns and indefinite pronouns

Elaboration:
Repeat nouns in full instead of using pronouns to refer to nouns in other requirement statements. Pronouns are words such as “it”, “this”, “that”, “he”, “she”, “they”, and “them.” When writing stories, pronouns are a useful device for avoiding the repetition of words; but when writing requirements, pronouns are effectively cross-references to nouns in other requirements and, as such, are ambiguous and should be avoided. When originally written the noun that defines the pronoun may have proceeded the pronoun in the previous requirement, however, as the set of requirements mature, individual requirements may be deleted, reordered, or regrouped, and the defining requirement is no longer nearby. This is especially true when requirements are stored in a requirement management tool where they exist as single statements that may not be in order. To avoid these problems, repeat the proper nouns rather than using a pronoun. Indefinite pronouns are words such as “all”, “another”, “any”, “anybody”, “anything”, “both”, “each”, “either”, “every”, “everybody”, “everyone”, “everything”, “few”, “many”, “most”, “much”, “neither”, “no one”, “nobody”, “none”, “one”, “several”, “some”, “somebody”, “someone”, “something”, and “such.” Indefinite pronouns stand in for unnamed people or things, which makes their meaning subject to interpretation, ambiguous, and unverifiable.

Examples:
Unacceptable: The controller shall send the driver his itinerary for the day. It shall be delivered at least 8 hours prior to his Shift. *{This is unacceptable because the requirement is expressed as two sentences (R19), and the second sentence uses the pronouns “it” and “his.”}* *Acceptable*: The Controller shall send the Driver_Itinerary for the day to the Driver at least 8 hours prior to the Driver_Shift. *{Note use of the glossary to define terms and to be explicit about the relationship between the driver, shift, and the itinerary for that particular driver.}*

Characteristics that are established by this rule:
C3 – Unambiguous
C4 – Complete
C7 - Verifiable

R32 - Express each requirement once and only once

Elaboration:
Avoid including the same requirement more than once, either as a duplicate or in similar form. Exact duplicates are relatively straightforward to identify; finding similar requirements with slightly different wording is much more difficult, but is aided by the consistent use of defined terms and by classification.

Examples:
Exact duplicates can be found by matching of text strings. The main problem is to identify similarities with different expressions. For example: “The system shall provide a report of financial transactions containing the information defined in <some standard or contract deliverable listing>” and “The system shall provide a financial report” are overlapping in that the first is a subset of the second. Avoidance of repetition can be assisted by classification so that a subset of requirements can be compared.

Characteristics that are established by this rule:
C1 – Necessary
C9 – Conforming
C11 – Consistent
C12 - Feasible

R33 - Avoid stating a solution unless there is rationale for constraining the design

Elaboration:
Focus on the problem “what” rather than the solution “how.” Every system endeavor should have a level of requirements that captures the problem to be solved without involving solutions. System requirements should provide a high-level specification of the overall solution. The first level of architecture may be laid out, but subsystems are considered as black-boxes to be elaborated at the next level down. When reviewing a requirement that states a solution, again ask yourself “for what purpose?” The answer will reveal the real requirement. (Notice that this question is subtly different from simply asking “Why?”, and encourages a teleological rather than causal response.) The answer to this question has the potential to help you do three things: 1. Rephrase the requirement in terms of the problem being solved. 2. Determine if the requirement is at the right level. 3. Identify which higher-level requirement(s) or need(s) this one addresses. Often when rationale is provided with a requirement, for requirements that state a solution, the rationale should answer the “for what purpose?” question, and thus the real requirement may be extracted from the rationale.

Examples:

Unacceptable: Traffic lights shall be used to control traffic at the junction. *{This is unacceptable because "Traffic lights" are a solution.}*

Acceptable (several requirements):

• The Traffic_Control_System shall signal "Walk" when the Pedestrian is permitted to cross the road at the junction. *{This is the motivating purpose.}*

• The Traffic_Control_System shall limit the Wait_Time of Vehicles traversing the Junction to less than 2 minutes during normal daytime traffic conditions.

Unacceptable: By pressing a button on the traffic-light pillar, the Pedestrian signals his presence, and the light turns red for the traffic to stop. *{This is unacceptable because this requirement contains solution biased detail.}*

Acceptable: The Traffic_Control_System shall allow the Pedestrian to signal intent to cross the road. *{This requirement allows freedom in determining the best solution, which may be a means of automatic detection rather than button pushing.}*

{Note that glossary definitions should be used for "Pedestrian", "Vehicle", and "Junction."}

Exceptions and relationships:

Sometimes solutions have to be described in requirements, even if it is very detailed for a given level, for example if the airworthiness authorities require the use of a specific template for a certain report; or if a naval customer requires that the new naval vessel be equipped with a specific gun from a specific supplier; or, if all vehicles in a fleet are required to use the same fuel. In these cases, it is not a premature solution, but a real stakeholder or customer requirement.

However, as a general rule, if something very detailed is expressed at a given level without proper justification, it may be a premature solution.

Examples of exceptions include:

1. There is a common known solution and not referring to this solution will require adding many requirements to specify it. This can result in a redundant set of requirements that are not necessary, will have to be verified – adding additional cost to the project.

2. The requirements are based on prototyping and an associated trade study that resulted a specific solution that meets the needs of the stakeholders.

3. The enterprise has prescribed a solution for a subsystem (for example, a car company may mandate that the next model make use of a particular engine).

Characteristics that are established by this rule:

C2 - Appropriate

R35 - Define quantities with a range of values appropriate to the level stated

Elaboration:

When it comes to defining performance, single-point values are seldom sufficient and are difficult to test. Ask yourself the question: "if the performance was a little less (or more) than this, would I still buy it?" If the answer is yes, then change the requirement to reflect this.

It also helps to consider the underlying goal: are you trying to minimize, maximize or optimize something? The answer to this question will help determine whether there is an upper bound, lower bound, or both.

State the quantities contained in a requirement with ranges or limits with a degree of accuracy that is appropriate to the level at which the requirement is stated. Care should be taken to avoid unspecified value ranges and ensure the quantities are expressed with tolerances or limits. There are two reasons for this:

1. Several requirements may have to be traded against each other, and providing tolerances or limits is a way of describing the trade-space. Seldom is a quantity in a requirement absolute. A range of values is usually acceptable, providing different performance levels.

2. Verification against a single absolute value is usually not possible or at best very expensive and time consuming, whereas verification against a defined range of values with upper and lower limits makes verification more manageable.

Care should also be taken to ensure the tolerances are no tighter than needed. Tighter tolerances can drive costs both in system design and manufacturing as well as verifying the system can perform within the tighter tolerances.

Examples:

Unacceptable: The Pumping_Station shall maintain the flow of water at 120 liters per second for 30 minutes. *{This is unacceptable because we don't know whether a solution that carries more or less than the specified quantities is acceptable.}*

Acceptable: The Pumping_Station shall maintain a flow of water at 120 ±10 liters per second for greater than 30 minutes. *{Now we know the range of acceptable flow performance, and that the 30 minutes is a minimum acceptable performance.}*

Unacceptable: The Flight_Information_System shall display the current altitude to approximately 1 meter resolution. *{This is unacceptable because it is imprecise. What is "approximately" in the context of a distance of 1 meter? Who has the option of deciding what is "approximately"? How will "approximately" be verified? What is the acceptable tolerance? Further, altitude is more commonly described in units of feet, not meters.}*

Acceptable: The Flight_Information_System shall display Current_Altitude with an accuracy of ±3 feet. *{Note that "Current_Altitude" must be defined in the glossary since there are a number of possible interpretations of the term.}*

Unacceptable: The System shall limit arsenic contamination in the drinking water to allowable levels. Rationale: Arsenic contamination in drinking water can cause health problems. *{This is unacceptable because allowable is ambiguous - allowable by whom? What specific concentration is allowable? In what market?}*

Unacceptable: The System shall limit arsenic contamination in the drinking water to 1 part per trillion. Rationale: Arsenic contamination in drinking water can cause health problems. *{This is unacceptable because the EPA contamination limit in drinking water is 10 parts per million. Requiring a tighter limit may be beyond the ability of current technology to measure or if measuring concentrations of 1 part per trillion are possible, the cost to do so may be unacceptably high. Also, no range is specified. Using less than is probably the real intent.}*

Acceptable: The System shall limit arsenic contamination in the drinking water to less than10 parts per billion. *Rationale: EPA set the arsenic standard for drinking water at 10 ppb (or 0.010 parts per million). The EPA has determined that concentrations of this level or less will protect consumers from the effects of longterm, chronic exposure to arsenic.*

Characteristics that are established by this rule:
C3 – Unambiguous
C4 – Complete
C6 – Feasible
C7 – Verifiable
C8 – Correct
C12 - Feasible

R37 - Define temporal dependencies explicitly instead of using indefinite temporal keywords

Elaboration:
Some words and phrases signal non-specific timing, such as “eventually”, “until”, “before”, “when”, “after”, “as”, “once”, “earliest”, “latest”, “instantaneous”, “simultaneous”, “while”, and ”at last”, are ambiguous and not verifiable. Indefinite temporal keywords can cause confusion or unintended meaning. These words should be replaced by specific timing constraints.

Examples:
Unacceptable: Continual operation of the pump shall eventually result in the tank being empty. *{This is unacceptable because “eventually” is ambiguous. Also, this statement is not written in the proper form. It is Written on “operation” rather than on a requirement on the pump which violates R3.}* *Acceptable*: The Pump shall remove greater than 99% of the Fluid from the Tank in less than 3 days of continuous operation.

Characteristics that are established by this rule:
C3 – Unambiguous
C4 – Complete
C7 - Verifiable

R38 - Use each term consistently throughout requirement sets

Elaboration:
R4 requires the definition of terms in each requirement. In addition to that rule, terms must be used consistently throughout the requirement set. A common ontology therefore needs to be defined for each project. Use just one defined term per concept in the whole set of requirements. Synonyms are not acceptable.

Examples:
Unacceptable: It would not be acceptable for one requirement to refer to an entity using one term and another to refer to the same entity using another term.

For example, in a subsystem specification, the following three requirement statements:

The radio shall ....

The receiver shall ....

The terminal shall ....

If each term refers to the same subject, the statements need to be modified to use the same word (or, if they are meant to be different, the words must be defined to be so).

Acceptable: Settle on only one term, define it in the glossary, and then use it consistently in each requirement.

Characteristics that are established by this rule:
C3 – Unambiguous
C8 – Correct
C9 – Conforming
C11 – Consistent
C13 – Comprehensible
C14 – Able to be Validated

R40 - Avoid the use of abbreviations in requirement statements

Elaboration:
The use of abbreviations adds ambiguity and needs to be avoided.

Examples:
Unacceptable: It would not be acceptable for one requirement to refer to the abbreviation ”op” meaning operation and another to refer to the ”op” meaning the operator. *Acceptable: Avoid the abbreviation and use the full term every time.*

Characteristics that are established by this rule:
C9 – Conforming
C11 – Consistent
C13 – Comprehensible
C14 – Able to be Validated

R44 - Conform to a defined structure or template for sets of requirements and patterns for individual requirement statements

Elaboration:
A well-structured set of requirements allows an understanding of the whole set of requirements to be assimilated without undue cognitive loading on the reader. Despite what has been stated about the

completeness of individual requirement statements, it is often easier to understand a requirement when it is placed in context with other related requirements.

Whether presented in document structure or as the result of database queries, the ability to draw together related groups of requirements is a vital tool in identifying repetition and conflict between requirement statements. Templates for organizing requirements will help ensure consistency and completeness of your requirement set. Patterns for individual requirement statements help ensure consistency and completeness of the requirement set.

Examples:
For sets of requirements, an outline can be defined that organizes requirements in categories or types such as: functional/performance, operational, quality (-ilities), design and construction standards, and physical characteristics. The outlines can come from international standards or an organizational standard tailored to the organization's domain and different types of products within that domain. (The organization for system level requirements may be different than the organization of a set of software requirements.) *To ensure requirements are consistently structured, the organization needs to include pre-defined patterns in their requirement development and management process. The patterns can come from an international standard or an organizational standard set of patterns for requirements statements based on type of requirement. If using a requirement management tool, a standard schema should be defined as part of your requirement development and management process. The patterns need to be tailored to the organization's domain and the different types of products within that domain.*

Characteristics that are established by this rule:
C10 – Complete
C11 – Consistent
C13 – Comprehensible
C14 – Able to be Validated