SHERPA ENGINEERING	SHERPA ENGINEERING 12 AV de Verdun - 92250 La Garenne-Colombes Tel. +33 1.47.82.81.85 - Fax +33 1.47.82.00.96 SA au capital de 412.400 € - APE : 7112B - SIRET : 413 367 228 00017

TECHNICAL NOTE TN 107.3

System Engineering applied to the MELiSSA Data Management System. Requirements.

Client	**ESA**
Contract References	**SHERPA:*P3099*** **ESA:*TEC-MMG/2011/154/In/BL***
Document Reference	**TN 107.3 System Engineering applied to the MELiSSA Data Management System. Requirements. Issue 1 rev 9.doc**
Document Version	**1.9**

Prepared by/Préparé par	Olivier Gerbi, Jean Brunet
Reference/Référence	TEC-MMG/2011/154/In/BL
Issue/Edition	1
Revision/Révision	9
Date of issue/Date d'édition	October 2011
Status/Statut	Final

APPROVAL

Title Titre	System Engineering applied to the MELiSSA Data Management System	Issue Edition	1	Revision Révision	9

Authors Auteur	Olivier Gerbi, Jean Brunet	Date Date	12/10/2011

Approved by Approuvé par	Brigitte Lamaze	Date Date	

CHANGE LOG

Reason for change	Issue/Edition	Revision/Révision	Status/Statut	Date/Date
Creation	1	0	Draft	27/07/2011
Update after meetings	1	1	Draft	02/09/2011
Update	1	2	Draft	05/09/2011
Update after meetings	1	3	Draft	27/09/2011
Update	1	4	Draft	28/09/2011
Update	1	5	Draft	29/09/2011
Update	1	6	Draft	30/09/2011
Update	1	7	Draft	30/09/2011
Update	1	8	Draft	04/10/2011
Update	1	9	Final	12/10/2011

Distribution List

Name/Nom	Company/Société	Quantity/Quantité
Brigitte LAMAZE	ESA	

TABLE OF CONTENTS

TABLE OF FIGURES

1. Introduction

This document aims to define the main definition and concepts of the system engineering, necessary for introducing the requirements and the way to write them.
These concepts are applied to MELiSSA and implemented into the MELiSSA Data Management System.

A chapter is devoted to the requirements elaboration. Some rules and tips are given for helping the user to write a requirement.
A methodology for the elaboration of requirements documents is proposed.

2. Acronyms

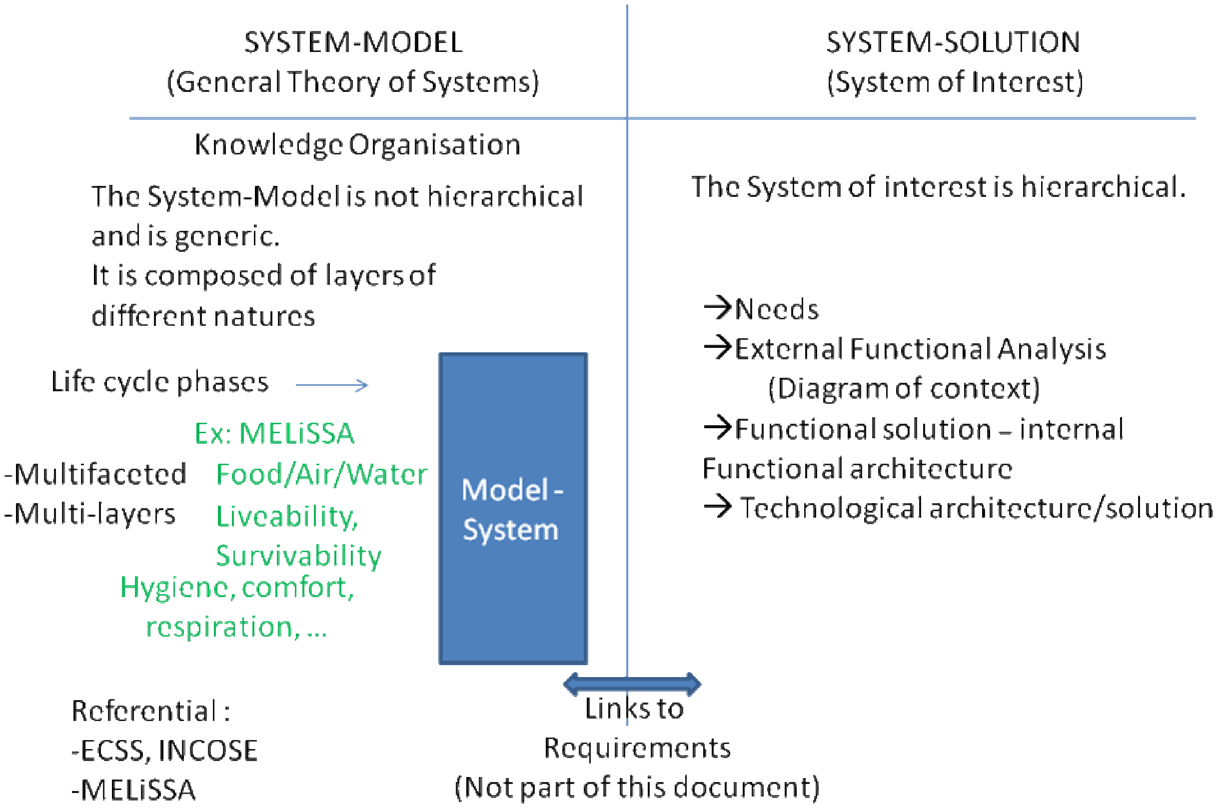
AFIS	: Association Française d'Ingénierie Système
DMS	: Data Management System
HW	: Hardware
INCOSE	: INternational Council on Systems Engineering
IVVQ	: Integration, Verification, Validation, Qualification
LSS	: Life Support Systems
MELiSSA	: Micro-Ecological Life Support System Alternative
RAMS	: Reliability, Availability, Maintainability and Safety
RD	: Reference Document
SE	: Systems Engineering
SW	: Software
TBD	: To Be Defined (or To Be Determined)
V&V	: Verification and Validation

3. Reference documents

[1] Systems Engineering Handbook. A guide for system life cycle processes and activities. Version 3.1. August 2007. INCOSE-TP-2003-002-03.1
[2] Glossaire de Base de l'Ingénierie d Systèmes. Version 1.2. 5 octobre 2004. AFIS
[3] Data Management System Specification. ESA/MELiSSA, JP Maigret/ Anissa Marouard, 10 février 2009.
[4] NASA Systems Engineering Handbook. NASA/SP-2007-6105 Rev1.
[5] ECSS-E-ST-10C – Space Engineering – System engineering general requirements. 6 March 2009
[6] ECSS-P-001B – Glossary of terms. 14 July 2004

4. Overview of the System Engineering approach applied to Life Support Systems

This following schematic shows the fundamentals of the approach defined for Life Support System.
It is voluntary limited to the "just necessary" for the proper understanding of the methodology to be followed for the establishment of requirements.



This document only focuses on the SYSTEM-SOLUTION and not the SYSTEM-MODEL (model of the system).

All definitions are provided in the chapter here after.

5. Definitions Referential

5.1. System's associated definitions, System Hierarchy, System Life Cycle

5.1.1. System-model

The system-model is a generic model based on the knowledge organisation. The system-model is generic and not hierarchical.
The system-model is not the scope of this document.

5.1.2. System-solution= SYSTEM in the rest of the document

An integrated set of elements, subsystems, or assemblies that accomplish a **defined objective**. These elements include products (hardware and software), processes, people, information, techniques, facilities, services, and other support elements.

5.1.3. SYSTEM OF INTEREST

The system whose **life cycle** is under consideration.

5.1.4. ENABLING SYSTEM

A system that complements a system-of-interest during its life cycle stages but does not necessarily contribute directly to its function during operation
Example: when the system enters the production stage, an enabling production system is required.
For C3 as system of Interest, one enabling system can be the system providing the necessary inputs to the system (C2 for instance).

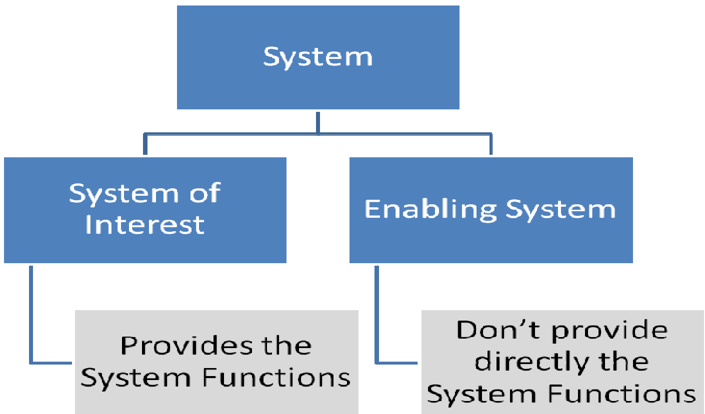


Figure 1: System of Interest & Enabling System

5.1.5. META-SYSTEM

The rules, standards, norms, principles, grammars, theories and sciences that must be met to design a system

5.1.6. System hierarchy

5.1.6.1. *UPPER SYSTEM*

Structured environment of the system consistent with the objectives or mission of the system. It is the upper system that justifies the existence of the system.
With an upper system, there is a loop between the system of interest and the upper system. The actors belong to the upper system.

In the frame of the MELiSSA project and in order to be compliant with the DMS, the hierarchy of systems is limited to the following:

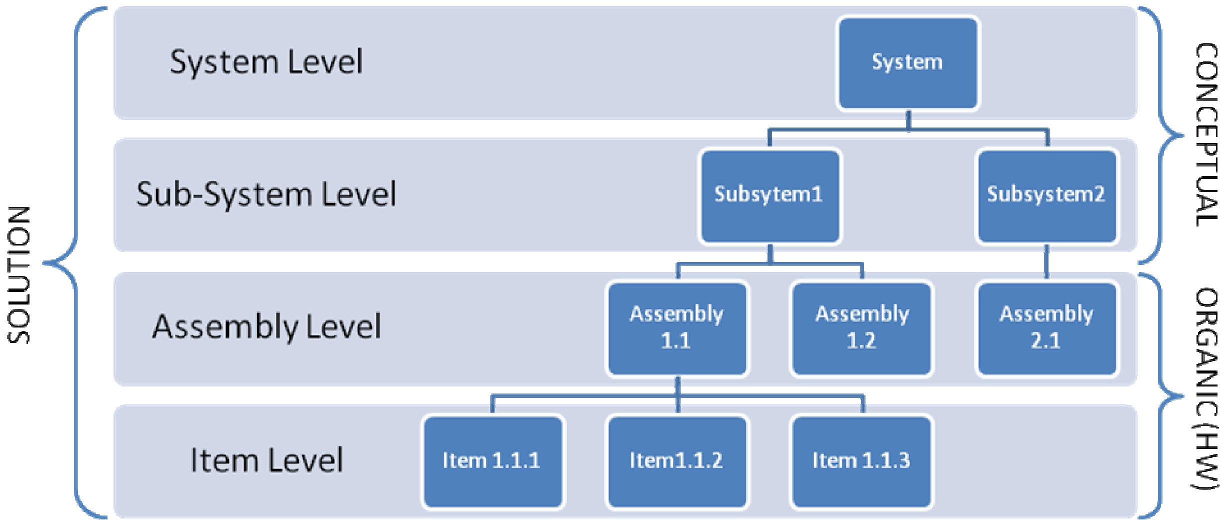


Figure 2: System Hierarchy

Hereafter the definitions (RD[1]) for the different levels.

We can discriminate, 2 families. The 2 upper levels (System and Sub-Systems) are at the abstraction/conceptual solution level ("System") and the 2 lower levels (Assembly and Items) is the organic Solution (hard-physical).

5.1.6.2. *SUBSYSTEM*

An integrated set of assemblies, components, and parts which performs a cleanly and clearly separated function, involving similar technical skills, or a separate supplier.

5.1.6.3. *ASSEMBLY*

An integrated set of components and/or subassemblies that comprise a defined portion of a subsystem.

5.1.6.4. *ITEM (or COMPONENT or PART)*

The lowest level of separately identifiable item.
Remark: these 3 concepts (item, component and parts) have been grouped in the frame of the project.

5.1.6.5. *Black box representation*

Representation of a device, system or object in terms of its input, output and transfer characteristics without any knowledge required of its internal working. The black box representation gives the external vision of a system following 4 external views:

• Contribution to a purpose, a mission, (**why**)
• Relations with organized outside (environment system) (**to what**)
• Which properties are emerging from global functions of the system (**what**)
• Its transformation during time (**evolution**)

5.1.6.6. *White Box representation*

Description of internal functioning of a system in respect with external requirements

5.1.7. SYSTEM LIFE CYCLE

The evolution with time of a system-of-interest from conception through to retirement.

The usual Life Cycle stages are the following:

LIFE CYCLE STAGES	PURPOSE
CONCEPT	*Identify stakeholders' needs* *Explore concepts* *Propose viable solutions*
DEVELOPMENT	*Refine system requirements* *Create solution description* *Build organic solution* *Verify and validate system*
PRODUCTION	*Produce systems* *Inspect and test [verify]*
UTILIZATION	*Operate system to satisfy users' needs*
SUPPORT	*Provide sustained system capability*
RETIREMENT	*Store, archive, or dispose of the system*

Figure 3: Life Cycle Stages

For memory:
In Space Project, another dimension can also be considered. It represents the phase of a project as defined in ECSS :0, A, B, C, D, E, F
For each phase, the Life Cycle stages are applicable.

Phase 0	Mission analysis-need identification
Phase A	Feasibility
Phase B	Preliminary definition
Phase C	Detailed definition
Phase D	Qualification and production
Phase E	Operations / utilization
Phase F	Disposal

The Phase (if applicable) can be indicated when establishing the definition of the activity.

5.2. Context Diagram and associated definitions

The system engineering approach shall start with the Context Diagram.
We can define it as:

5.2.1. Context Diagram

The diagram summarizes the process activities and their inputs and outputs from/to external **actors**; some inputs are categorized as controls and enablers.
A control governs the accomplishments of the process; an enabler is the means by which the process is performed.

A schematic representation is the following.

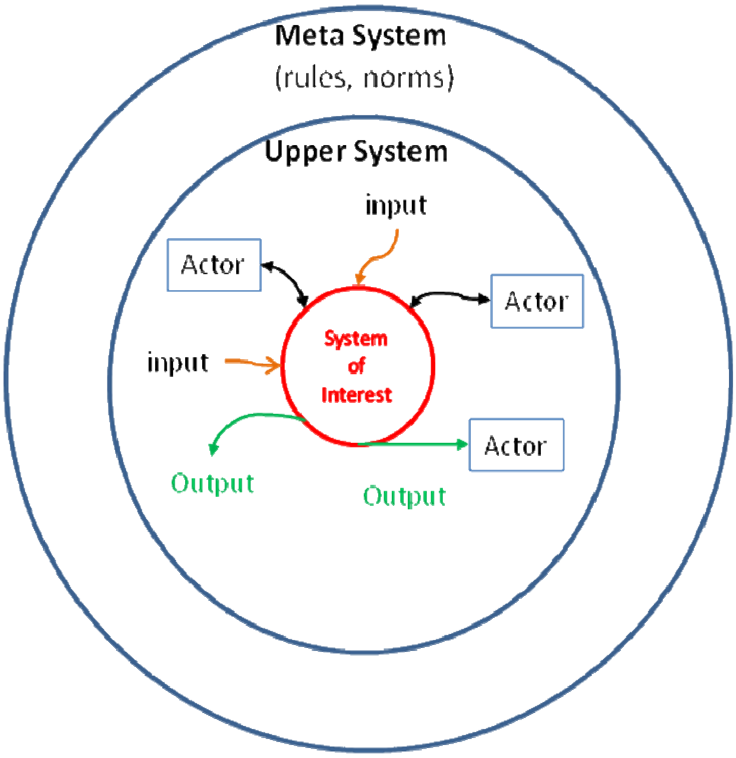


Figure 4: Context Diagram

This diagram is the highest level view of a system.

The objective of a system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of system requirements and constraints.

5.2.2. Actor

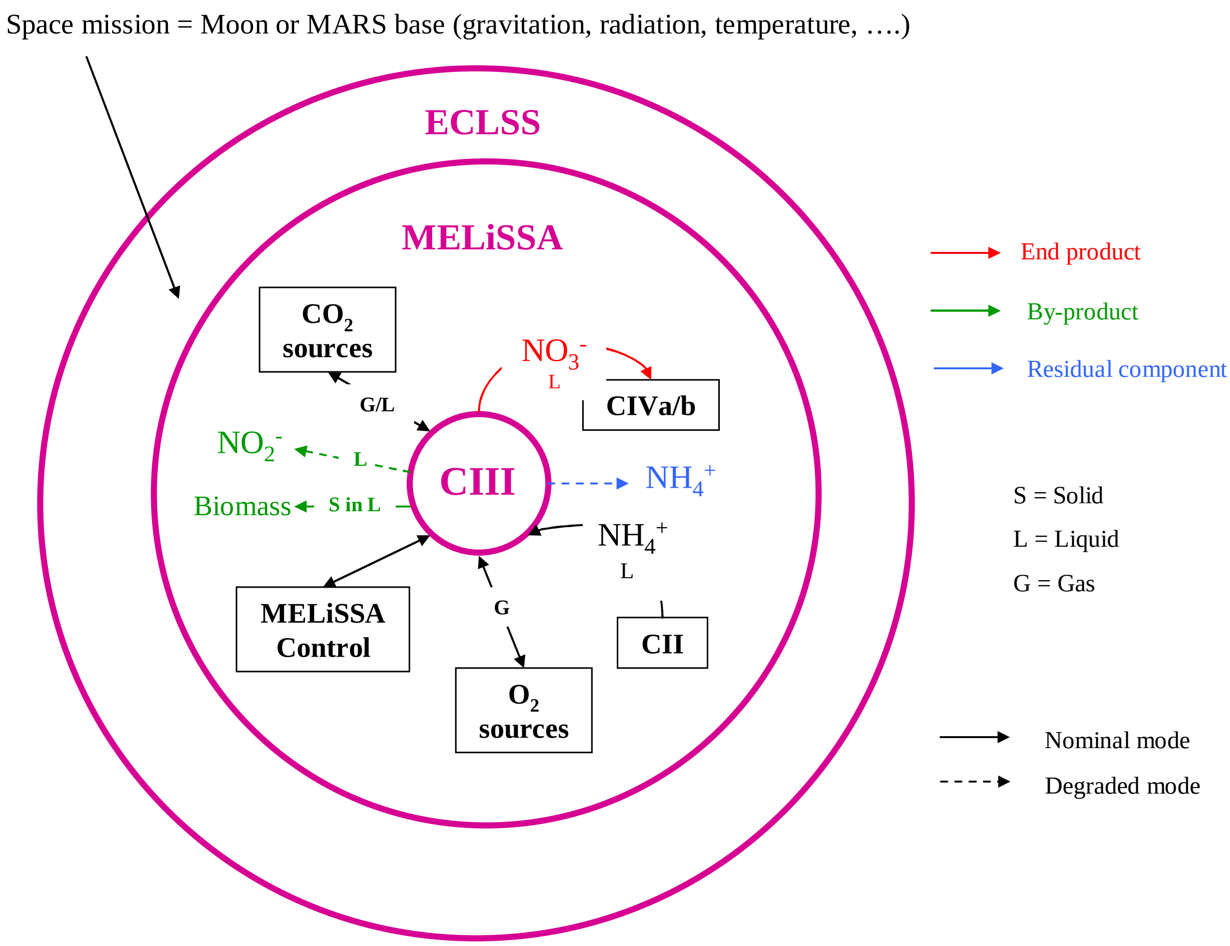
An actor specifies a role played by a person or thing when interacting with the system.
The actor is external to the system.

The environment of the system of interest is defined as followed:

5.2.3. Environment

The surroundings (natural or man-made) in which the system-of-interest is utilized and supported; or in which the system is being developed, produced or retired.

Example of Context Diagram :



Context Diagram – Black Box view

In this example (which can be not exhaustive),

- System of Interest : CIII
- Actors : CII, CIVa/CIVb, …
- Inputs: $NH4^{+}$ …
- Outputs: $NO2^{-}$,$NO3^{-}$

5.3. Mission, Operational Scenario (or Mission Profile) and Use Case

The mission and the operational scenario (or Mission Profile) shall not be mixed up.

5.3.1. Mission

The specific task, duty or function defined to be accomplished by a system

5.3.1. Use Case

A use case defines the interactions between external actors and the system under consideration to accomplish a goal.

The system use case details *what* the system will do in response to an actor's actions

5.3.2. Operational Scenario (or Mission Profile), operational modes and situations

The sequencing of the System's states during a mission and in operation
A scenario is a dynamic chain or sequence of use cases
It represents a sequence of events expected during the operation of the system. It includes the environmental conditions and expected results; in particular, the expected outputs to the inputs variation.
One system may have several mission profiles (or operational scenario)

<u>Example :</u>
For C3 as the system of interest,

- Mission of the C3 is the nitrification, ie transform ammonium into nitrate in the liquid phase.
- Operational Scenario: in nominal mode, the system is filled with medium from influent to bioreactor. Ammonium is transformed to nitrate. The bioreactor is harvested in a continuous way.
- USE CASE: has to cope with ammonium resulting from wastes of 1 human , urine is to be treated

5.4. Introduction to IVVQ activities

IVVQ is the acronym for Integration, Verification, Validation and Qualification
IVVQ activities shall demonstrate that the system is conformed to the specified requirements and address the specified needs

Hereafter the definitions:

5.4.1. Integration

Progressive Integration of components following the integration plan

5.4.2. Verification

Confirmation through the provision of objective evidence, that specified requirements have been fulfilled.
The solution satisfies all the specified requirements

5.4.3. Validation

Confirmation through the provision of objective evidence, that the requirements for a specific intended use have been fulfilled.

5.4.4. Qualification

Demonstration of the satisfaction of user needs in operational conditions.

Each requirement must be verifiable. It is mandatory to foresee the means of verification (inspection, analysis, test, simulation …) and associated criteria value when writing a requirement.
If the verification is seen at the requirement level, we can consider the validation at the needs level.

In the MELiSSA Data Management System, both notions (validation and verification) are mixed in a single word, Validation.

6. Needs, Requirements, and associated definitions

This chapter is devoted to the writing of requirements.
First, the main definitions are given. The requirement tree, compliant with the MELiSSA Data Management System is detailed.
Moreover, some rules to write a "good" requirement are provided.
The check-list and the naming rules (for DMS compatibility) complete this chapter.

6.1. Introduction

Requirements are the foundation of the project. They form the basis for design, manufacture, test and operations.
It is essential that a set of requirements be established early. Changes in requirements later in the project can have a significant cost impact.
The objectives of requirements analysis are:
- To identify and express verifiable requirements that state needs. It is an iterative activity in which new requirements are identified and constantly refined as the concept develops and additional details become known.
- To obtain a balanced set of requirements based on user needs. It is always a trade-off activity.

Therefore, requirements definition is a process that works "top-down" and "bottom-up".

6.2. Definitions

The requirement definition and other associated concepts as stakeholder, needs or constraints are explained hereafter.

6.2.1. Stakeholder (ISO15288)

A party having a right, share or claim in a system or in its possession of characteristics that meet that party’s needs and expectations.

Comments for stakeholder:

- Can be an individual or an organization
- Stakeholder examples grouped in 2 categories :
  - Stakeholder interacting directly with the system :
    - User
    - Operator
    - Maintenance operator
  - Other stakeholders concerned by, or involved in, the system:
    - Investor
    - Acquirer
    - Contracting authority
    - Prime contractor
    - Developer
    - Enterprise

We can also define the needs, which are at the stakeholder level and which are the input for the requirement definition, as shown in the following figure:

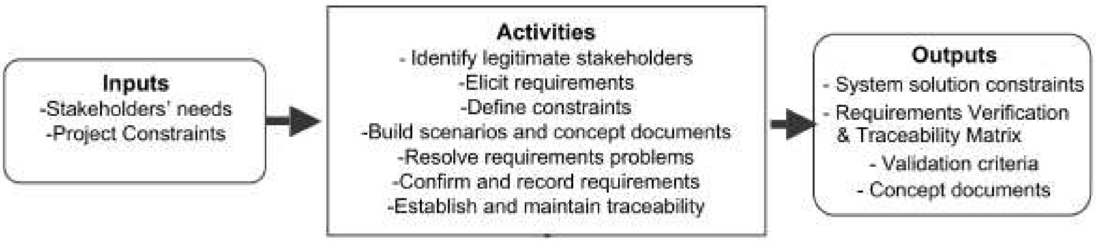


Figure 5: Stakeholders' needs

6.2.2. Needs

Necessity or desire felt by a user

The above definition concerns the “initial needs”. Needs can be associated to each stakeholder.
An addressed need is translated into one or more requirements.

6.2.3. Requirement

A statement that identifies a system, product or process' characteristic or constraint, which is unambiguous, clear, unique, consistent, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability.

At this time it is useful to define also the stakeholder.

6.2.4. Specification

Requirements are grouped in a document called **requirement specification**.
Remark: There are different levels of requirements and associated specification (see chapter 8)
A document that prescribes completely, precisely, and verifiably the requirements, design, behaviour or characteristics of a system.

It is also necessary to define a specific class of requirement which is the constraint.

6.2.5. Constraint

A type of requirement that cannot be traded off. It is a restriction imposed on the system.

6.2.6. Criteria

When writing a requirement, it is also important to define the criteria for its validation (or verification).
On contrary of the needs which can be qualitative only, each requirement has to be quantified by **criteria values** which can have some tolerance.
These criteria will be proved by some testing, simulation/modelling, inspection (e.g. visual inspection, non-destructive control) or review.

Another term must be defined as it will be used for the requirement elaboration: the justification.

6.2.7. Justification

A Justification file can be produced (see RD[5], Annex O). Each requirement has to be legitimated; justification can be achieved by different means:
- Results of trade-offs
- Link with a requirement of the level above
- Existing knowledge….

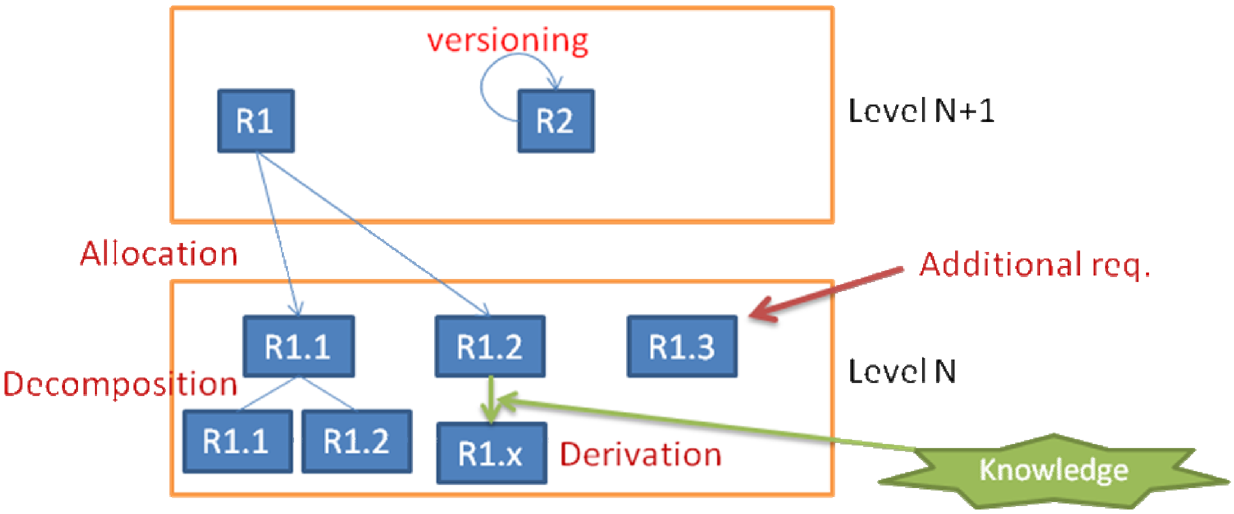
The aim of this document is to trace the origin of the requirements.

7. Requirement tree

This chapter describes the process of elaboration of the requirements.

7.1. Overview of a requirement tree

Requirements are either derived or decomposed from **Needs**



7.1.1. Allocation

Once the top-level set of system requirements has been established, it is necessary to allocate and flow them down to successively lower level.
Allocation is the top-down approach and ensures the traceability (traceability matrix can ne produced in the MELiSSA Data Management tool).

7.1.2. Derived Requirement

Detailed characteristics of the system-of interest that typically are identified during elicitation of stakeholder requirements, requirements analysis, trade studies or validation
Sum of the derived requirements is more than the initial requirement.
<u>Example</u>:
Req1: High and Low operational temperatures shall be set individually.
Derived requirement: Low temperature shall be set in a range of 1°C

7.1.3. Decomposed Requirement

The decomposition process represents a refinement of need or source requirement.
Sum of all these requirements represents the initial need.
<u>Example</u>:
Req1: system shall operate at -40°C and +70°C
Decomposed in
Req1_A: system shall operate ay low temperature (-40°C)
Req1_B: system shall operate at high temperature (+70°C)

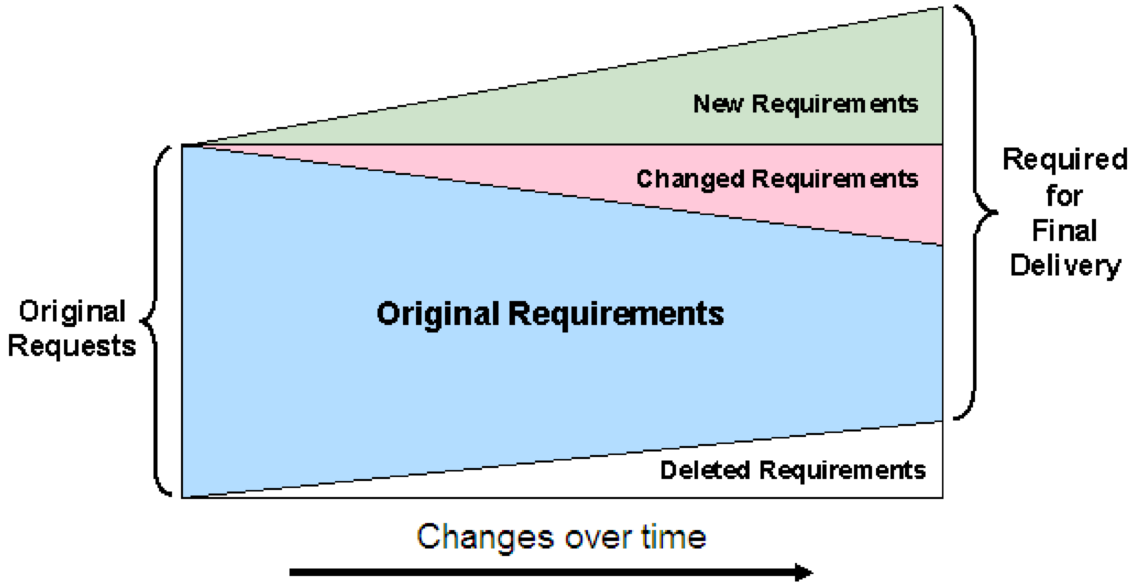
7.1.4. Additional Requirement

A new requirement can appear at a level. For example it can be a new constraint.

7.1.5. Versioning

At each level the requirement (as it is an object of the MELiSSA Data Management System) has its own life cycle (**versioning**)

Requirements changes are inevitable.



7.2. How to write properly a requirement

Source Documents

The database must first be populated with the source documents that provide the basis for the total set of system requirements that will govern its design.
Source documents will include:

- User needs
- Customer requirements document
- …

Good Requirement

As mentioned in the definition of a requirement, it must be unambiguous, clear, unique, consistent, stand-alone (not grouped), and verifiable.
Moreover it must be:

- Traceable to an identified source (e.g. document, parent requirement…)
- Not redundant and not in conflict with any other requirement

Requirements must be written with extreme care. The language used must be clear, exact, and in sufficient detail to meet all reasonable interpretations.

In the sentence of the requirement the best is to use "**shall**". The imperative form shall be used in identifying the requirement.
Moreover, the statement is stated positively (as opposed to "shall not").

For information, "will" is mainly interpreted as giving information but not stating a requirement.

All the words with uncertainty should be avoided in the requirements:

- Pronouns. Words such as "he", "they", "it can lead to ambiguity or confusion.
- Adjectives and adverbs. "Many", "few", "real-time" … should be avoided.
- "Etc", "and so on" or any unbounded list.

If used **TBD** (To Be Defined or To Be Determined) is an official word for uncertainty and should be logged in the specification document.

Examples of requirements:

- The hardware shall have a mass of 100 kg plus or minus 5 %
- The system shall operate at a temperature of 55°C plus or minus 0.5°C in nominal operation

Some traps to be avoided

- **Trap1**: The operator shall be able to calibrate each sensors in less than 1 hour

This sentence is an operational statement ant not a requirement. The requirement refers to WHAT the product must satisfy and not an activity involving the product.

The good requirement can be:

- The sensors of the system shall be calibrated in less than 1 hour.

- **Trap2**: The valve for temperature control shall be a TT1, reference X1Y1Z1

This sentence is not free of implementation. Requirements should state WHAT is needed and NOT HOW to provide it. If you ask: "why do you need this requirement", you can get the real requirement.

So, the good requirement can be

- Temperature shall be controlled in a continuous mode.

7.3. Check-list

The following questions should be considered for every requirement:

1/ **Is each requirement clear and unambiguous?**

You can check:
- are all the terms defined (glossary)?
- Is it possible to interpret the requirement in multiple ways?
- Does the requirement conflict or contradict another requirement?
- Do the requirements express one though per statement?

2/ **Is each requirement a proper requirement?**

- The requirement should specify WHAT is to be done at that level and NOT HOW it is to be done at that level

3/ **Is the requirement necessary?**

Unnecessary requirements are:
- Unnecessary specification of design which should be left to the discretion of the designer
- Redundant requirement covered in some other combination of requirements

4/ **Is each requirement achievable?**

- The expertise of potential subcontractors can help ensure achievable requirements.

5/ **Do the requirement pass the traceability test?**

- Do all requirements trace to the higher level specification? Are all the higher level requirements allocated?

6/ **Is each requirement verifiable?**

- Each requirement must be verified (test, demonstration, analysis, inspection)
- If multiple tests are required, the requirement must be broken into multiple requirements
- There is no problem with one test verifying multiple requirements

  - Are the requirements free of unverifiable terms (e.g., easy, fast, appropriate, small, robust, minimize, quickly or any other "ly" or "ize" words)

And for specific requirements:

- Functionality
  - Are all the described functions necessary and sufficient to fulfil the system mission?

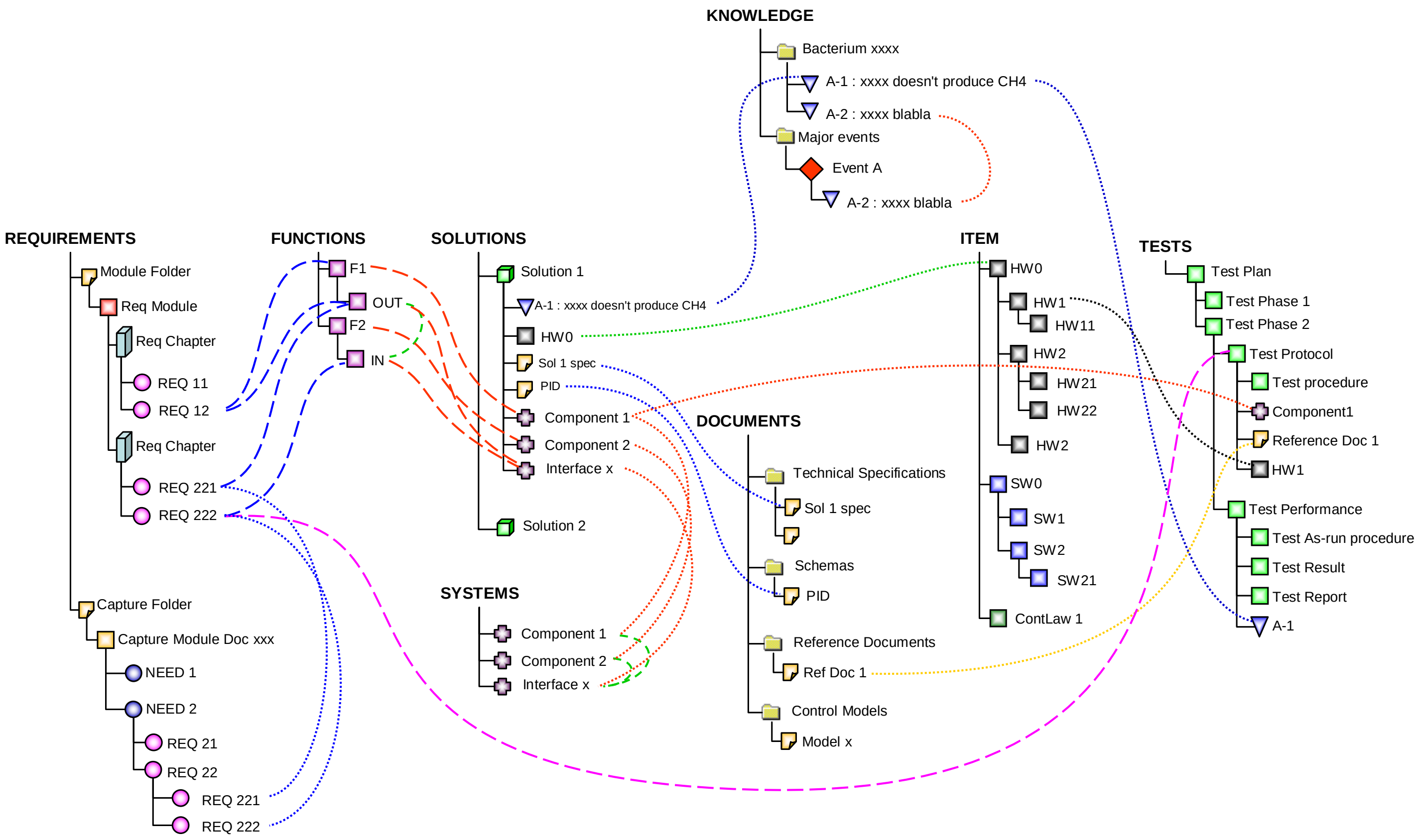
- Performance
  - Are all required performances specifications and margins listed?

- Interfaces
  - Are all external interfaces clearly defined?
  - Are all internal interfaces clearly defined?
  - Are all interfaces necessary, sufficient and consistent with each other?

7.4. Implementation in the MELiSSA Data Management tool

REQUIREMENTS class is one of the objects of the MELiSSA Data Management tool.
REQUIREMENTS are linked to the other objects of the database. Mainly:

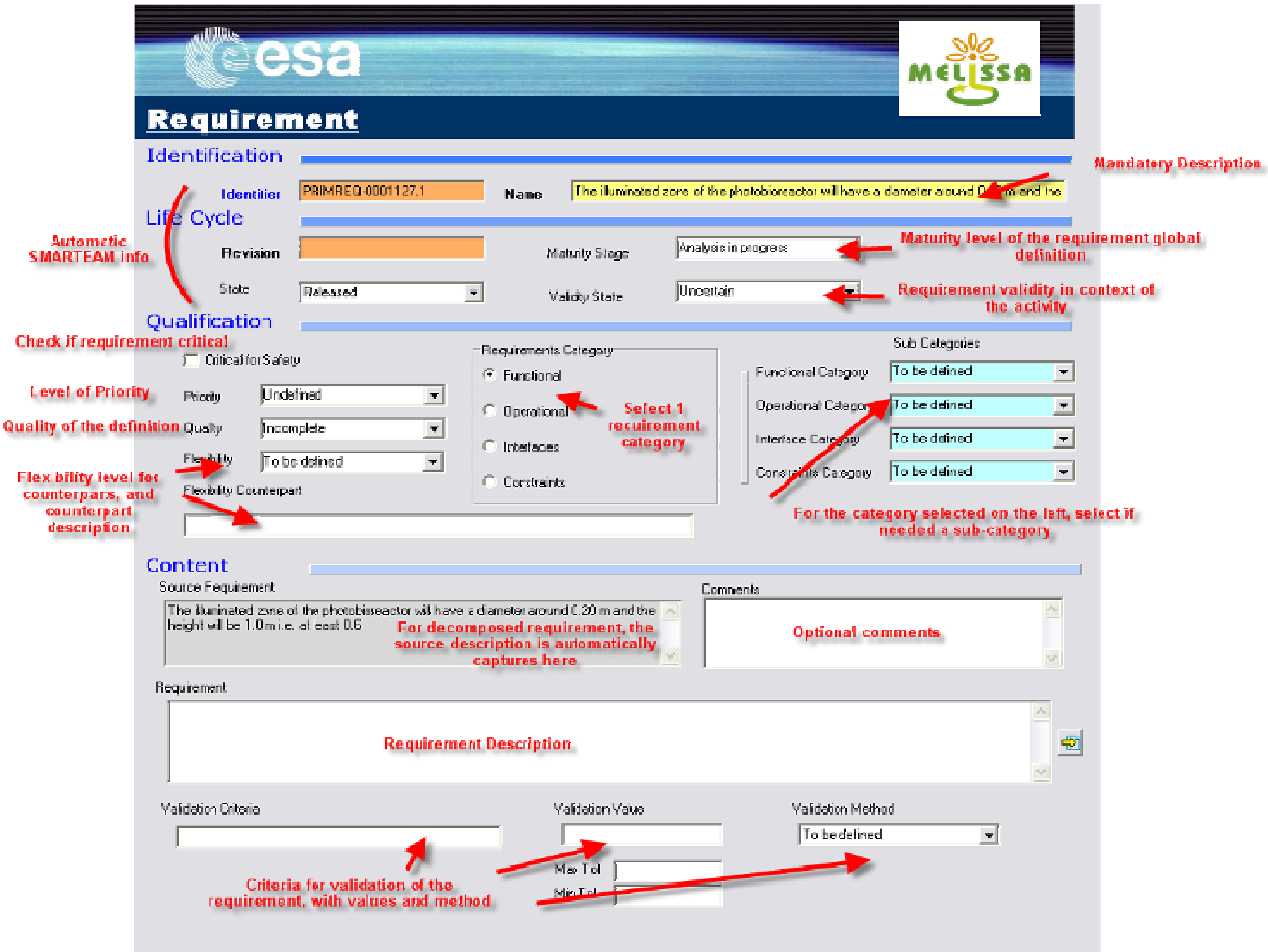
- DOCUMENTS class, e.g.
  - For the specification document
  - For the justification file
  - For other requirements sources documents
- FUNCTIONS class
  - This class is the link between the REQUIREMENTS and the SOLUTIONS. The Functions are linked to the item (HW).
- TESTS class
  - For the verification (or validation) of the requirement



Definition of the **SOLUTION** :
A solution is a set of information composed of the system architecture that constitutes the solution, the specification documents, that describes the required characteristics of the system/sub-system/components involved in the

architecture, plus references to the assumptions made, the requirements addressed, the designed functional architecture and the Hardware/Software/Control models configuration implemented.

7.4.1. Requirement Class



Justification of the requirement shall be indicated in the “Comments” cell.

7.4.2. Categories of requirements

Four categories of requirements are handled in the MELiSSA Data Management System

- Functional
- Operational
- Interface
- Constraint

Each category is divided into sub-categories.

7.4.2.1. *Functional*

- Functional/Performance
  - What are the realisations or transformations of the system, and their application domain. "Performance" aspects and "Functional" aspects have been mixed because it has been considered that they are intimately linked, and that there is no need to specify generic functions without assigned performance

- Control
  - used specifically for "Control" actions

7.4.2.2. *Operational*

- Modes & scenarios :
  - Describes the main operating situation: nominal, degraded, emergency, maintenance, stand by …). Associated operational scenarios may be described

- Operational Environment :
  - Specific requirements depending of environment (heat resistance…)

- Reliability/Availability/Maintainability/Safety (RAMS) :
  - Specific requirements for dependability, safe functioning. Include material, software, people

- Human factors & ergonomics :
  - describes operational relations between human & system (HMI, ease of use, training..)

- Resources :
  - describes what the system consumes or produces

- Maintenance :
  - Describes how the stakeholders expect in terms of maintenance tasks (where? When? How often...). NB: not to be confused with "Maintenance" constraint (see below) (eg include calibration needs)

- Transportation & Storage :
  - specific requirement for these particular life situations

- Documentation :
  - documentation being part of the system in operation, specific requirements may occur in term of presentation, format, quantity, quality, medium, etc

7.4.2.3. *Interface*

- Functional Interface
  - Specifies the « flows »

- Physical Interface
  - Specifies the physical components

7.4.2.4. *Constraint*

A constraint differs from a requirement in the sense that it restrains the scope of the possible solutions. It doesn't describe the "problem", but imposes conditions that the "solutions" will have to comply with. Many types of constraints may occur.

- Physical constraints
  - Mass, volume, surface, material…

- Design constraints
  - Solution architecture, technology…

- Production constraints
  - Means of production, site…

- Maintenance constraints
  - maintenance means, standard to be applied, etc… e.g. access needed for calibration

- Logistics constraints
  - means, frequency…

- <u>Commissioning/Retirement constraints</u>
  - how to put the system in service and how to dismantle or destroy it at the end of its life

- <u>Cost & Delay constraints</u>

- <u>Quality constraints</u>

- <u>Regulation & Norms constraints</u>
  - references of the texts to comply with

- <u>IVVQ Process constraints</u>
  - constraints regarding the integration, the verification, the validation and qualification processes

7.5. Naming rules and attributes

The hierarchy of the requirement listing is organized in a way to be as close as possible to the logic of the MELiSSA Data Management System. Each requirement obtains a unique identifier which satisfies the syntactic rules given below:

Req.id = <Req level >-<Req.category>-<Req.subcategory>-<Req.number>

Requirement level: e.g. SYS for system level
Requirement category: see 3.6
Requirement subcategory: see 3.6
Requirement number: consecutive numbering

All the naming rules ensure that a requirement identifier is unique.

The following table gives the naming rules for each category and sub-category.

Category	Sub-Category	Naming Rule
Functional		FUN
	Functional/Performance	PERF
	Control	CTRL
Operational		OP
	Modes & Scenarios	MODE
	Operational Environment	ENV
	RAMS	RAMS
	Human factors & ergonomics	HUMAN
	Resources	RESS
	Maintenance	MAINT
	Transportation & Storage	TRANS
	Documentation	DOC
Interface		INT
	Functional Interface	FUN
	Physical Interface	PHY
Constraint		CON
	Physical	PHY
	Design	DES
	Production	PROD
	Maintenance	MAINT
	Logistics	LOG
	Commissioning/Retirement	COM
	Cost & Delay	COST
	Quality	QUAL
	Regulation & Norms	NORM
	IVVQ process	IVVQ

Ex : C3-FUN-PERF-001 is a requirement for the system of interest C3, category functional, sub-category performance.

Moreover, each requirement is qualified with the following attributes:

7.5.1. Priority

It is the ranking of the requirement. It can be:

- High, Medium, Low, Undefined

7.5.2. Quality

Attribute from the user indicating the quality of the requirement. It can be:

- Complete, Incomplete, Vague, Un-checkable

7.5.3. Flexibility

This attribute gives the flexibility level of the requirement. It can be:

- Mandatory, Barely negotiable, negotiable if counterparts, negotiable.

An another attribute (field) giving the counterparts description shall be filled if “negotiable if counterparts” is selected.

7.5.4. Critical for Safety

Can the requirement have an impact on the system safety for the operator:

- Yes/No

The user is also invited to define the following attributes when writing a requirement:

7.5.5. Maturity

It is the maturity level of the requirement. It can be:

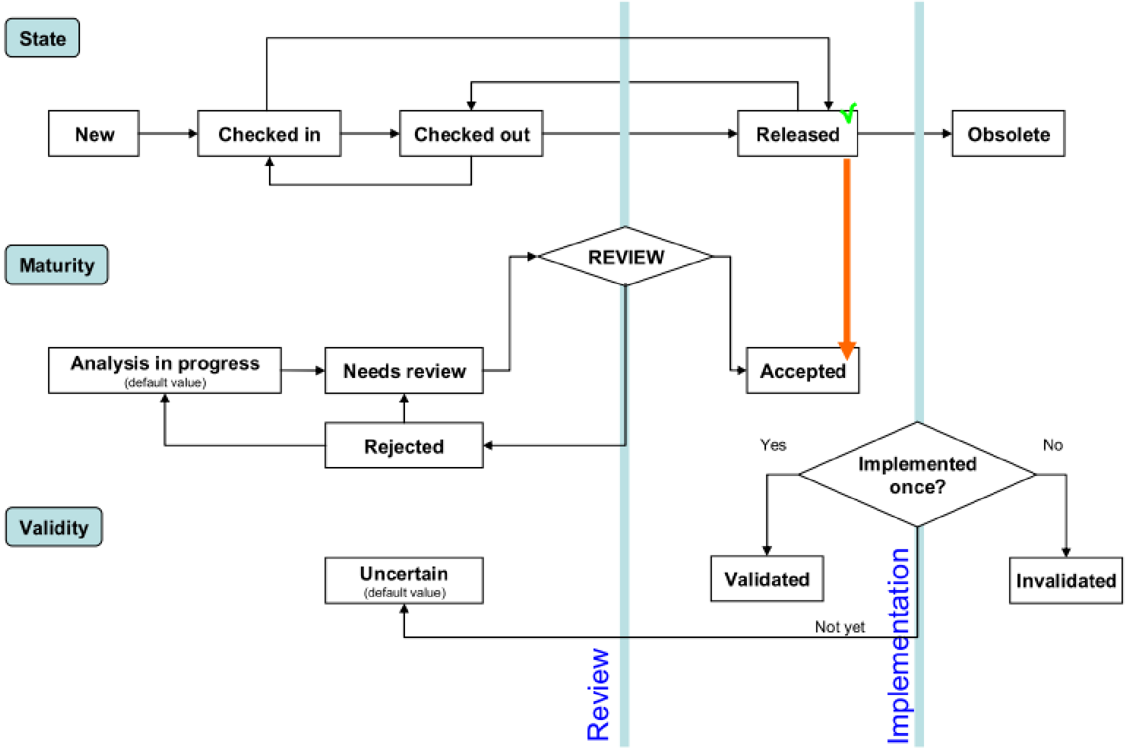
- Analysis in progress, need review, accepted, rejected

7.5.6. Validity

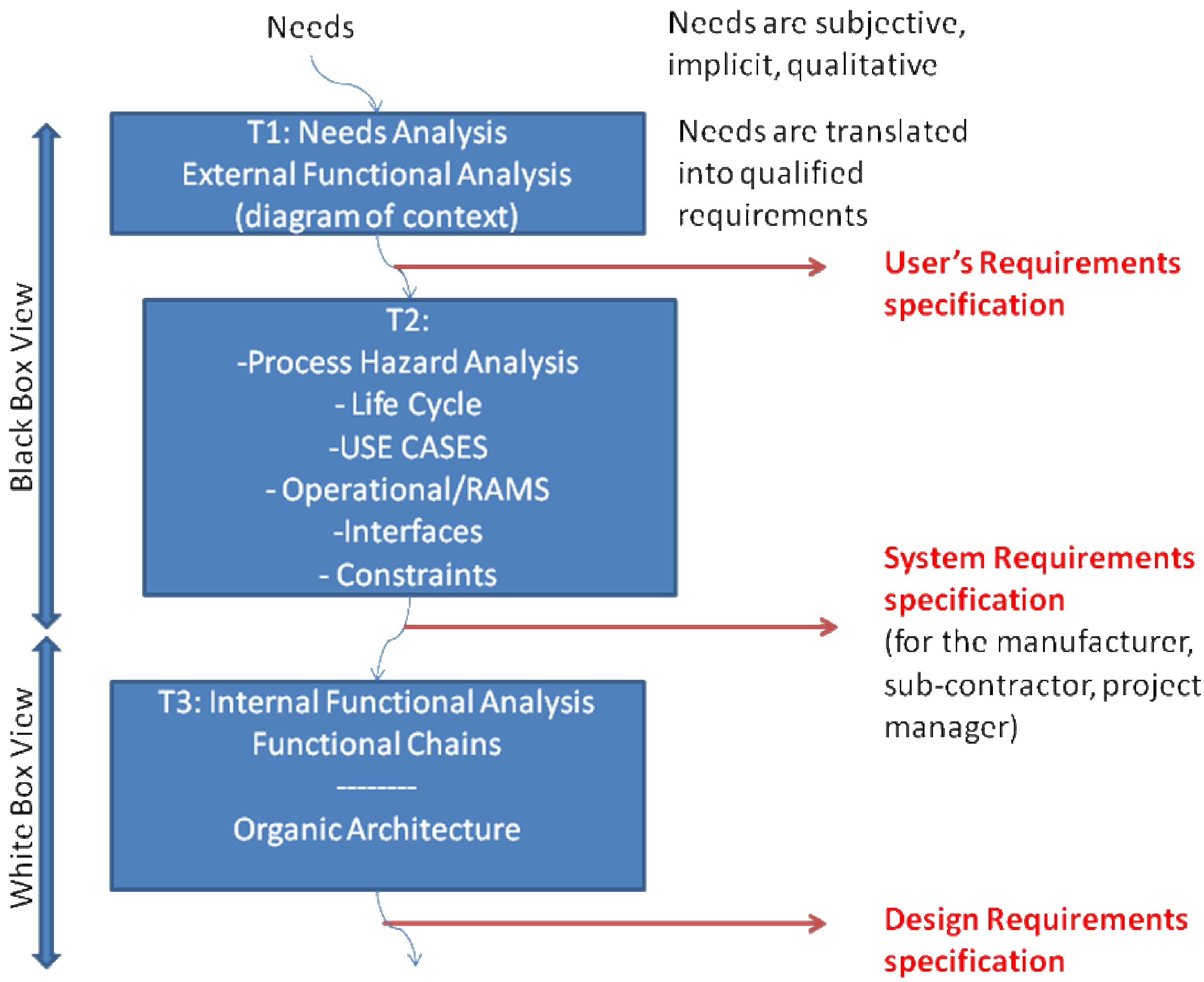
It is the requirement validity in context of the activity. It can be:

- Validated, Uncertain, Invalidated.

In addition to the normal life cycle (**State**) of the object requirement, the maturity and validity are linked with the following process:


State
New
Checked in
Checked out
Released
Obsolete
Maturity
REVIEW
Analysis in progress
(default value)
Needs review
Accepted
Rejected
Yes
Implemented once?
No
Validity
Validated
Invalidated
Uncertain
(default value)
Not yet
Review
Implementation

8. Methodology for the elaboration of specification documents



Task1: Identification of the functions (external point of view, black box view)
Translation of the identified needs into requirements through Needs Analysis
→ **User's requirements specification**

Taks2: Includes life cycle phase, use cases, operational, rams, interfaces, constraints

➔ **System Requirements specification**

White Box view, identification of the chains of functions (internal functional analysis, organic architecture

➔ Design requirements **specification**

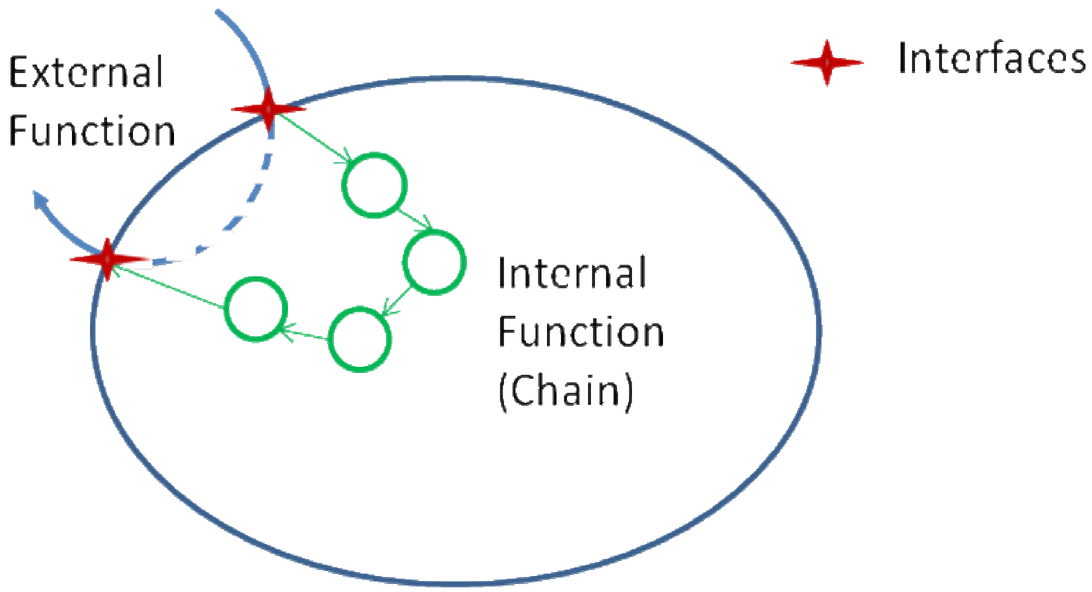


Figure 6: From the external function to the internal function

9. Template TN document (Draft)

See the TN template for requirement specification for details.

1. Introduction
2. Scope
    a. System of interest
    b. Mission of the system
    c. Stakeholders
    d. Sub-Systems
3. Context Diagram
    a. Context Diagram
    b. Actors
    c. Inputs
    d. Outputs
    e. Life Cycle, Scenario, Use Case
4. Needs
5. Requirements
    a. Meta Level
    b. Upper Level
    c. System of Interest Level
        i. Functional Requirements
        ii. Operational Requirements
        iii. Interface Requirements
        iv. Constraint Requirements

Requirements are classified following the 3.6 and 3.7 categories and naming rules.

10. List of documents & reviews

User's requirements

System requirements

Design requirements

SRR : System Requirement Review