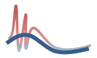
Systems Architecture Motivations

Take Away

Purpose Anticipate design risks during the early phases of the project to minimize systemic problems and thus save time and cost

Main categories of systemic problems



Modeling problems

Model and reality do not match

Problem type: the system design is based on a model which does not match with reality

Initial choice in design phase with late unexpected consequences

Problem type: the impact of a wrong design choice appears late in a system life-cycle

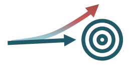
Integration problems

The robustness of a system is destroyed by a "domino effect"

Problem type: a local problem spreads step by step and has global consequences

The system has undesirable emergent properties

Problem type: an integrated system has unexpected or undesired emerging properties



Project problems

The project system has integration issues

Problem type: the engineering of the system is not done in a collaborative way

The mission of the product is diverted by the project system

Problem type: the project forgets the mission of the product and « indulges» itself

Key concepts

A **system** is a set of interrelated **components** (covering hardware, software and humanware) working together toward some **common mission.**

Every product or system that we develop is always used as part of a **larger system**

Every project can benefit from **good systems architecture.**

Systems architecture is **not just for large** complex "solution" projects.

A mouse is a system

… so is an aircraft!

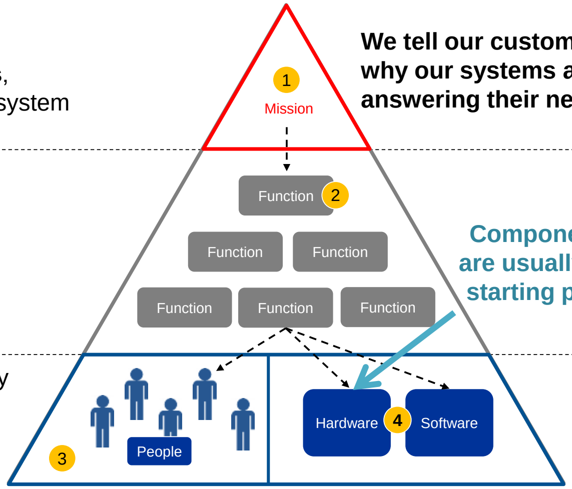
Key principle

EVOLVING FROM A TECHNOLOGY-ORIENTED OR BOTTOM-UP APPROACH …

… and ends with customers, **hopefully satisfied** by the system that we are proposing them

… which have certain functional behaviors …

System design is traditionally a **bottom-up** process that starts with technology and possibly people …



We tell our customers why our systems are answering their needs

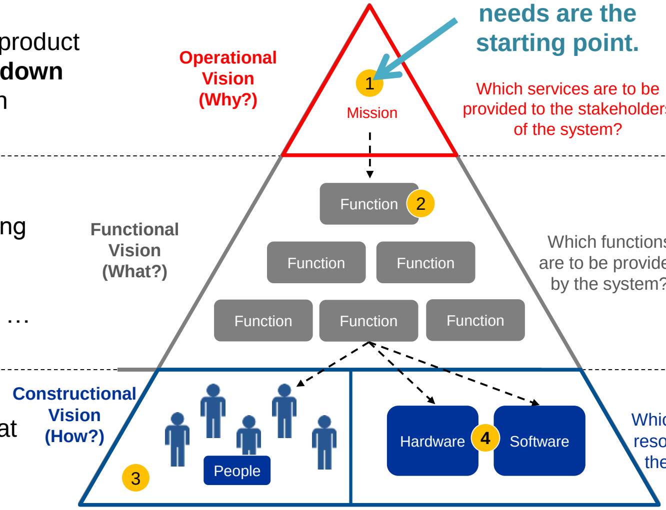
Components are usually the starting point

… TO A CUSTOMER-FOCUSED AND TOP-DOWN SYSTEM STRATEGY

In a MBSE approach, product design is rather a **top-down** process that starts with stakeholders …

… continues by defining the functions that the system shall offer to answer to their needs …

… and ends with the concrete resources that **effectively answer their needs**



Stakeholders and needs are the starting point.

Which services are to be provided to the stakeholders of the system?

Which functions are to be provided by the system?

Which concrete resources form the system?

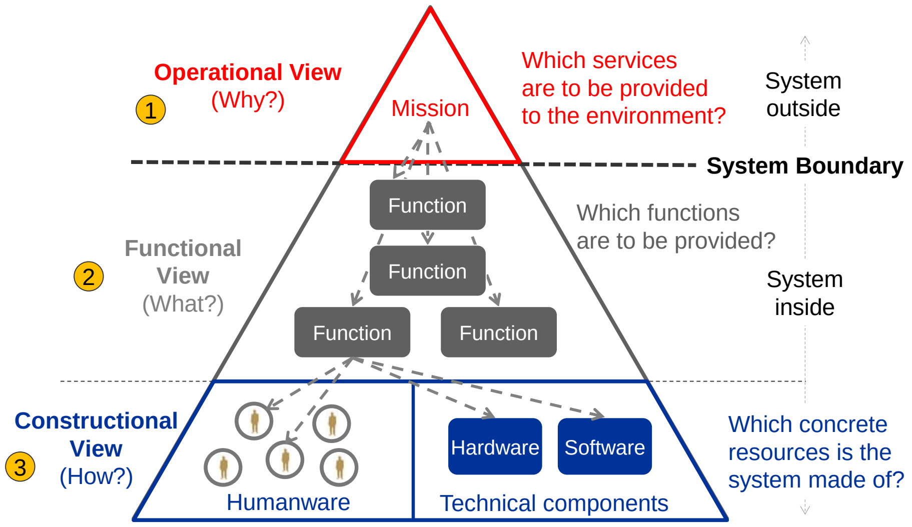
Systems Architecture framework

Take Away

Definition

System architecture is the discipline synthesizing the methods and the tools which allow an exhaustive & coherent modelling of a system (in its triple operational, functional & constructional dimensions) in order to manage it efficiently during its lifecycle (design, test, deployment, maintenance, …).

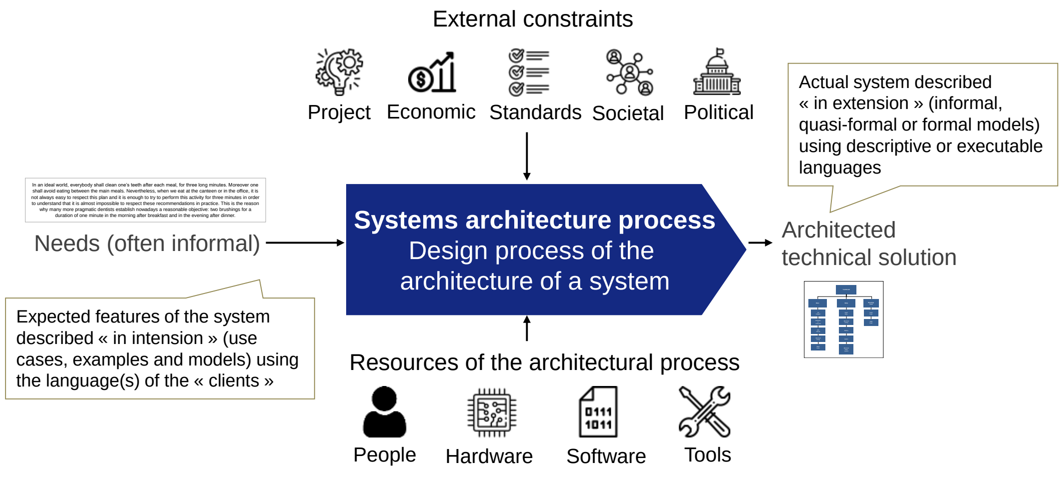
Framework



Architectural view pyramid

- The **operational vision** of a system defines the **mission** of the system, analyzed here as a **black box** from the **external perspective** of the system **stakeholders**
- The **functional vision** of a system defines the **abstract functions** of the system, analyzed as a **grey box**, that are required to **deliver the system mission**
- The **constructional vision** of a system defines the **concrete components & building blocks** of the system, analyzed as a **white box**, that **implement the functions** of the system

Process



Key concepts

- Each **system** has a standard representation which highlights its double input/output and internal behaviour that transforms inputs into outputs depending on its internal states
- An **interface** represents an interaction, an exchange, an influence or a mutual dependence between at least two systems
- **Integration** is the process that enables to build a system based on other systems (hardware, software & humanware) that are organized in such a way that the resulting integrated system can perform – in a given environment – its mission
- A **model** is an abstract representation of a system, often organized in views according to an architectural framework

Stakeholder analysis

Take Away

Purpose

Have a comprehensive view of the external systems that impact the design of the system of interest, and define with no ambiguity the perimeter of the system.

Key concepts


User

Conformity assessment body


Toothpaste


Dentist


Electronic toothbrush

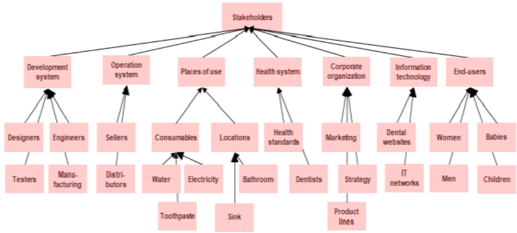

Bathroom

A **stakeholder** is an **external body that influences or interacts with the considered system.**

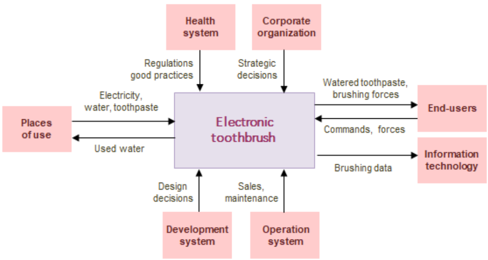
A stakeholder:

- is not necessarily a person,
- affects or is affected by the system (directly or indirectly),
- may come or not into contact with the system,
- has needs or imposes constraints relatively to the system

Deliverables

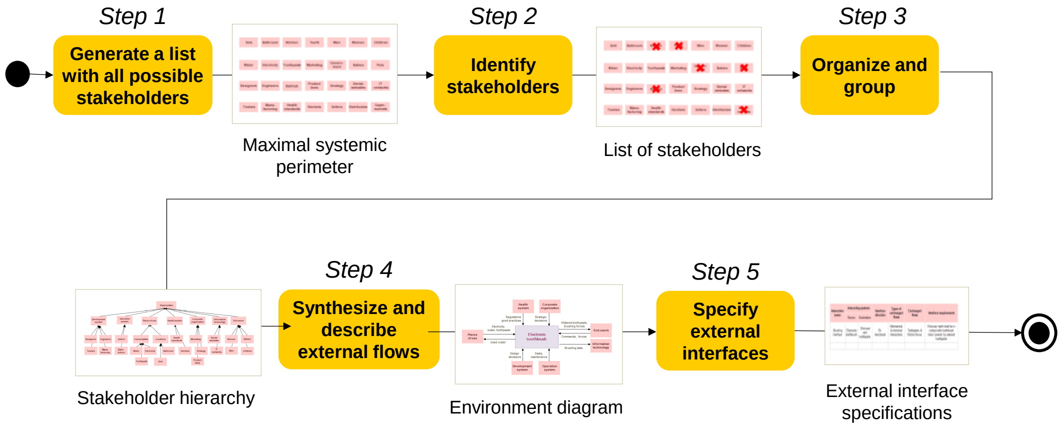


Stakeholder hierarchy



Environment diagram

Process



Key points

- Use the 7x7 rule to keep the diagrams readable
- Tell a story with a chosen Input, Output, Resource, Constraint arrangement
- Don't try to be comprehensive yet, as you will be improving your analysis from the other views!
- Your external flows should be matter, energy or information

Main cross-analyses to perform from this view

- Check that all stakeholders have needs and use cases
- Check the consistency between stakeholders and lifecycle

Needs analysis

Take Away

Purpose

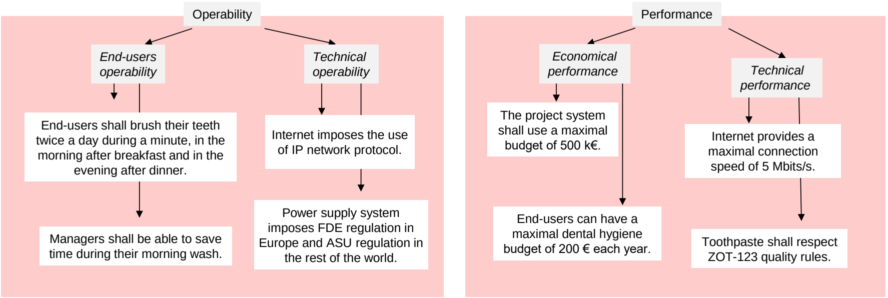
Express in an unambiguous, measurable and testable way the expectations of all external systems and characterize their expected level of performance. Needs are like a contract performance with all stakeholders for the system of interest.

Key concepts

Need			
Domain	Main category to which the need belongs	**Reference**	A unique code for the need
Statement			
Need pattern to respect The <**EXTERNAL SYSTEM**> (who) shall be able / imposes <**TO DO / TO HAVE SOMETHING**> (what or how) with an <**EXPECTED LEVEL OF PERFORMANCE**> (how much) in a <**LIFECYCLE PHASE**> (when and/or where).			
Satisfaction criteria			
How does one measure and quantify – from the perspective of the stakeholder – that the need is really fulfilled?			

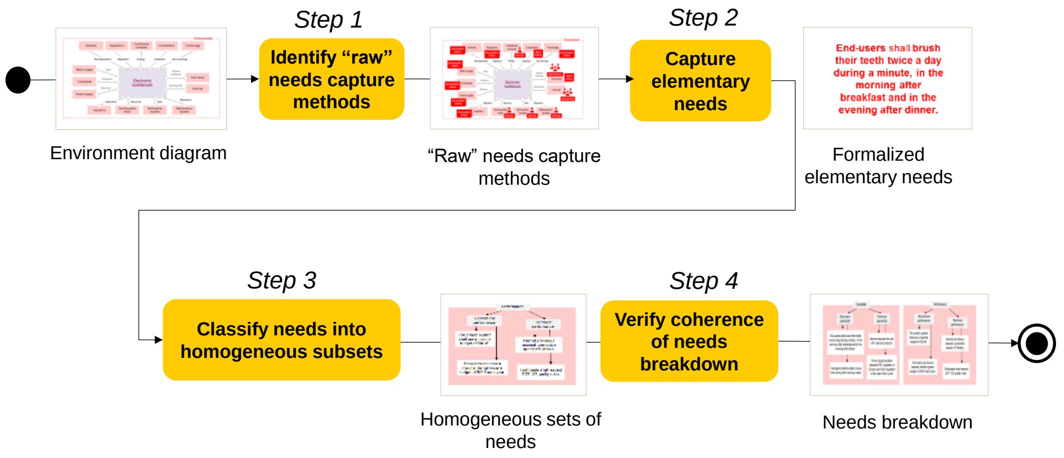
A **need** relative to a **system** is a **feature**, **expected** or **imposed** by **one or more stakeholders** of its environment that has an impact on the system of interest and that is necessary to respect to be accepted by the stakeholders

Deliverables



Need breakdown

Process



Key points

- Use the 7x7 rule to keep the need breakdown understandable
- If a stakeholder expresses a need as "the system shall do", try to understand the core reasons by asking up to 5 times "why"
- You can use and adapt PESTEL and/or OAPSET frameworks to find new needs and classify them
- Always verify that needs have a real influence on the target system
- Never forget to share the needs with their stakeholders !

Main cross-analyses to perform from this view

- Check that each stakeholder has at least one need
- Check that each lifecycle phase is mentioned in at least one need
- Check that use cases are aligned with "to do/to have something"

Lifecycle analysis

Take Away

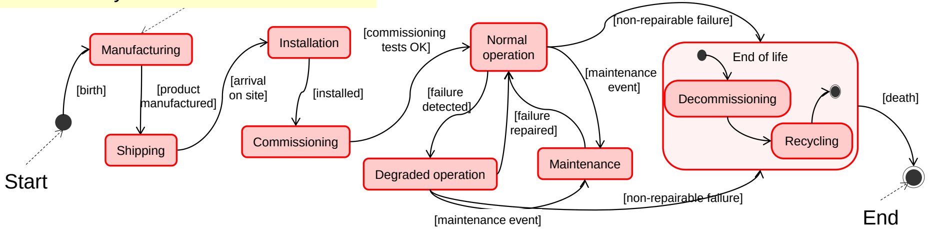
Purpose

Identify the operational contexts of the system and identify the events that allow to pass from an operational context to another

Key concepts

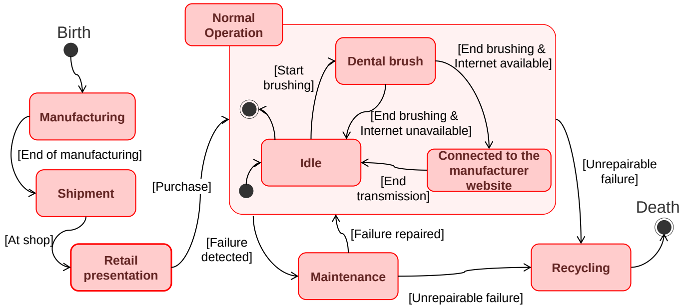
This is a **lifecycle phase**, that is to say a **period of time** (duration > 0) during which the **system environment** has some stability

This is a **transition**, that is to say an **event** (duration = 0) that makes switch from one lifecycle phase to another, and marks a major change in the system environment



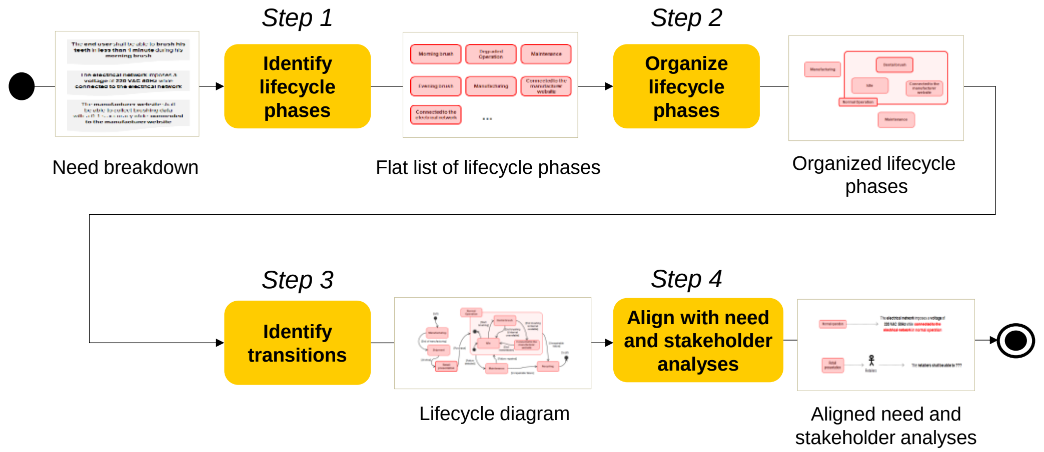
A **lifecycle phase** of a system is a **homogeneous period of time** from the perspective of the stakeholders of the system. Its **lifecycle** models the **succession of all lifecycle phases** and the **transitions** between lifecycle phases among time, from birth to death of the system.

Deliverables



Lifecycle diagram

Process



Key points

- You should zoom on the most valuable phases for your problem
- Go through the complete life of your system so that you can identify missing lifecycle phases
- Two phases can be consecutive, simultaneous or one included in the other

Main cross-analyses to perform from this view

- Check that your needs are aligned with your lifecycle phases
- Check that each phase is characterized by a stable configuration of the system environment

Use cases & Operational scenarios analyses

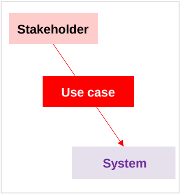
Take Away

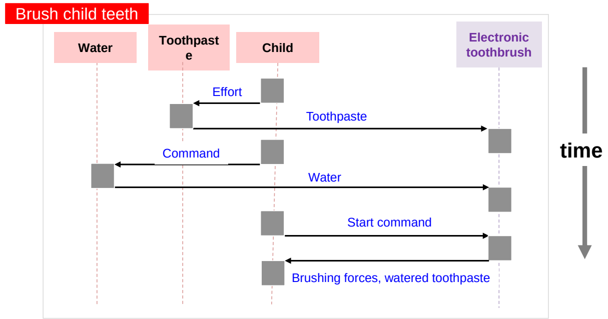
CESAMES

Purpose

Understand how the system will be used by its stakeholders and interact with its stakeholders.

Key concepts





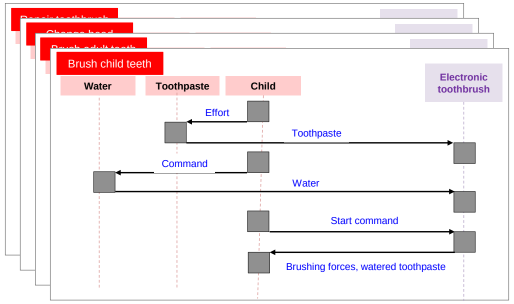
A **use case** describes an **action that can be performed** by one or several stakeholders when using the system.

A use case can be described through an **operational scenario** which specifies – using a sequence diagram – the sequence of **activities** and the external **exchanges** that take place between the system of interest, considered here as a black box, and the stakeholders during the considered use case.

Deliverables

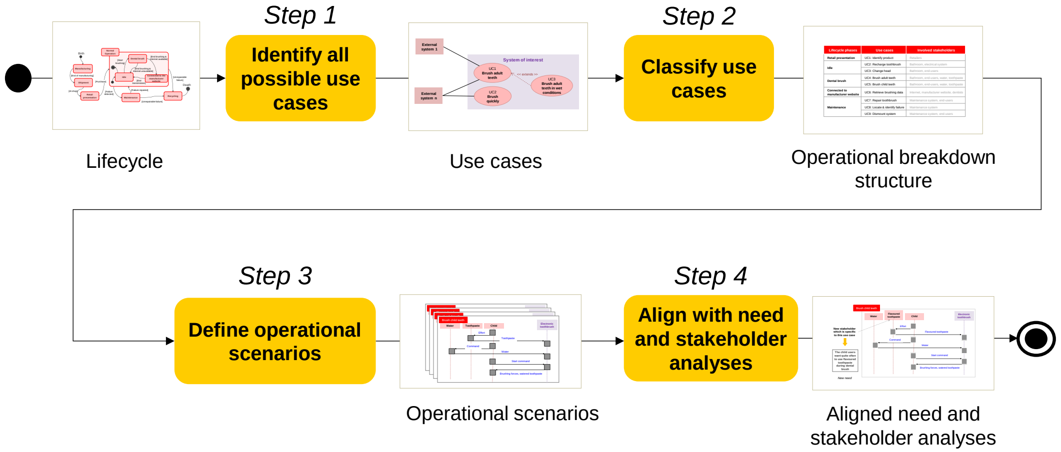
Lifecycle phases	Use cases	Involved stakeholders
Retail presentation	UC1: Identify product	Retailers
Idle	UC2: Recharge toothbrush	Bathroom, electrical system
	UC3: Change head	Bathroom, end-users
Dental brush	UC4: Brush adult teeth	Bathroom, end-users, water, toothpaste
	UC5: Brush child teeth	Bathroom, end-users, water, toothpaste
Connected to manufacturer website	UC6: Retrieve brushing data	Internet, manufacturer website, dentists
Maintenance	UC7: Repair toothbrush	Maintenance system, end-users
	UC8: Locate & identify failure	Maintenance system
	UC9: Dismount system	Maintenance system, end-users

Operational breakdown structure



Operational scenarios

Process



Key points

- A use case shall be named using the pattern “do something” where the subject is the stakeholder
- You don’t need to illustrate all your use cases with operational scenarios, select those which are the most valuable for your problem. You can also play scenarios in your head to do your cross-analyses.
- In scenarios, you can represent exchanges in parallel, alternatives and loop instructions

Main cross-analyses to perform from this view

- Check that each use case is covered by at least one need
- Check that there is at least one use case for each lifecycle phase
- Check that each external interaction in operational scenarios is consistent with the input and output flows in the environment diagram
- Check that all stakeholders identified in scenarios are in the environment diagram

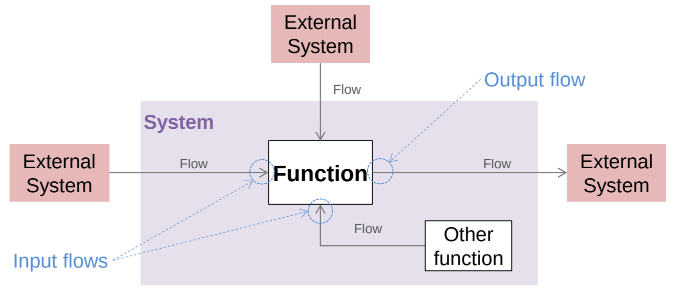
Functional analysis

Take Away

Purpose

Have a comprehensive view of the behaviour of a system.

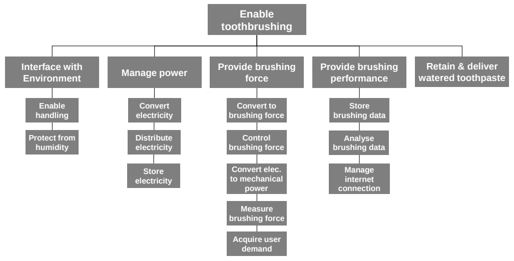
Key concepts



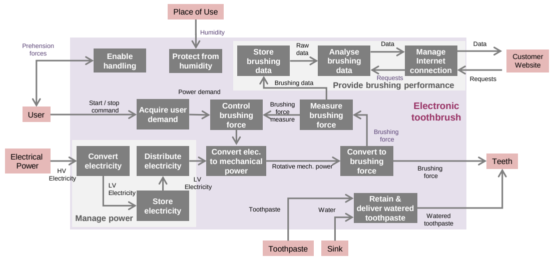
A **function** describes a **transformation** performed by the system between its **input and output flows** in order to provide an **adequate answer** to use cases

- Input flows can come from external systems or other functions of the system
- Output flows can go to external systems or other functions of the system

Deliverables

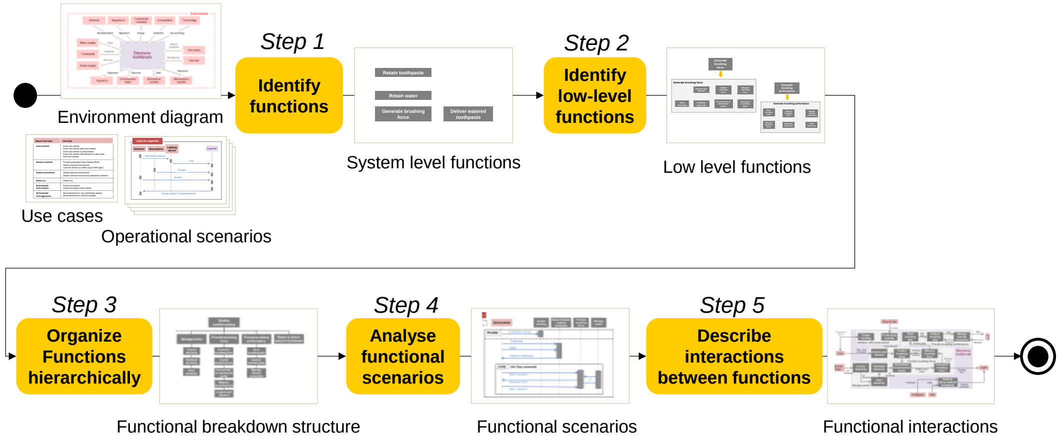


Functional breakdown structure



Functional interaction diagram

Process



Key points

- Functions shall be technology independent
- Go through each operational scenario (even if it's in your mind) to identify the functions that the system shall achieve to consume inputs and generate outputs
- Use the 7x7 rule to keep the functional breakdown structure readable
- You can stop the functional decomposition as soon as the lowest level of functions can be allocated to sub-systems of the system
- Do not mix functions with use cases ! The subject of the verb has to be the system

Main cross-analyses to perform from this view

- Check that each function is allocated to at east one use case
- Check that each low-level function can be allocated to a single component

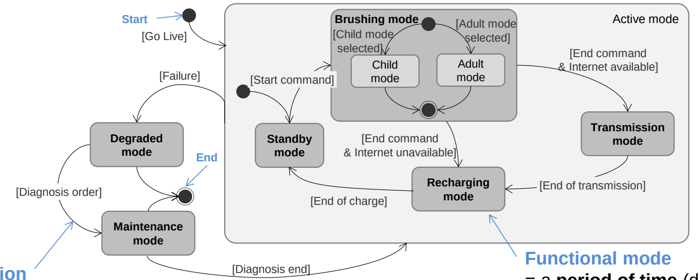
Functional mode analysis

Take Away

Purpose

Understand the evolutions of what the system should do with time.

Key concepts

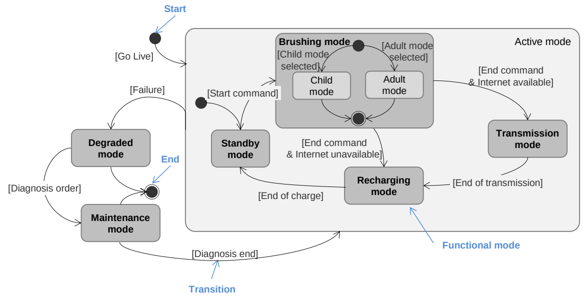


Transition
= an **event** (duration = 0) that makes switch from one functional mode to another

Functional mode
= a **period of time** (duration > 0) during which the functional behaviour is stable

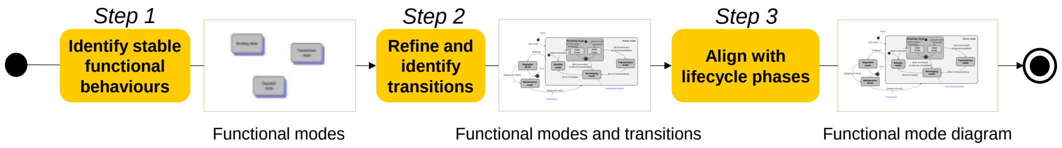
A **functional mode** of a system is a **functionally coherent period of life** of the system, i.e. a period of time which is characterized in an unambiguous way by the set of functions that the system is using during it

Deliverables



Functional mode diagram

Process



Key points

- Start from the functional interaction diagram to identify periods of time during which the functional behaviour is stable
- Do not forget to consider degraded functional modes
- To build your state machine diagram, do not hesitate to group, divide and merge functional modes

Main cross-analyses to perform from this view

- Check that all the lifecycle phases are covered with functional modes (and vice-versa). The correspondence is not necessarily 1-to-1

Functional requirement analysis

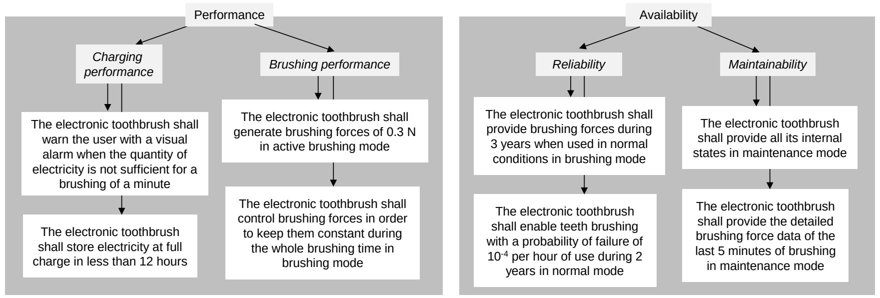
Take Away

Purpose Express in an unambiguous, measurable and testable way how the expected functions of the system answer to stakeholders’ needs and characterize the level of performances of these functions

Key concepts

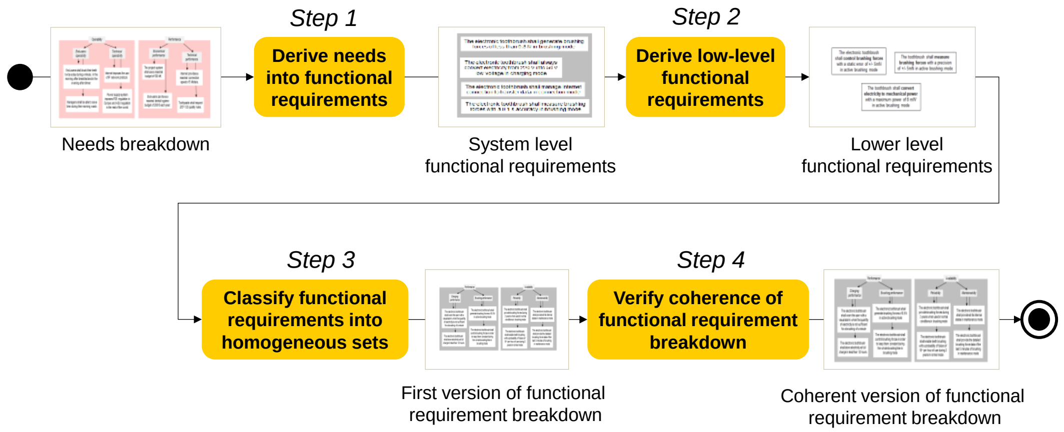
Functional Requirement			
Domain	Main category to which the requirement belongs	**Reference**	A unique code for the Requirement
Statement			
Functional requirement pattern to respect The <**SYSTEM**> (who) shall <**DO SOMETHING**> (what) with an <**EXPECTED LEVEL OF PERFORMANCE**> (how much) in a <**GIVEN FUNCTIONAL MODE**> (when and/or where).			
Satisfaction criteria			
How does one measure and quantify that the functional requirement is really fulfilled?			

Deliverables



Functional requirement breakdown

Process



Key points

- The question answered by a functional requirement is “What the system shall do?”
- Use the 7x7 rule to keep the functional requirements breakdown understandable
- You can use and adapt the OAPSET framework to classify your functional requirements
- Functional requirements define the performances of a function

Main cross-analyses to perform from this view

- Check that each functional requirement is linked to a need by a derivation relationship
- Check that your functional requirements and your functions are consistent: functional requirements can be seen as specifications of a function, alternatively, functions can also be seen as a set of consistent requirements
- Check that your functional requirements and your functional modes are consistent

Constructional analysis

Take Away

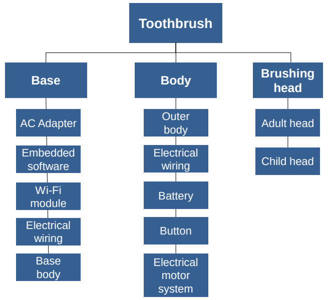
Purpose Find an optimal solution that results from a trade-off between what is desired (functions, style…) and what is possible (feasibility, technology, physical constraints,...)

Key concepts

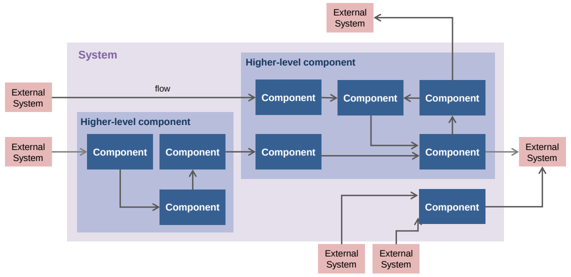
A **component** is a **concrete building block** of the system

- Components refer to the **nature of the considered part** of the system
- A component can itself be considered as a system.

Deliverables

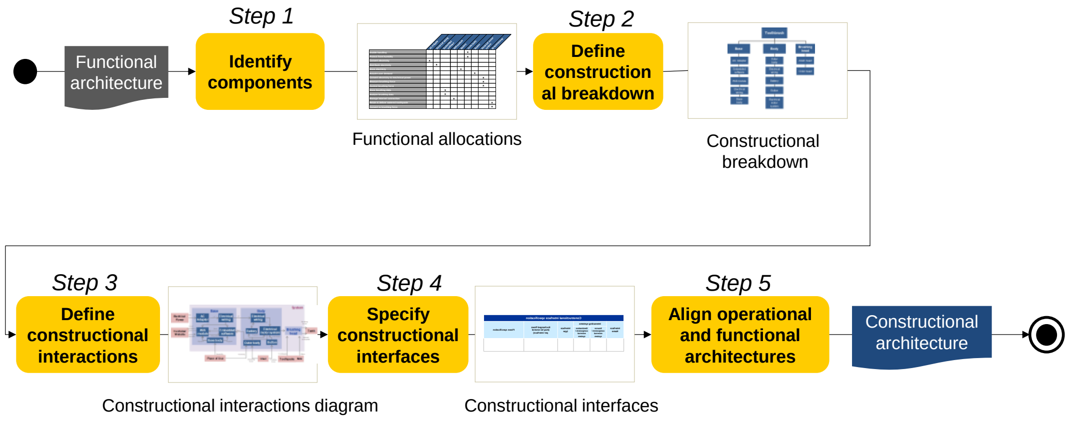


Product breakdown structure



Constructional interactions diagram

Process



Key points

- Start from low-level functions and identify the components that will implement them
- Decouple the components of the system in order to minimize their mutual interfaces
- Use the 7x7 rule to keep the constructional breakdown understandable

Main cross-analyses to perform from this view

- Check that each component is linked to either a function or a need
- Check that each low-level function is allocated to a single component
- Check the consistency between the different architectural levels following the choices you made at constructional architecture level

Constructional requirement analysis

Take Away

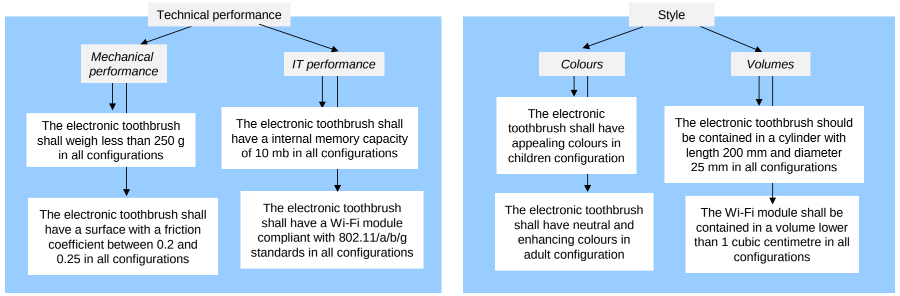
Purpose

Express in an unambiguous, measurable and testable way how the components of the system answer to stakeholders' needs and system's functions and characterize the level of performance of these components.

Key concepts

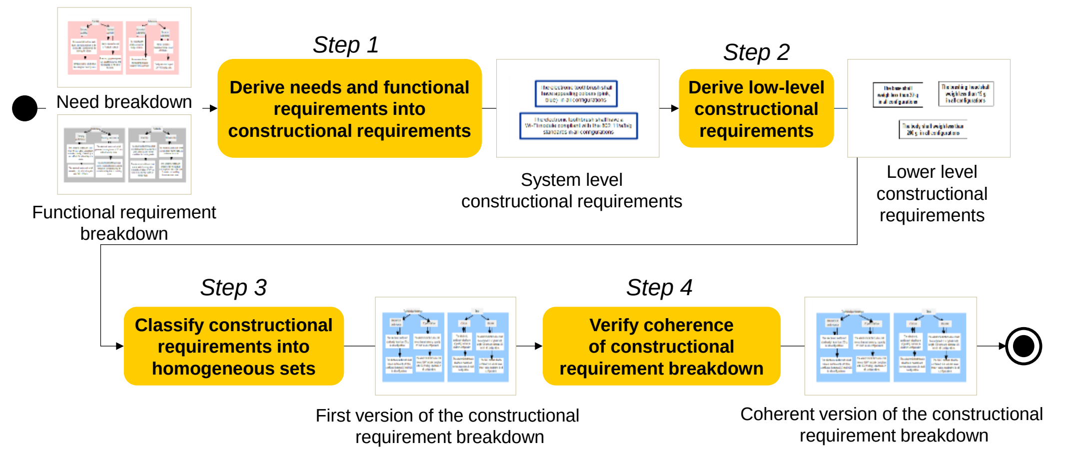
Constructional requirement			
Domain	Main category to which the requirement belongs	**Reference**	A unique code for the Requirement
Statement			
Constructional requirement pattern to respect The <**SYSTEM**> (who) shall <**BE / BE MADE OF SOMETHING**> (how) with an <**EXPECTED LEVEL OF PERFORMANCE**> (how much) in a given <**TECHNICAL CONFIGURATION**> (when and/or where).			
Satisfaction criteria			
How does one measure and quantify that the constructional requirement is really fulfilled?			

Deliverables



Constructional requirement breakdown

Process



Key points

- The question answered by a functional requirement is "What the system shall be?"
- Use the 7x7 rule to keep the constructional requirements breakdown understandable
- You can use and adapt the OAPSET framework to classify your constructional requirements
- Constructional requirements define the structural features of the concrete components that make the system

Main cross-analyses to perform from this view

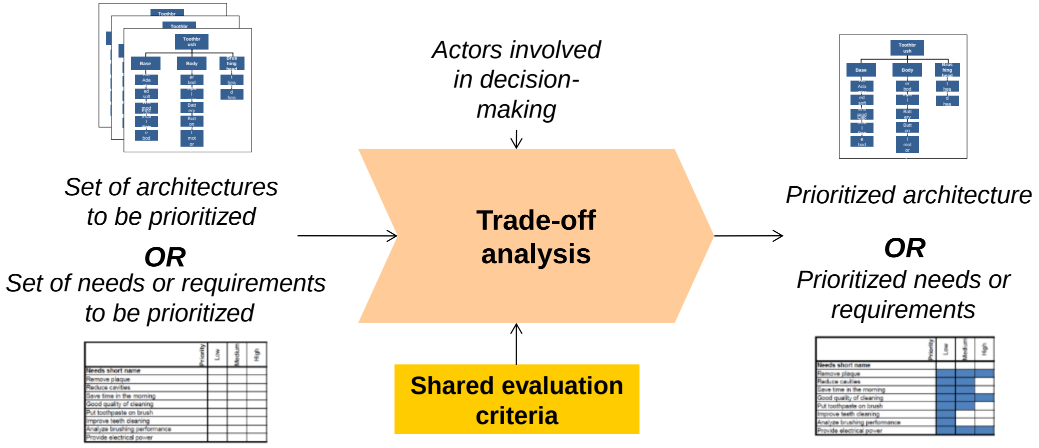
- Check that each constructional requirement is linked to a need or a functional requirement by a derivation relationship
- Check that your constructional requirements and your components are consistent
- Check that your constructional requirements and your technical configurations are consistent

Trade-off analysis

Take Away

Purpose Help the actors involved in a decision making process to prioritize an architecture or a set of needs or requirements in a rational way using shared evaluation criteria

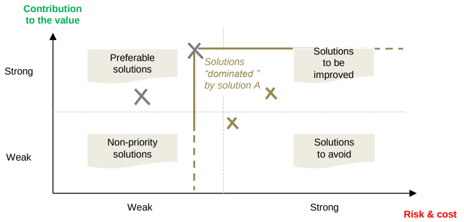
Key concepts



Deliverables

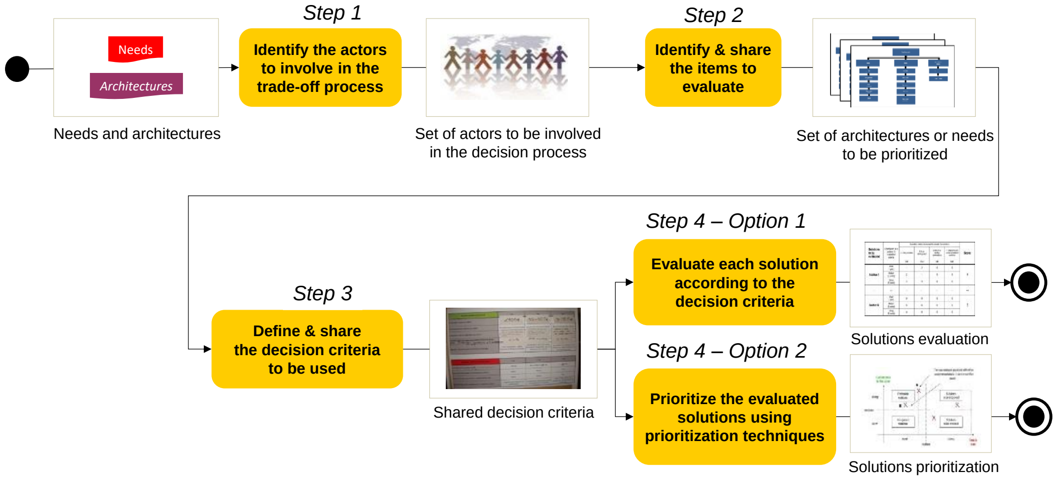
Solutions to be evaluated	Contribution of solution to decision criteria	Decision criteria to be used to evaluate the solutions					Score
		A. User friendliness	B. Short development time	C. Reuse of existing technology	D. Cost of non quality reduction	E. Improvement of services provided to customers	
		2/40	13/40	3/40	16/40	6/40	
Solution 1	Weak (1 pt.)	1	2	0	1	0	4
	Medium (3 pt.)	2	1	0	4	0	
	Strong (9 pt.)	3	3	6	1	6	
...	...	...	...	...		...	...
Solution N	Weak (1 pt.)	0	0	0	0	0	3
	Medium (3 pt.)	6	6	6	3	6	
	Strong (9 pt.)	0	0	0	3	0	

Global evaluation for each solution



Prioritization of each solution

Process



Key points

- The deliverables of a trade-off analysis are decision-helping deliverables not the decision itself !
- The good selection of actors is key to get a balanced decision. The environment diagram is a very powerful tool to help you identify them.
- To help you define decision criteria, these questions can help you:
  What are the benefits and quality, cost, delay and performance impacts of a solution ?
  What are the operational problems induced by a solution ?
  What is the technical complexity of a solution ?
- Actors need to give each criterion a weight to evaluate its relative importance.
- All actors usually do not have the same importance as well. You also need to give each actor a weight for their vote.
- Option 1 leads to a global evaluation of each solution following the evaluation criteria
- Option 2 leads to a visual comparison of each solution which enables you to compare both their positive and negative contributions

Validation and verification

Take Away

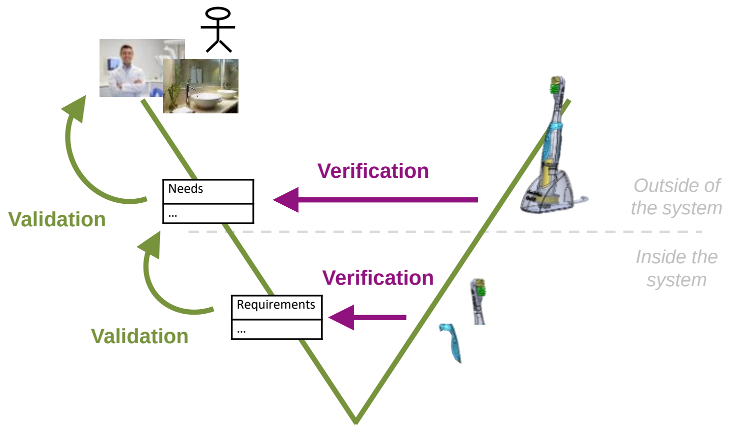
Purpose

Guarantee that the system is operationally, functionally and constructionally consistent and takes correctly into account all its expected properties

Key concepts

Validation answers the question:
are we doing **the right system**?

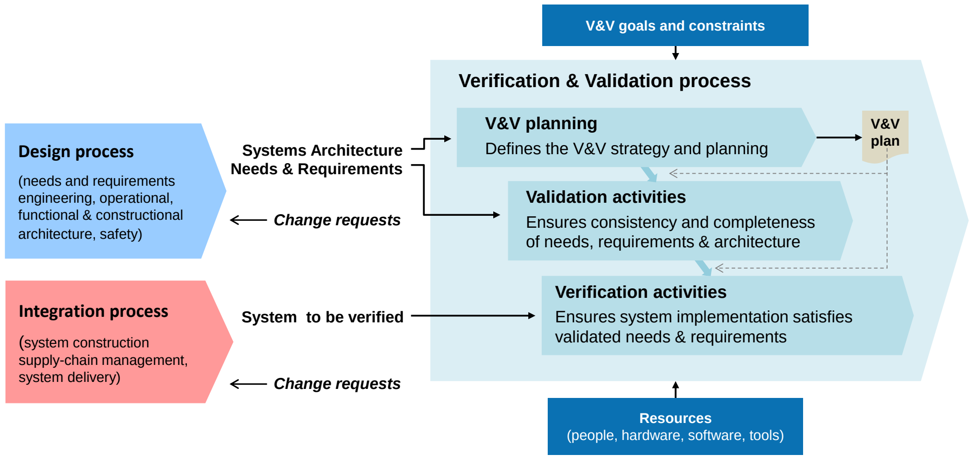
Verification answers the question:
are we doing **the system right**?



Key points

- Remedying an anomaly when a system is in service is often much more expensive than when it is detected and corrected during the engineering and V&V phases
- Be aware of the main difficulties linked to verification & validation: poor design/V&V integration, psychological difficulties, lack of time & budget, incomplete coverage
- Verification & validation are recursive processes that should be conducted at each level of a system

Process



Good practices

V&V method	Model-oriented V&V practices	Integration-oriented V&V practices
Analysis	▪ Manual or automatic analyses of a model (syntactic rules verification, crossed analyses, completeness analysis, etc.)	▪ Functional demonstrations (e.g. users interfaces, components behaviours, etc.) ▪ Prototyping (e.g. for safety analyses, etc.)
Review	▪ Model self-examinations ▪ Specifications pear reviews (quality & completeness of needs, requirements & descriptions)	▪ Pear reviews of the integrated system ▪ More or less formal reviews of the integrated system by the stakeholders ▪ Returns on experience
Test	▪ Simulations (e.g. using MATLAB & Simulink)	▪ Unitary and integration tests of the integrated system components (at each systemic level) ▪ Formal qualification of the integrated system with its stakeholders