Concepts

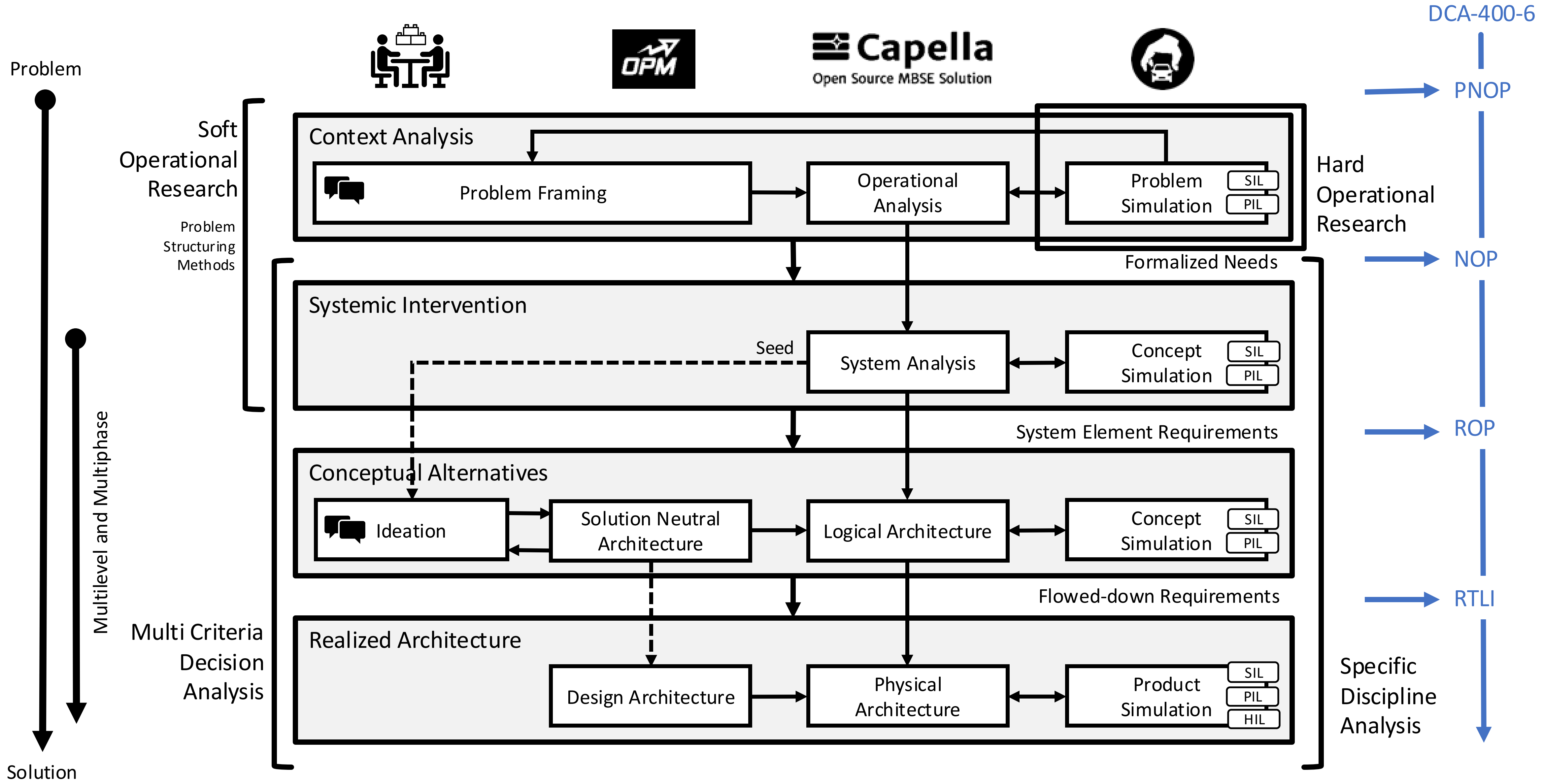
[TE-265][2024]

Prepared by Prof. Christopher Shneider Cerqueira

SEMANA		TEORIA	INDIVIDUAL	PESO	GRUPO	PESO
1	1	Estrutura e Filosofia do Curso	AI-01 - Resumo Cap 1 - HB INCOSE	10%		
05-Aug	1	O que é Engenharia de Sistemas? INCOSE				
	1	Elementos da Eng Sis.				
	1	Introdução aos diagrams clássicos.				
2		* (Viagem ao EUA)	AI-02 - Leitura/Resumo paper sobre representações clássicas.	10%		
12-Aug						
3		* (Viagem ao EUA)	AI-03 - Exercício sobre arquitetura e escrita de requisitos.	10%		
19-Aug						
4	1	Metodologias de MBSE e uso de modelos.	AI-04 - Resumo Artigo de Metodologias	10%		
26-Aug	1	Revisão de UML-SysML.				
	1	OPM				
	1	Arcadia				
5	1	OPM	AI-05 - Lista de exercícios	10%		
02-Sep	1					
	1					
	1					
6	1	Blocos e Classes	AI-06 - Lista de Exercícios	20%		
09-Sep	1					
	1	Máquina de Estados				
	1					
7	1	Casos de Uso	AI-07 - Lista de Exercícios	20%		
16-Sep	1					
	1	Sequência				
	1					
8	1	Integração dos pontos de vistas em um	AI-08 - Resumo sobre Ciclo de Vida de Modelos	10%	AI-08 - Descrição e Contorno do Problema.	100%
23-Sep	1	Associação dos artefatos de SE com modelos				
	1	Análise Operacional				
	1					
				100%		**100%**
SEM						
30-Sep						

SEMANA		TEORIA	INDIVIDUAL	PESO	GRUPO	PESO
9	1	Apresentação das necessidades			AG-09 - Apresentação Necessidades	20%
07-Oct	1	Intervenção Sistêmica				
	1	Associação com Requisitos				
	1					
10	1	Apresentação da Arq e Req de sistema	AI-10 - Exercícios de Arquitetura Funcional	20%	AG-10- Apresentação Arq / Caixa Preta	20%
14-Oct	1	Conceitos de Arquitetura Funcional				
	1	Arquitetura Conceitual				
	1					
11	1	Utilização de modelos para outros processos			AG-11 - Geração de documentos	10%
21-Oct	1					
	1	Exportação automática de documentos				
	1					
12	1	Apresentação da arquitetura Conceitual	AI-12 - Explorar RCE lendo arquivo do Capella	20%	AG-12 - Apresentação Arq. Conceitual e Proposta de VV	20%
28-Oct	1	Co-Engineering / CDF / RCE				
	1	Arquitetura Concreta				
	1					
13		* (ADS-HLG)	AG.13 - Explorar Plugin M2DOC (extra)	20%		
04-Nov						
14		* (ADS-HLG)	AG-14 - Explorar Plug in P4C (extra)	20%		
11-Nov						
15	1	Metamodelo	AG=5 - Figura do Metamodelo	20%	AG-15 = Relatório de Proposta de plugin	20%
18-Nov	1	Capella Studio - Criação de plugins				
	1					
	1					
16	1	Apresentação final			AG-16 - Apresentação do Projeto Completo	20%
25-Nov	1					
	1					
	1	Encerramento do Curso				
				100%		**110%**
EXAME *02-Dec* *13-Dec*		**Grupo: Apresentação / Relatório / Gravação / Código de um: plugin ou doc**				**100%**

MMMF

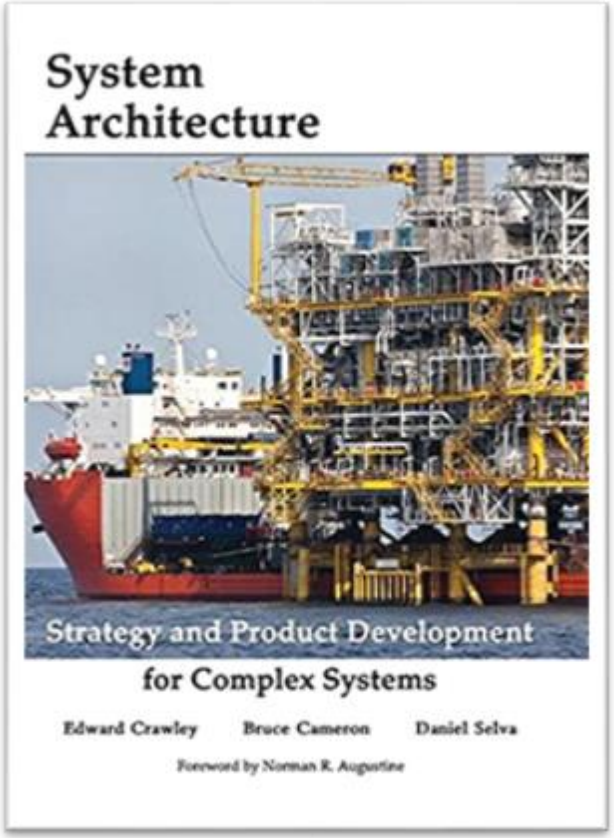


Problem
Solution
Multilevel and Multiphase
Soft Operational Research
Problem Structuring Methods
Multi Criteria Decision Analysis
Capella
Open Source MBSE Solution
OPM
Context Analysis
Problem Framing
Operational Analysis
Problem Simulation
SIL
PIL
Hard Operational Research
Formalized Needs
Systemic Intervention
Seed
System Analysis
Concept Simulation
SIL
PIL
System Element Requirements
Conceptual Alternatives
Ideation
Solution Neutral Architecture
Logical Architecture
Concept Simulation
SIL
PIL
Flowed-down Requirements
Realized Architecture
Design Architecture
Physical Architecture
Product Simulation
SIL
PIL
HIL
Specific Discipline Analysis
DCA-400-6
PNOP
NOP
ROP
RTLI

IDEATION USING OPM

Context

- **Sistemas entregarão um valor para as partes interessadas**.
- O valor vem com um forma (elementos de arquitetura e propriedades) e com funções (ações e eventos)
- **Sistemas Complexos possuem múltiplas funções**.
- Os Engenheiros de Sistemas devem identificar e organizar essas funções, gerenciando a complexidade.



OPM Function

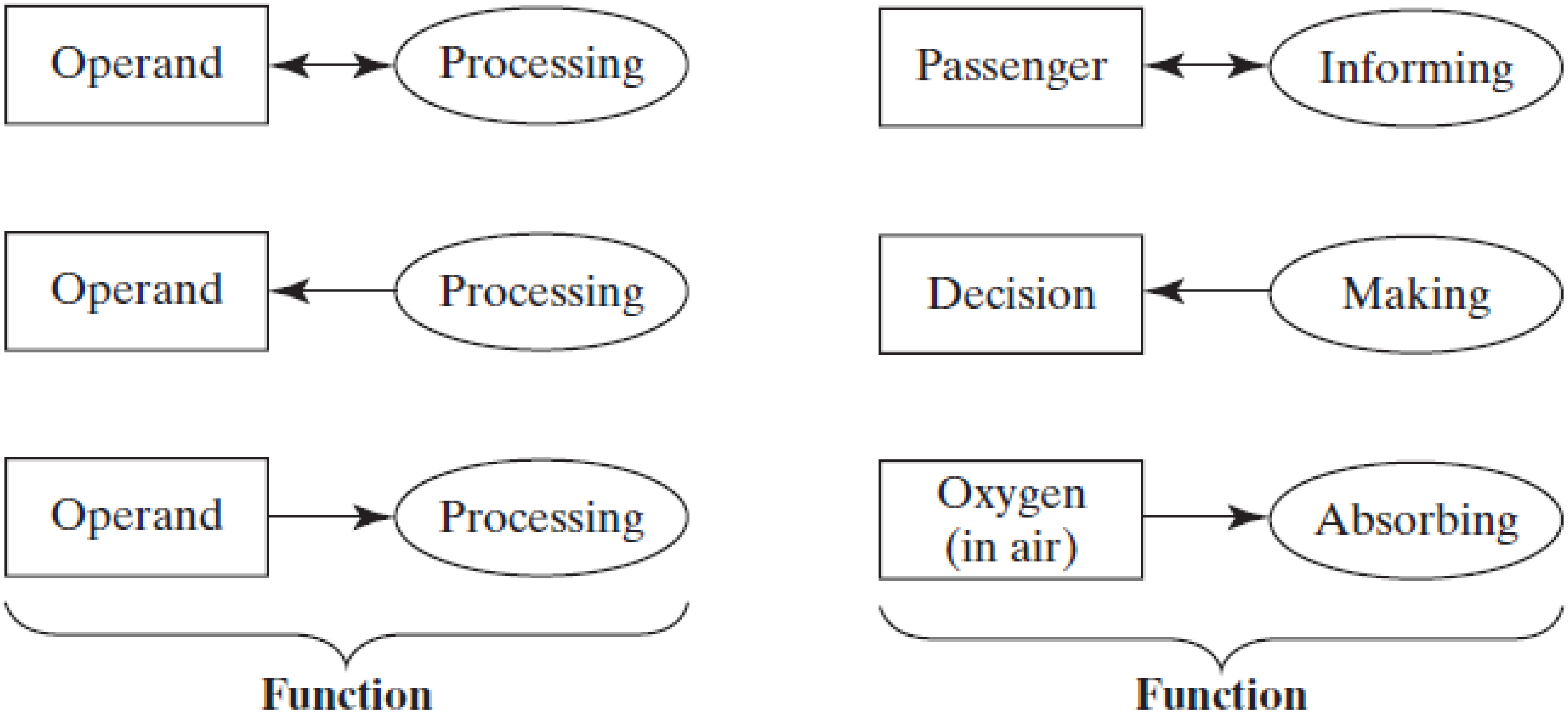


FIGURE 5.2 OPM diagram of process + operand yielding function.
From top to bottom, these represent the process affecting the operand, consuming the operand, and producing the operand.

OPM Funcion + Form (instrument)

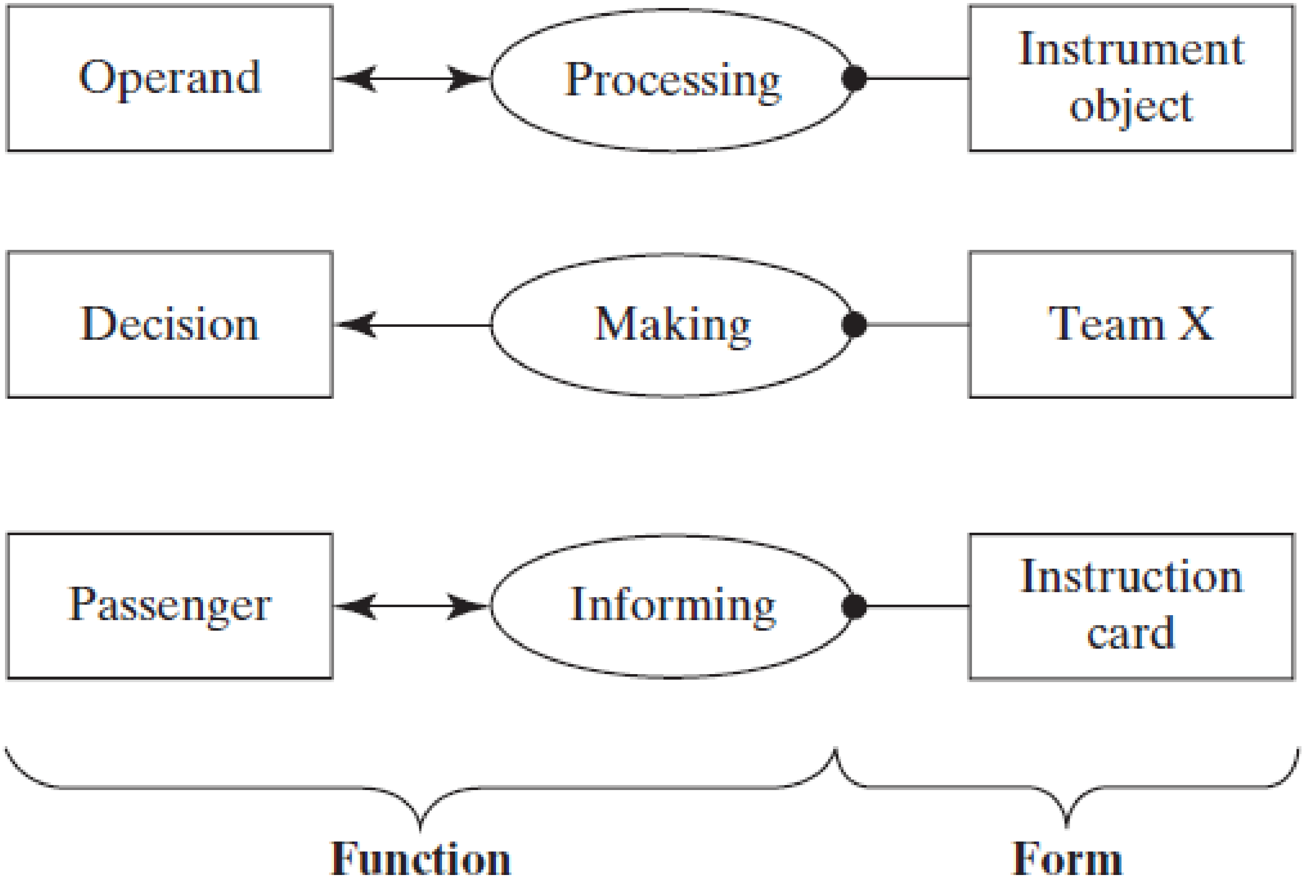


FIGURE 5.3 OPM representation of the canonical system architecture: The function as a process and an operand that the process affects, and the form as an instrument object.

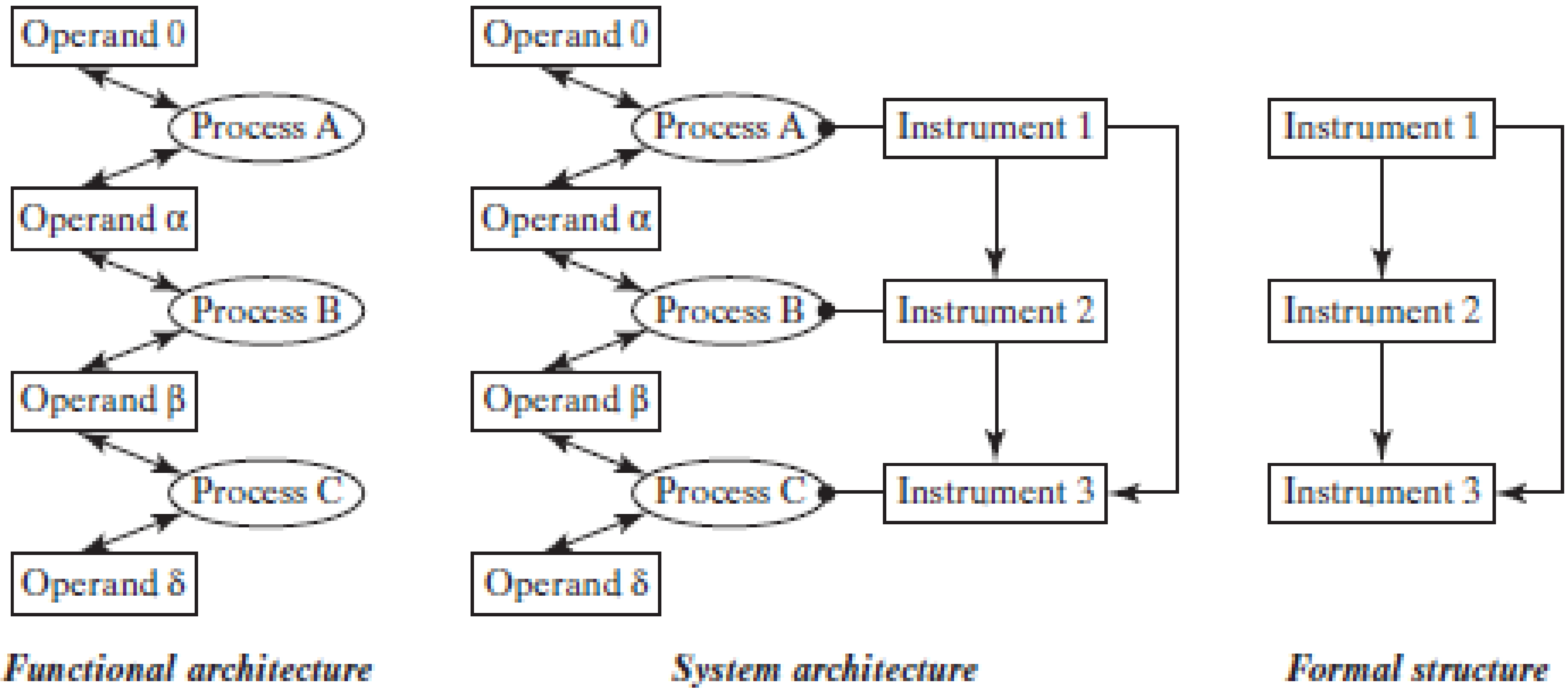


FIGURE 6.1 System architecture as the combination of functional architecture and elements and structure of form.

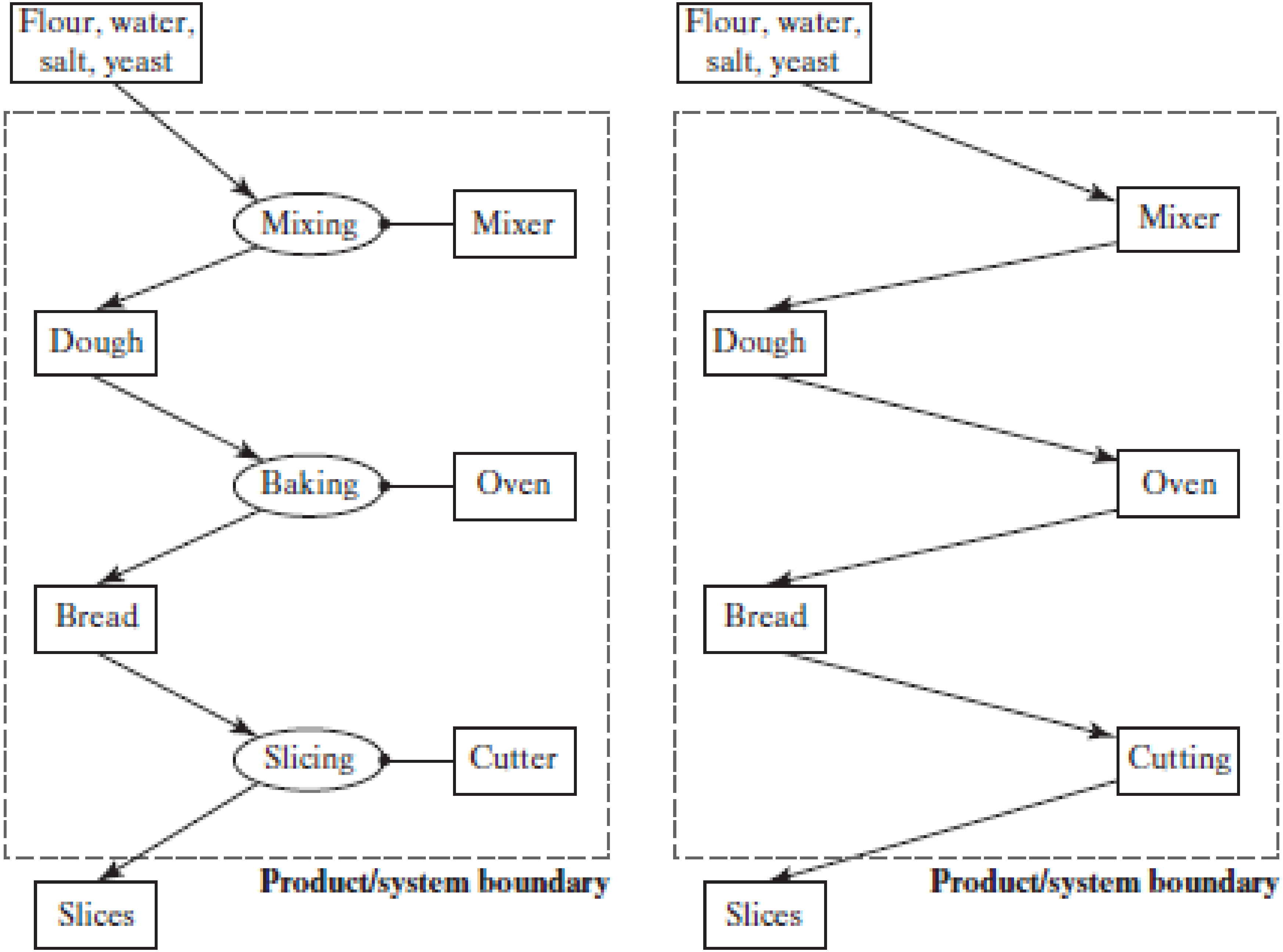


FIGURE 6.3 Simple system architecture of sliced bread making.

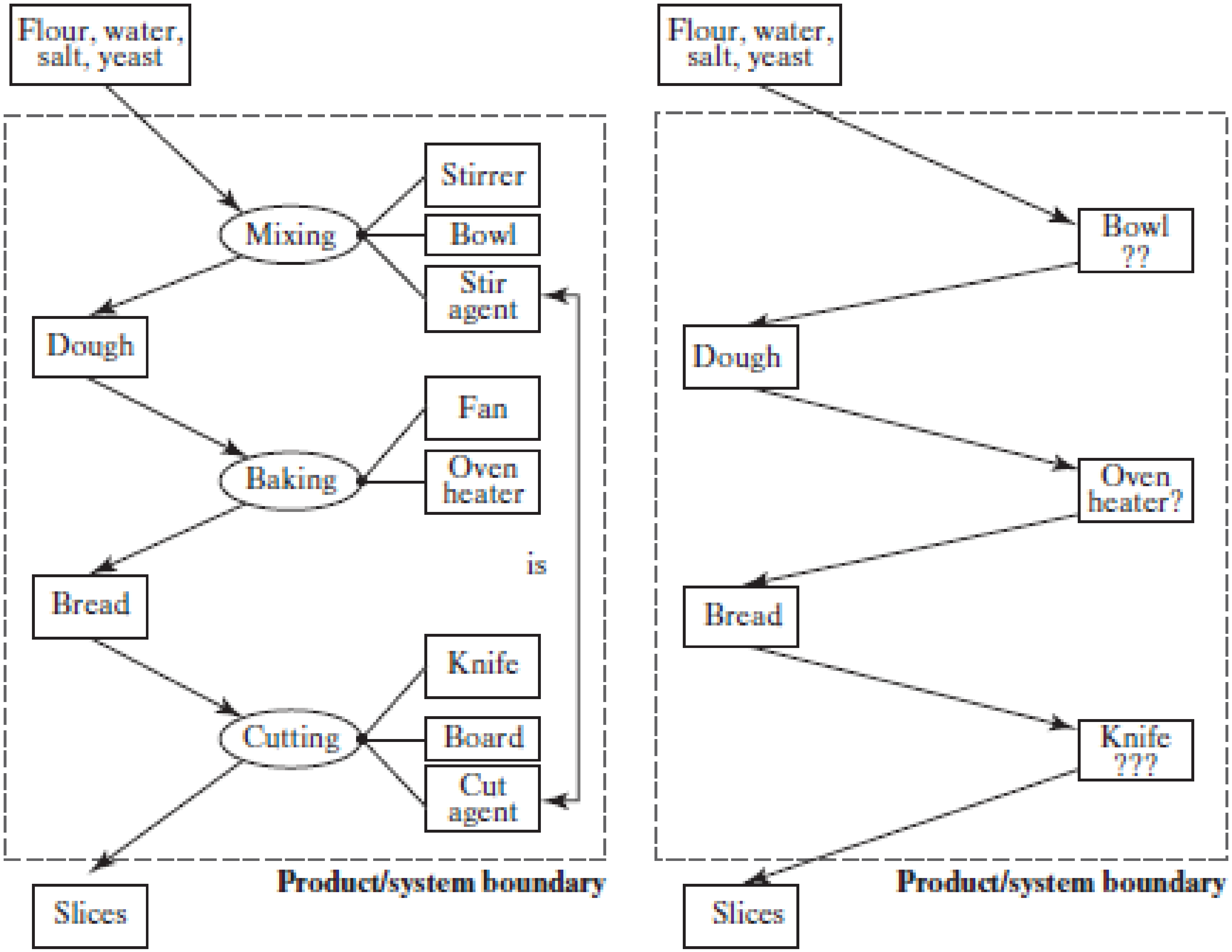


FIGURE 6.4 A not-so-simple system architecture of sliced bread making.

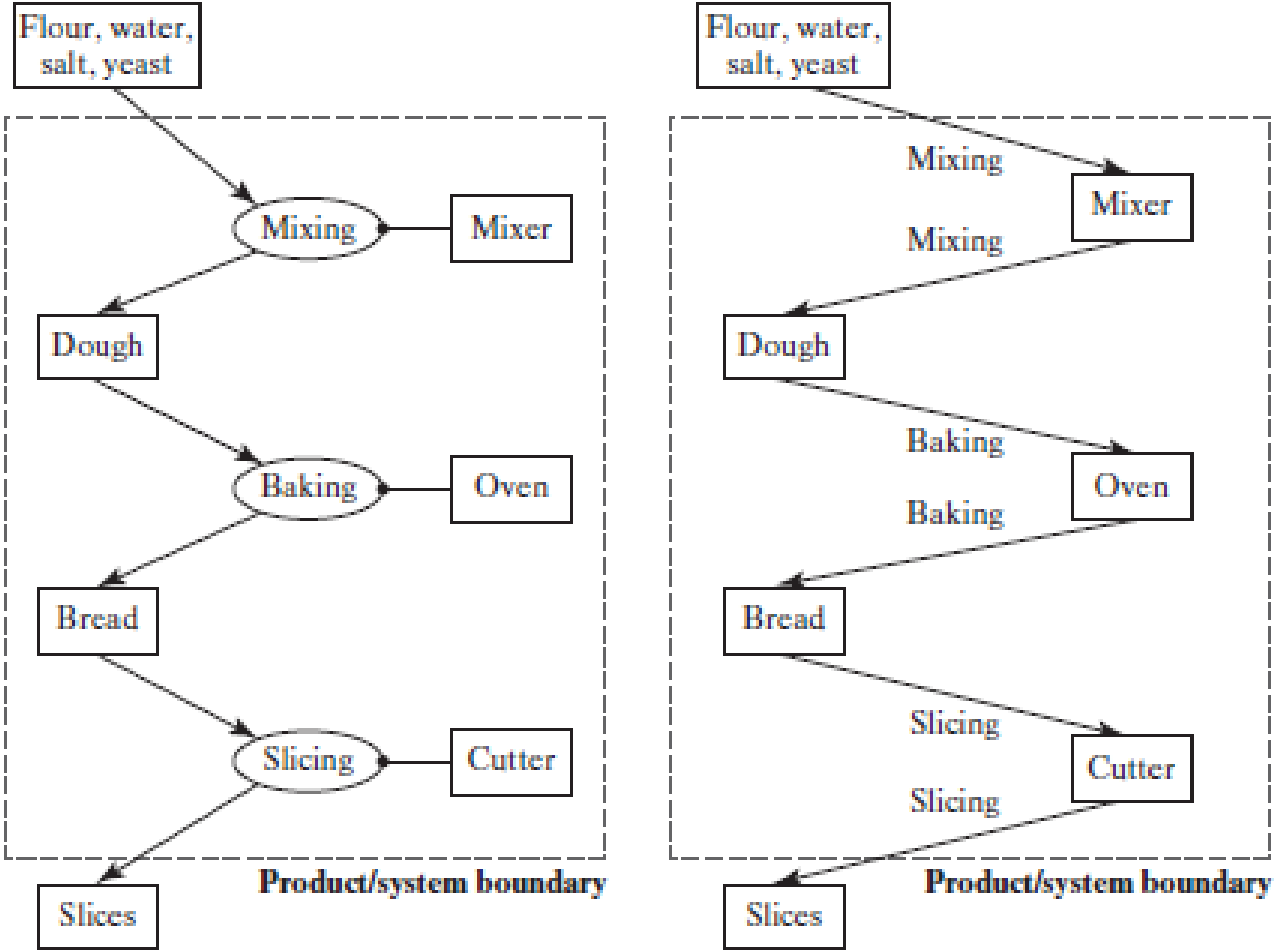


FIGURE 6.16 Projection onto the objects of making sliced bread.

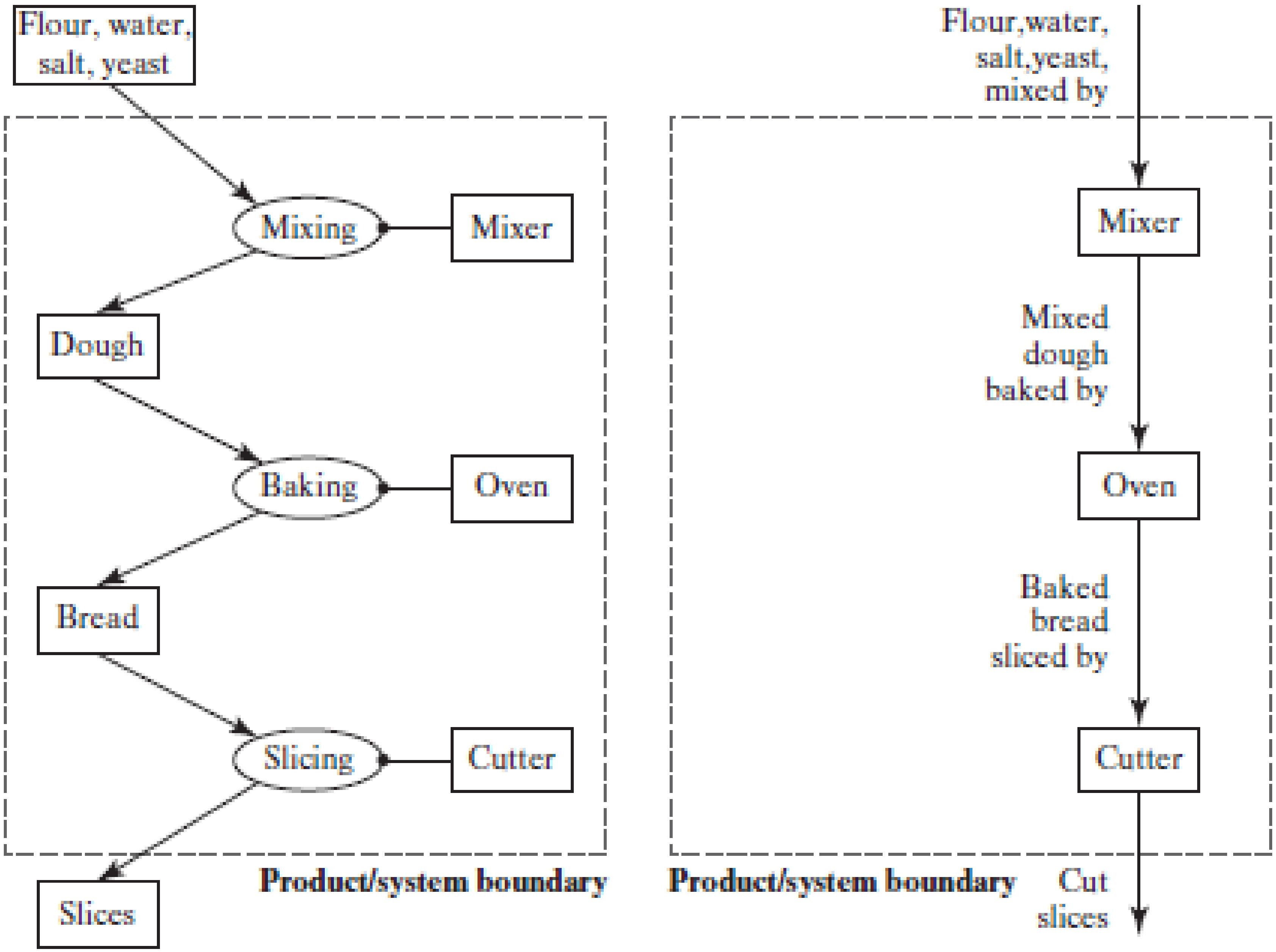
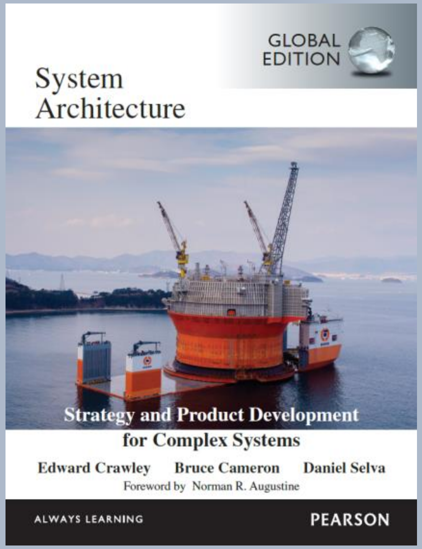


FIGURE 6.19 Projection onto the form of making sliced bread. (Compare with Figure 6.16.)



Solution-Neutral (Solution)

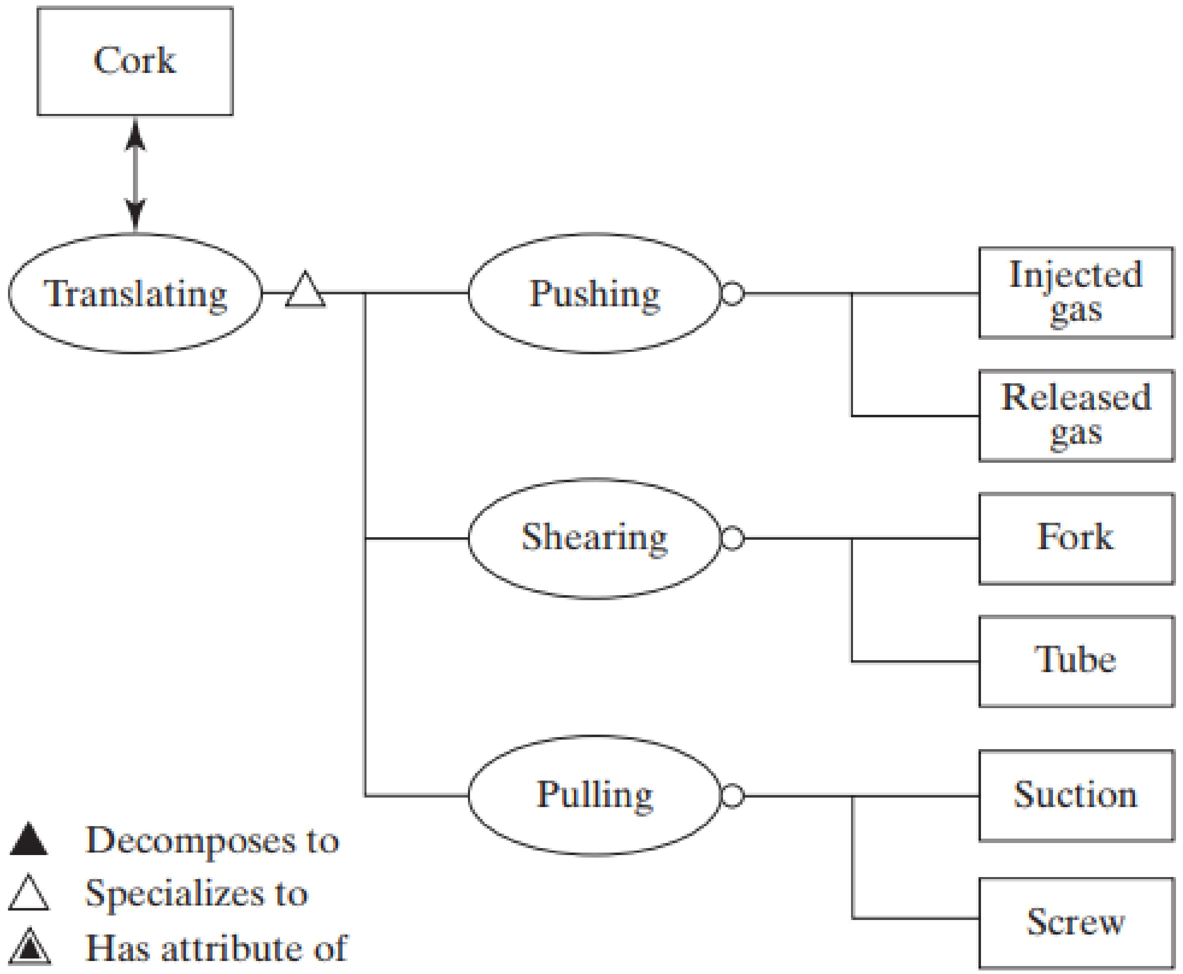
Chapter 7 - 8

Solution-neutral questions

Questions	Produces
7a. Who are the beneficiaries? What are their needs? What is the solution-neutral operand whose change of state will meet these needs? What are the value-related attribute and the solution-neutral process of changing the states? What are the other attributes of the operand and process?	A solution-neutral framing of the desired function of the system
5a. What is the primary externally delivered value-related function? The [specialized] value-related operand, its value-related states, and the [specialized] process of changing the states? What is the abstraction of the instrumental form? [What is the concept? What are several other concepts that satisfy the solution-neutral function?]	An operand–process–form construct that defines the abstraction of the system
5b. What are the principal internal functions? The internal operands and processes? [What are the specializations of those processes? What are the concept fragments? What is the integrated concept? What is the concept of operations?]	A set of processes and operands that represent the first-level and potentially second-level downward abstractions of the decomposed system.

EX.: Concept of remove a cork from a wine flask.

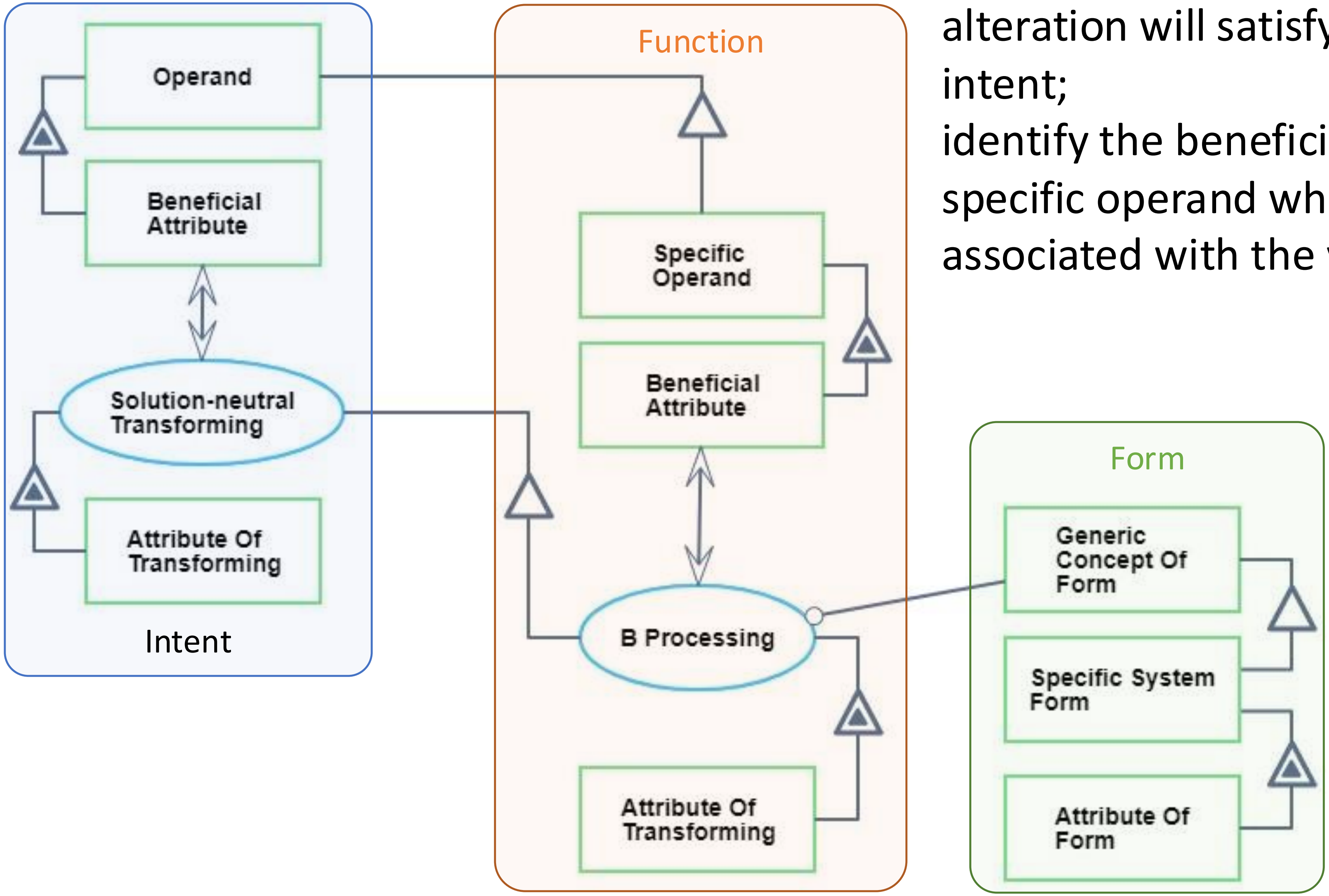
- This solution is actually sold as a household product, complete with a thin, hollow needle for piercing the cork and a manual air pump.
- Alternatively, the waiter at fine dining restaurants (which I don't go to) sometimes uses a fork-like device to pull the sides (shear) of the cork.



Solution Neutral reasoninng

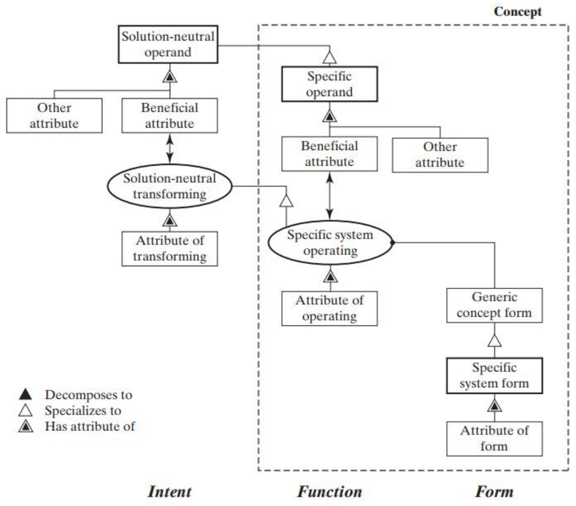
- Solution-neutral reason is to fnid the function of a system declared without reference to how the function is achieved (should only indicate the problem).
- **Poor system requirements often contain clues about a solution**, intended function or form, and these clues can lead the architect to a narrower set of potential options.
- **Use solution-neutral functions whenever possible** and use the hierarchy of solution-neutral statements to allow for better exploration of the problem.

Solution-Neutral



Identify the specific operand, whose alteration will satisfy the functional intent;
identify the beneficial attribute of the specific operand whose change is associated with the value, and so on.

Examples



Solution-Neutral Operand	Solution-Neutral Process	Specific Operand	Specific Process	Specific Instrument
Fluid	Moving	Water	Pressurizing	Centrifugal pump
Array	Sorting	Array entries	Sequentially exchanging	Bubblesort
Cork	Translating	Cork	Pulling	Screw
Traveler	Transporting	Traveler	Flying	Airplane
Book	Buying	Internet	Accessing	Home DSL connection

Removing a Wine Bottle Cork

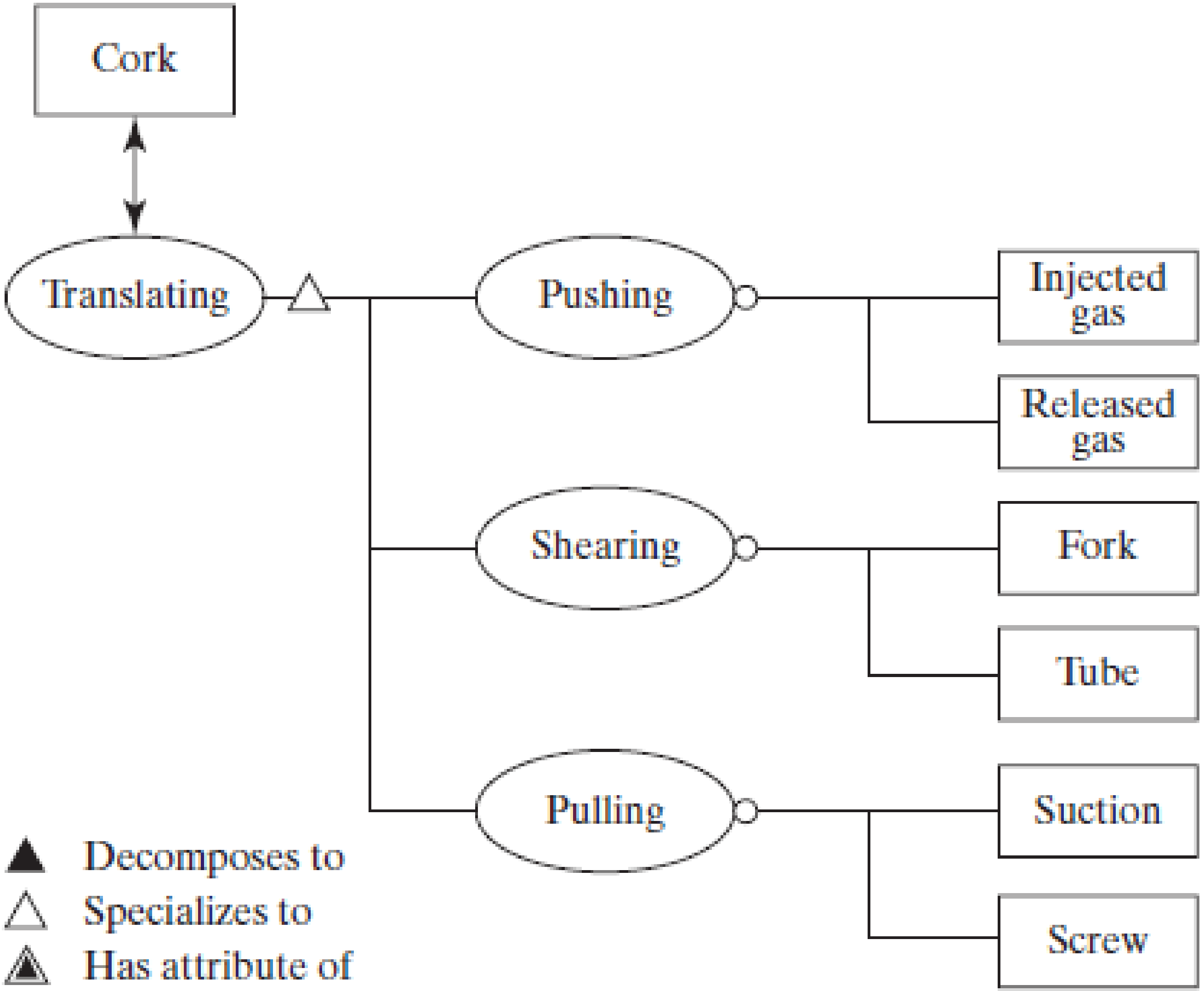
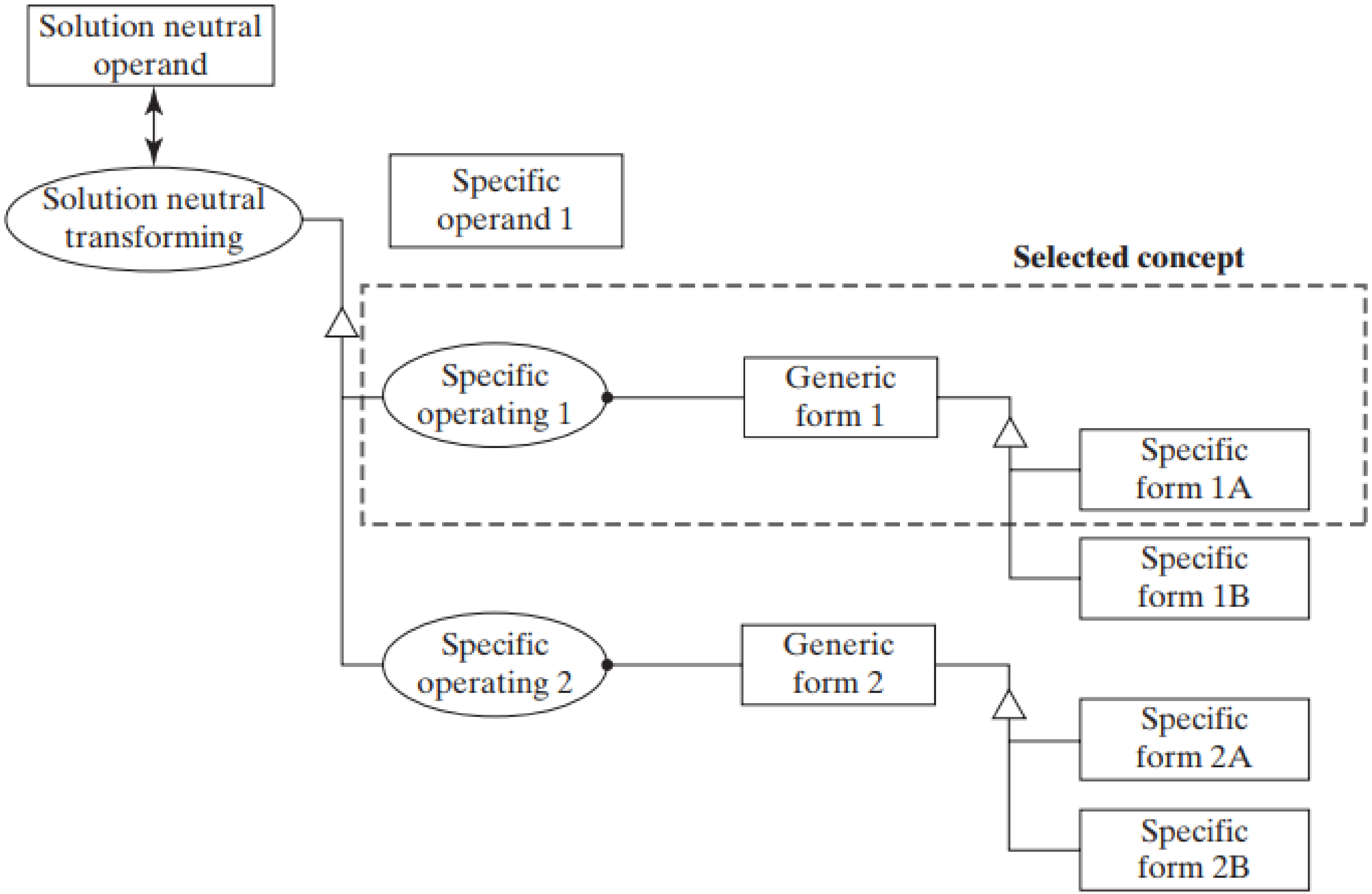


FIGURE 7.1 Concepts for removing a wine bottle cork.

Selectable Options



Example of options

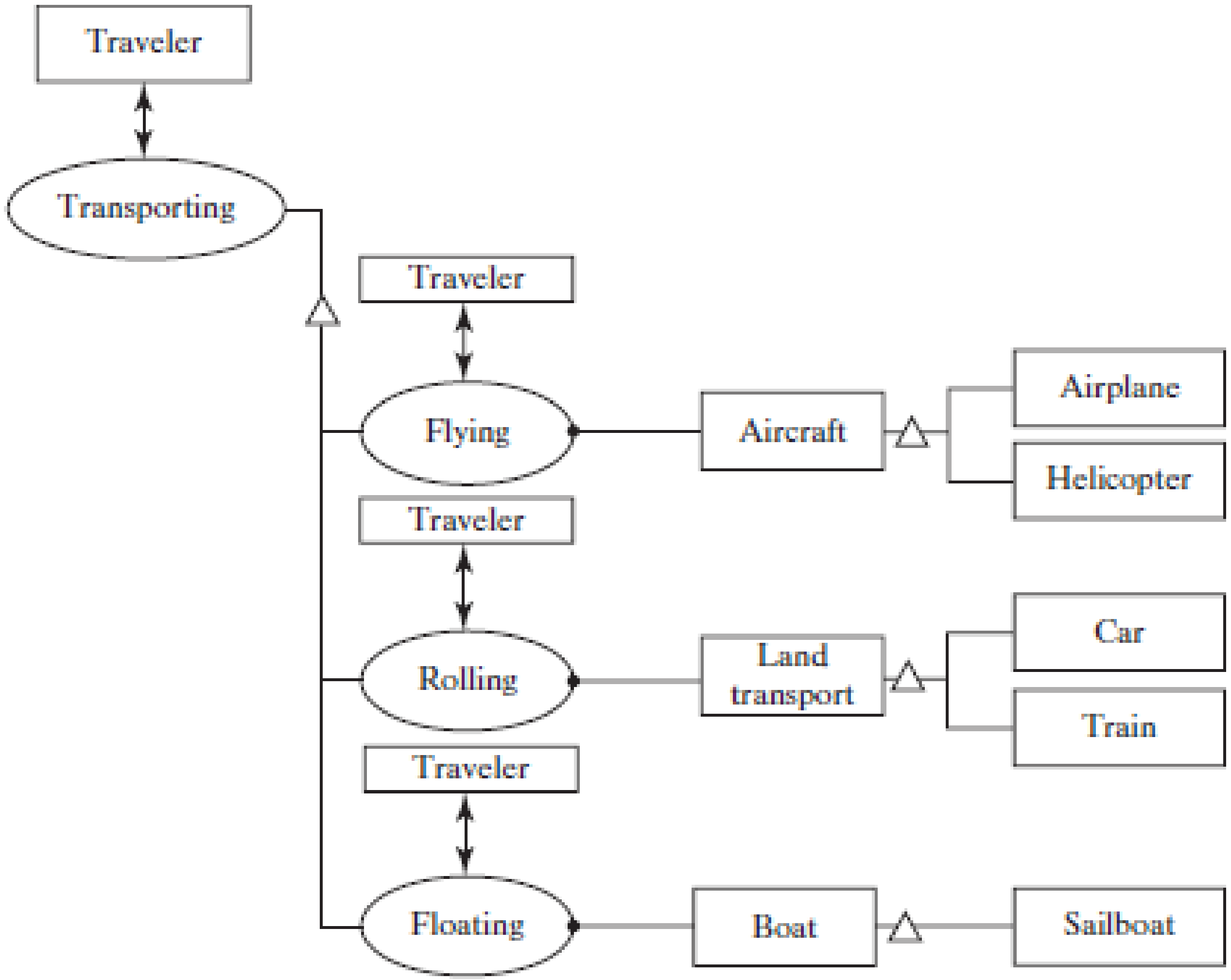
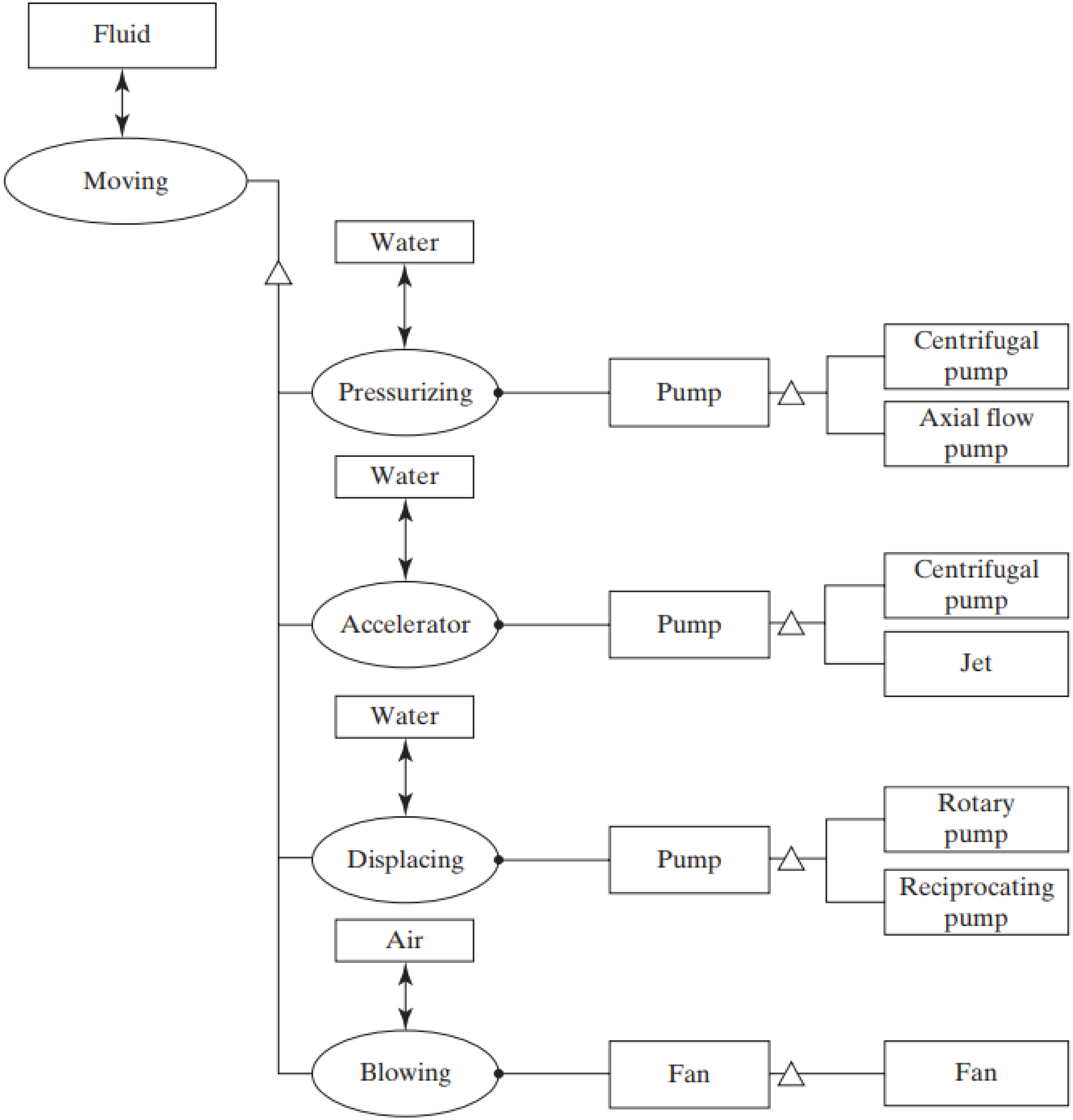
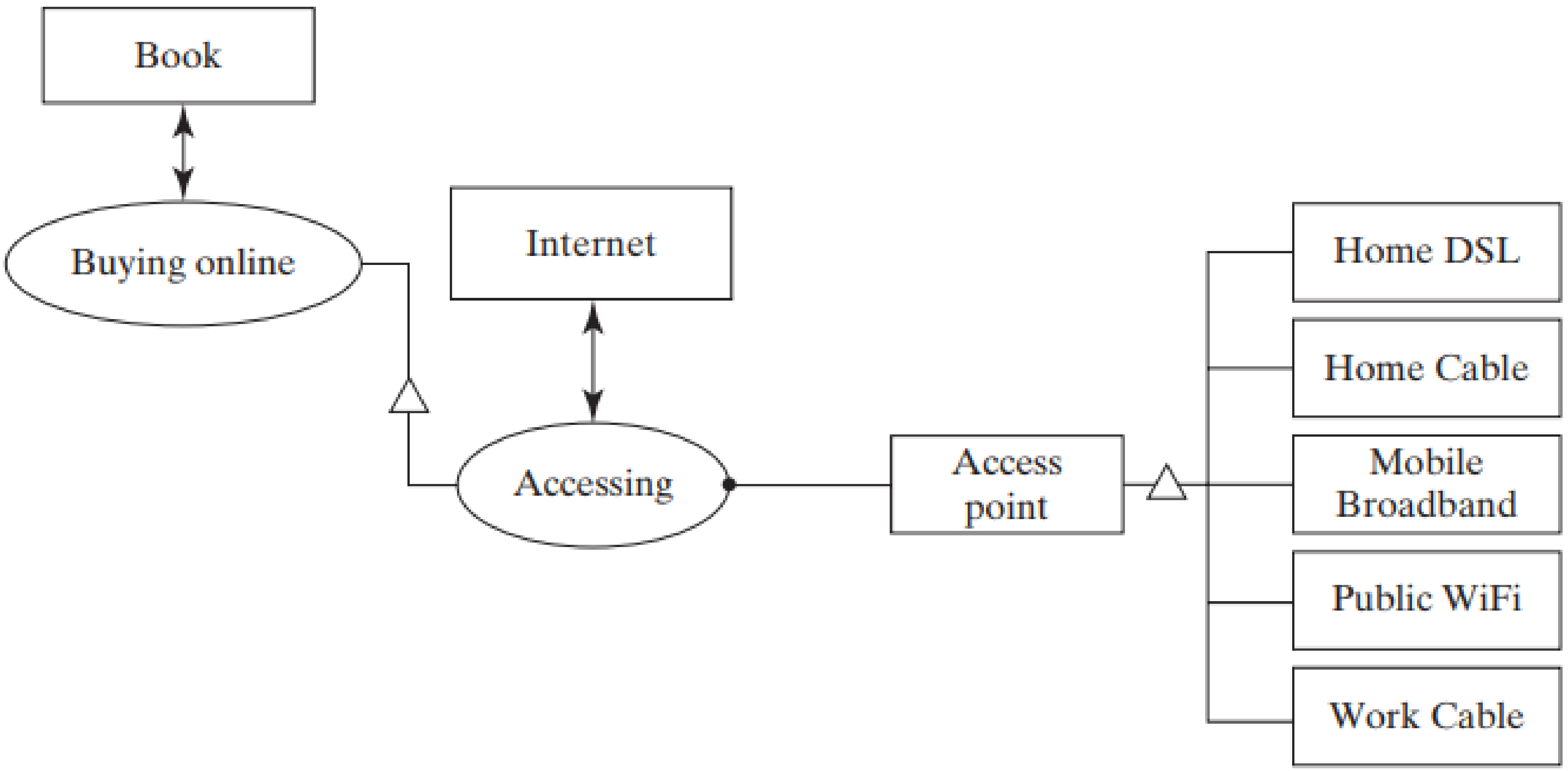
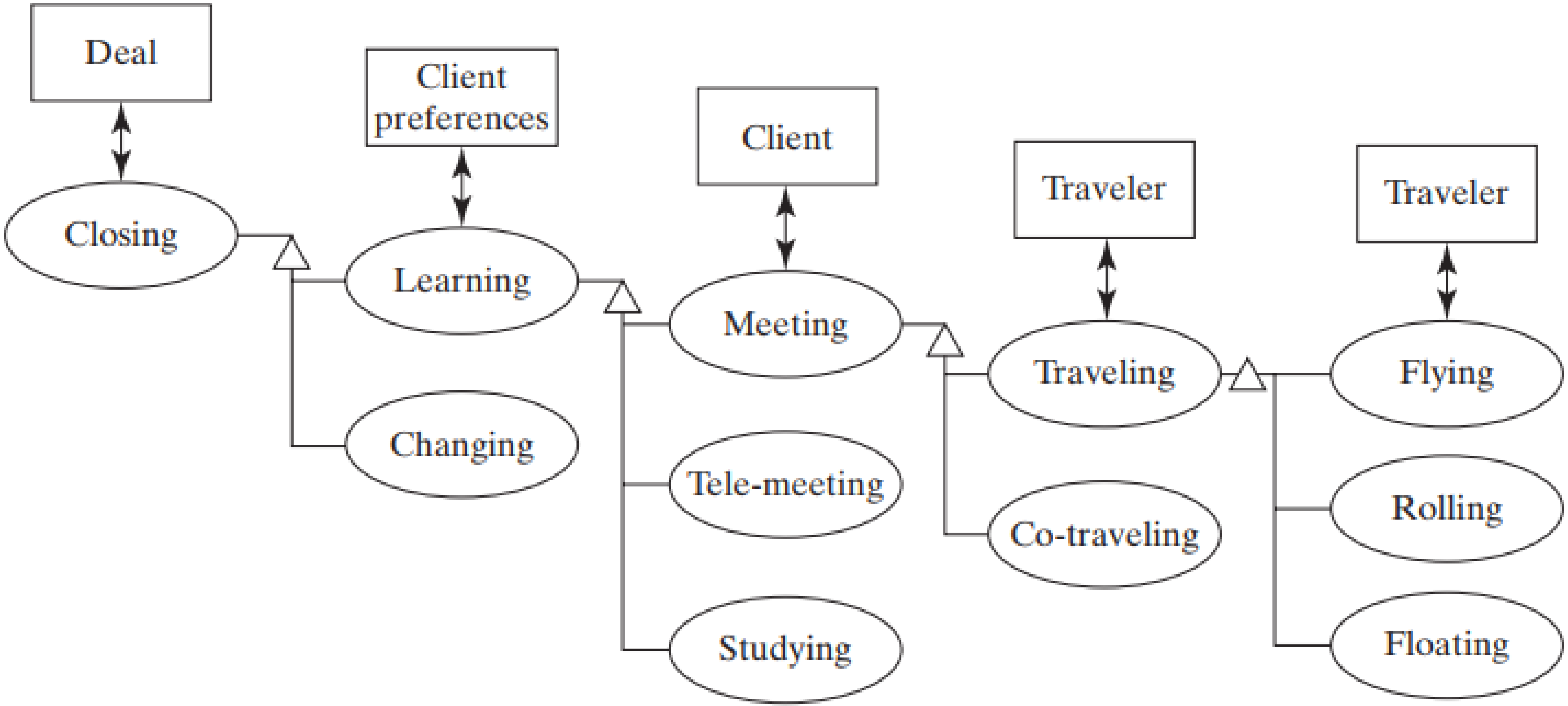


FIGURE 7.8 Solution-neutral function and solution-specific concept options for "transportation service" system.

SYSTEMS ARCHITECTURE USING OPM


Fluid
Moving
Water
Pressurizing
Pump
Centrifugal pump
Axial flow pump
Water
Accelerator
Pump
Centrifugal pump
Jet
Water
Displacing
Pump
Rotary pump
Reciprocating pump
Air
Blowing
Fan
Fan


Book
Buying online
Internet
Accessing
Access point
Home DSL
Home Cable
Mobile Broadband
Public WiFi
Work Cable



Deal
Client preferences
Client
Traveler
Traveler
Closing
Learning
Changing
Meeting
Tele-meeting
Studying
Traveling
Co-traveling
Flying
Rolling
Floating

Solution domain

T2 – Consider the whole problem, the whole solution and the full lifecycle

- Systems Engineering is concerned with the **whole problem and the whole solution**, including how the "**intervention system**" will interact with its environment as part of a larger system when it is deployed, and all the enabling systems and services required to establish and maintain system effectiveness throughout its lifecycle until eventual satisfactory disposal.
- We need to consider the **full lifecycle of the entire solution, including all the enabling systems that go along with the system of interest**

PROBLEM SPACE

- viewing the **problem as a system**,
- understanding how the **interdependencies between the elements in the problem space create the "problem symptoms"**, and how the "intervention system" might alleviate the problem symptoms
- **understanding stakeholder interactions** and interdependencies and establishing **overall agreed purpose** and success criteria
- anticipating and aiming to minimize potential adverse or unintended consequences of the intervention system
- scanning for and early detection of anomalous behavior and unintended consequences – not all can be anticipated beforehand

Table 4: Applying SE Tenets to the Problem Domain

Tenet	Problem Domain application
Understand what success means	Establish a clear definition of purpose and success criteria. Understand what success means for each stakeholder, and establish shared vision, purpose and success criteria among all stakeholders.
Whole problem whole system whole lifecycle	View the problem as a system. Recognise that introducing a new system that solves or suppresses the existing problem may create new ones. Situation may include technical, societal and naturally occurring systems.
Understand and manage interdependencies	Understand how the elements of the problem interact, and how the new system will interact with and change the environment. Understand the relationships and interdependencies between stakeholders, and how that affects their views of purpose and success criteria. Understand how different success criteria interact to constrain or expand the solution space.
Adapt the parts to serve the purpose of the whole	It may be possible to solve the problem by changing the existing operational practices rather than adding a new system.
Systems Engineering at multiple levels	Each layer of the solution provides the problem for the layers below.
Base decisions on evidence and reasoned judgement	Maintain clear line of sight from all decisions to purpose and success criteria. Characterise the problem – e.g. is it simple, chaotic, complicated or complex? – and use an appropriate approach (e.g. for complex – "probe sense respond"). Where "evidence" is not available or inconclusive, use reasoned judgement to make the best decisions possible at the time.
Uncertainty, change, risks, opportunities and expectations	Understand dynamics of change in problem space. Accept we don't know everything, we can't know some things, and almost everything changes. Realise that sometimes we can't fully understand the problem until we start work. Manage stakeholder expectations to be realistic.
Structure and behaviour	Look for structures in the problem space that cause the problematic behaviours. Look for leverage points that will allow an intervention system to alleviate the problem while minimising adverse consequences.
Feedback	Look for the feedback mechanisms that keep the problem going in spite of attempts to solve it. Understand how the feedback mechanisms in the problem space might be exploited to amplify the effects of a successful intervention.
Value	Understand stakeholders' perceptions of value. Seek opportunities to manage and maximise value.
Systemic and Systematic	Use appropriate shared model-building methods and tools to enable the transdisciplinary group to frame the problem, for example Soft Systems Methodology (SSM), Interpretive Structural Modelling (ISM), Causal Loop and System Dynamic Modelling (CLM/SDM).
Respect the people	Ensure you understand who is affected by the problem and possibly who is benefiting from it. Bring those people together and facilitate in such a way that they all have equal opportunity to express those opinions, and that those opinions are valued and considered.

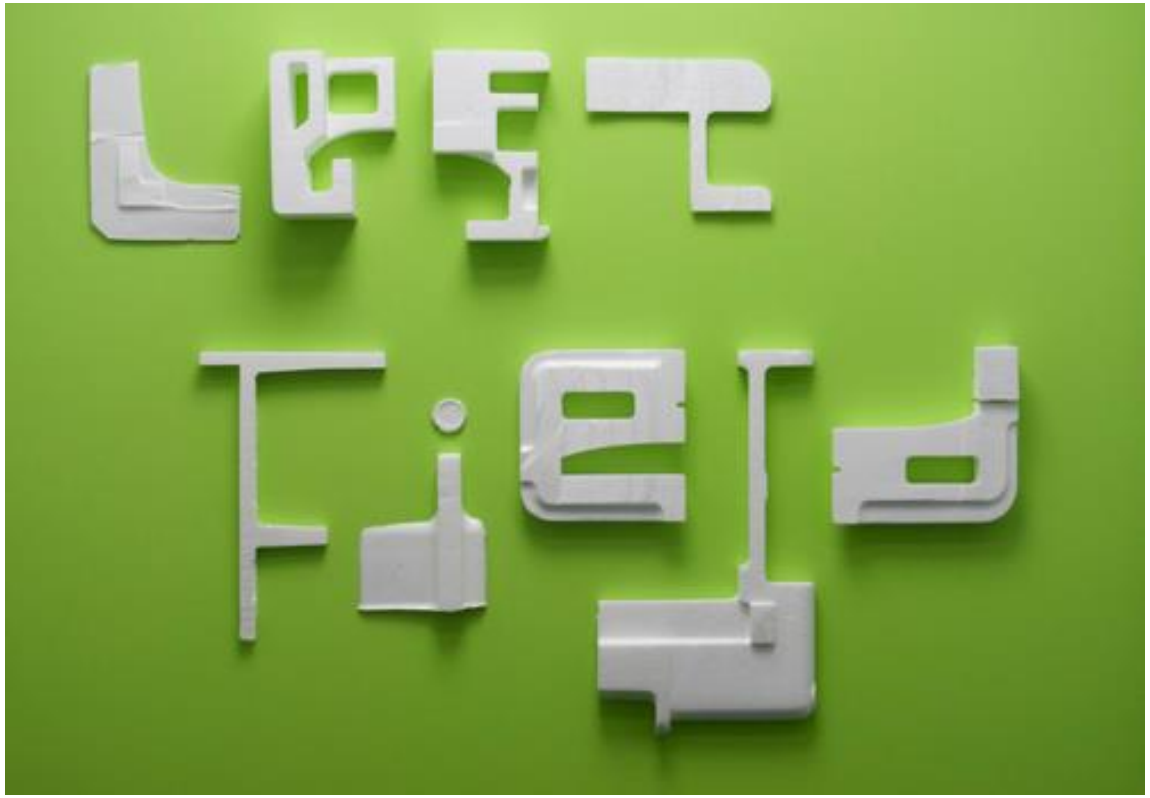
Solution space

Table 5: Applying SE Tenets to the Solution Domain

Tenet	Solution Domain Application
Understand what success means	Establish a clear agreed objective and success criteria for the intervention system. Understand how every part of the solution contributes to fitness for purpose.
Whole problem whole system whole lifecycle	The solution may be a product, product line, service or enterprise. Consider all aspects of the solution including enabling systems and services required for a successful system that is fit for purpose.
Understand and manage interdependencies	Systems Engineering focuses, **not** on the design of the parts, but on the required properties of the parts and on how they need to interact to create the required properties, capabilities, behaviours and outcomes of the whole solution. Understand how key attributes of the solution interact to shape the "fitness landscape" of potential solutions. Simplify and manage interfaces.
Adapt the parts to serve the purpose of the whole	Parts may need to be sub-optimised, and off-the-shelf parts reconfigured, so that they work harmoniously as part of the system.
Systems Engineering at multiple levels	Several layers of the solution may have parts that are worth treating as systems in their own right.
Base decisions on evidence and reasoned judgement	Select a suitable approach based on evidence, guided by purpose, and taking into account the levels of risk, uncertainty and change. Take decisions at the "last responsible moment", using reasoned judgement if evidence is not available or inconclusive.
Uncertainty, change, risks, opportunities and expectations	Keep options open until at least one is definitely suitable. Architect to allow opportunities to be exploited, and to minimise cost and disruption of likely late changes. Design for adaptability and evolutionary learning. Manage participants' expectations to be realistic.
Structure and behaviour	Apply equal attention to behavioural/logical view of the system as to the physical and structural view (develop both logical and physical architectures).
Feedback	Understand and carefully manage the feedback loops in the solution and in the interaction between the solution and its environment so that the system will behave in an intuitively acceptable way both in envisaged and (to the greatest extent possible) in unexpected situations.
Value	Focus on maximising value. When working to meet a cost target, make sure the value proposition of the solution remains compelling.
Systemic and Systematic	Keep attending both to the big picture ("are we building the right system, will it solve the problem?") and the detail ("will it work as expected in all relevant situations?")
Respect the people	Consider the implications of different cultures involved in using and supporting the system. Realise that the technical part of the system is usually there to help the people using it, not the other way round. Make sure that the technical and process solutions fit the mental models and reasonable expectations of the intended users, operators and other stakeholders. Make sure that unintended negative consequences for all stakeholders are avoided or mitigated to the greatest extent possible.

- In the solution space, the SE approach involves
  - identifying **potential solution** approaches,
  - selecting a **suitable approach** based on evidence and **expert judgement, guided by purpose**, and taking into account the **levels of risk, uncertainty and change**;
  - defining the solution, the component parts and their properties, and the required enabling products and services to design, make, test, deploy, use, assess, support, evolve and eventually retire and dispose of the system

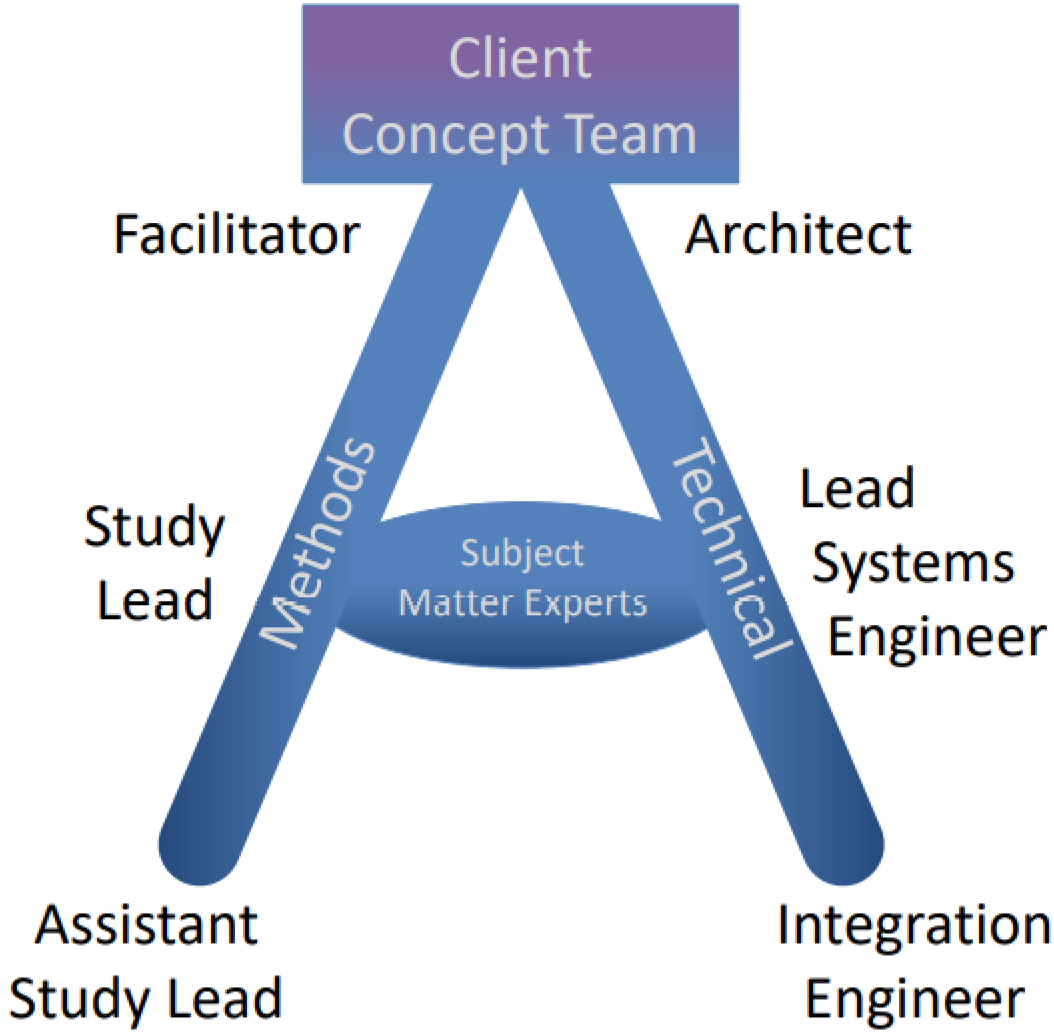
Essa fase é feita por estruturadores de arquitetura... Exemplo é o left field do jpl



https://www.jpl.nasa.gov/thestudio

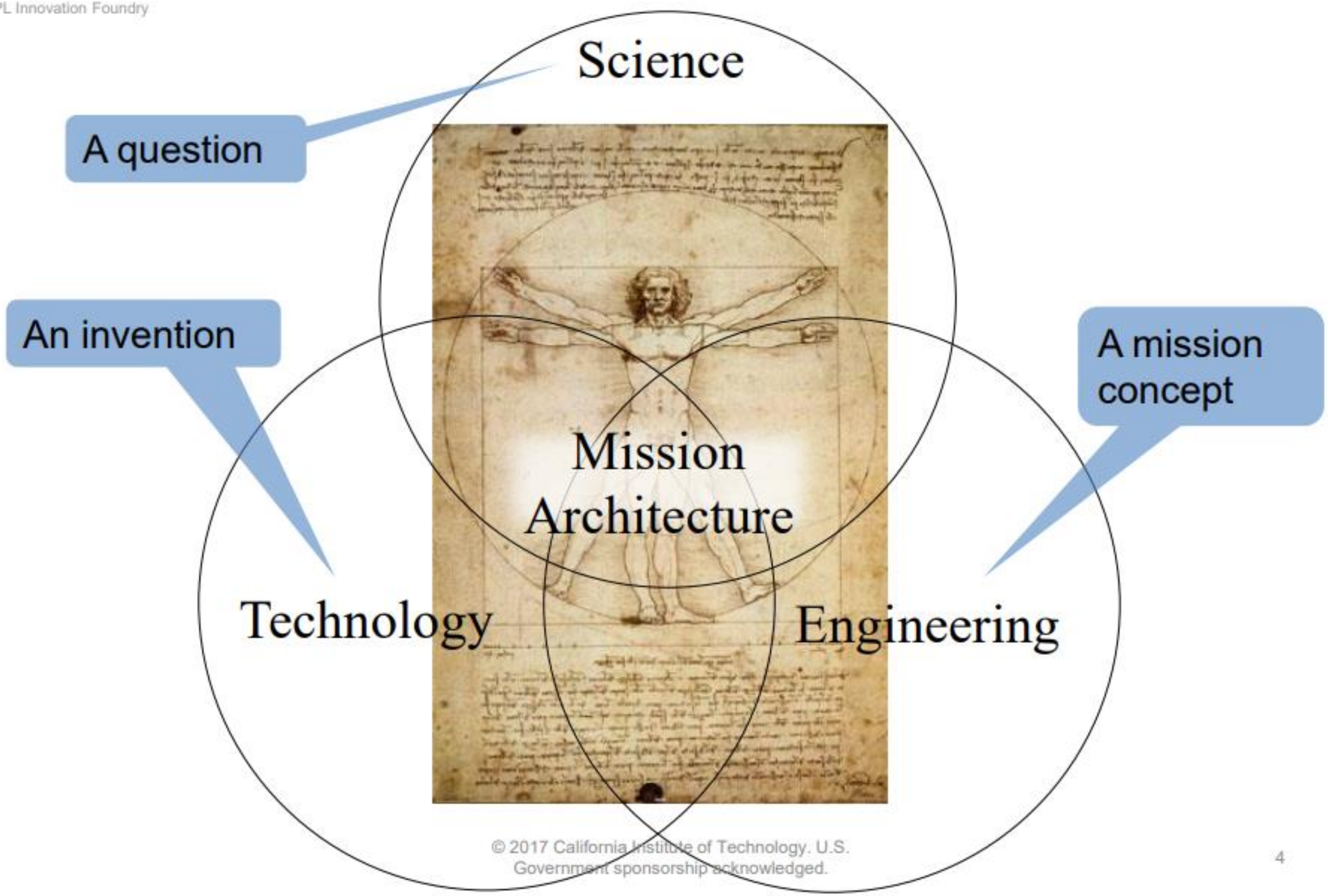
The “A-Frame”

JPL Innovation Foundry



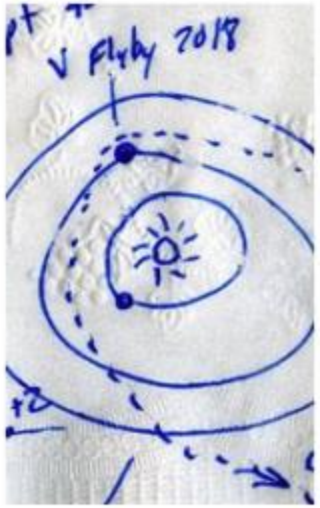
- Each A-Team study has a 3-6 person “A-Frame Team” from two points of view:
  - Innovative Methods
  - Technical Expertise
- Additional subject matter experts are brought in as needed (customized)
- The client may also add members from the Concept Team

Every mission starts with a spark

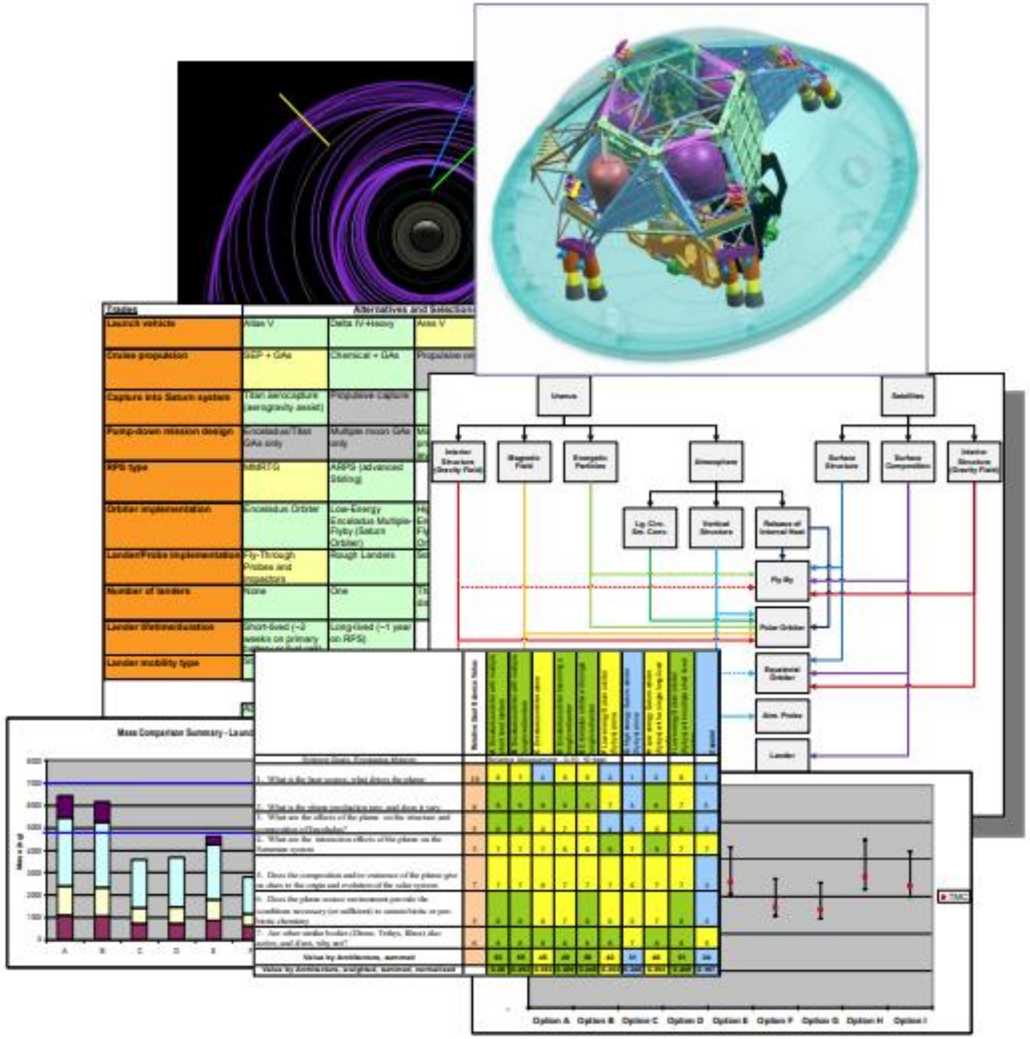


JPL Innovation Foundry

…then the concept is developed



or



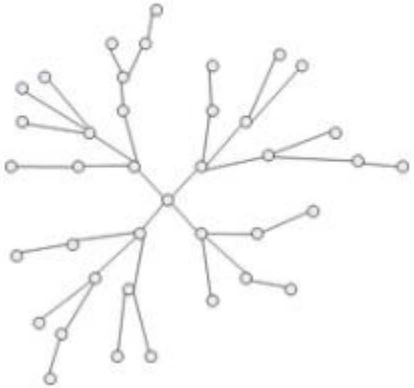
One person's concept is another's doodle…

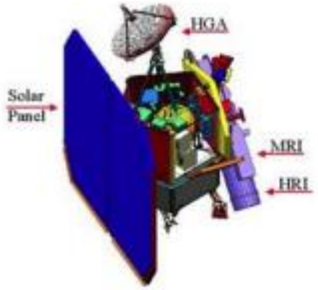
JPL Innovation Foundry

A-Team Methods

CML 1

Capturing ideas and linking associated ideas

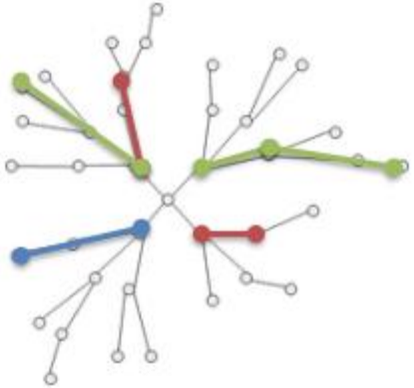




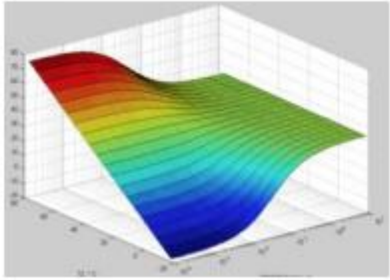
Research, bringing in previous studies

CML 2

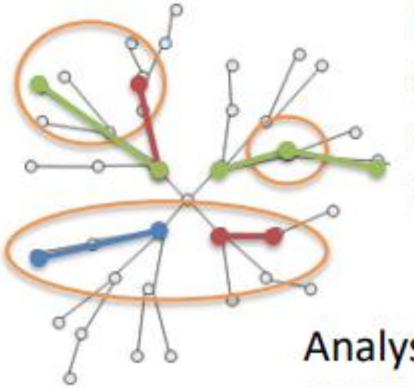
Testing assumptions, relationships, and links



Analyzing feasibility, finding FOMs and thresholds

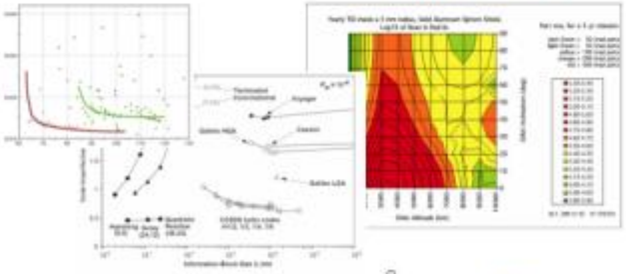


CML 3



Building seed science cases and concept architectures

Analysis and trade space exploration



Rapid prototyping of concepts

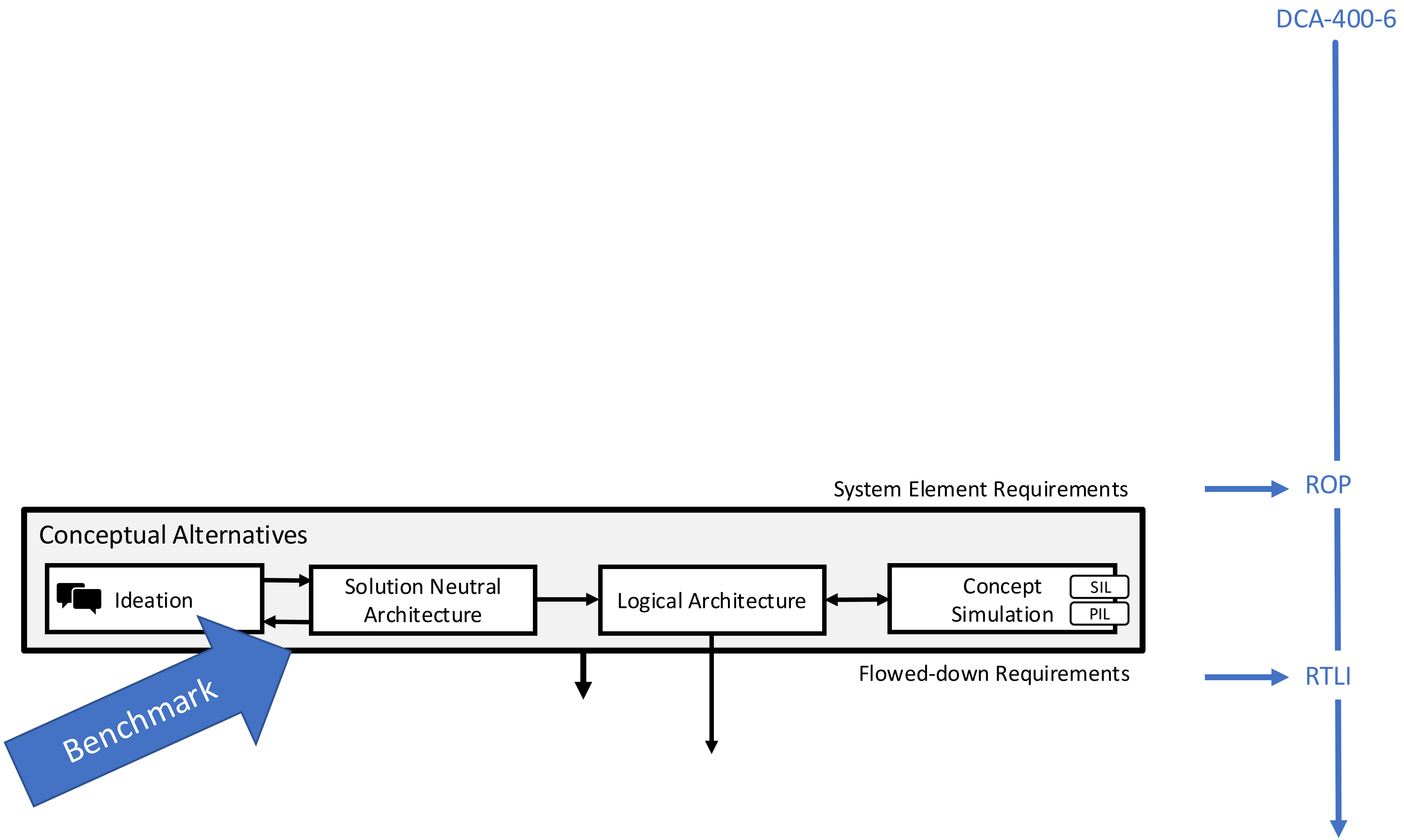
DECISION MATRIX – PUGH'S METHOD

Criteria	Wt	Vendor 1	Vendor 3	Vendor 4
		Alternatives		
Cost	.30	4	4	4
Response time	.17	3	3	5
Training time	.17	2	4	5
Ease of use	.17	1	4	4
Strong team	.10	3	4	
Team experience	.10	3	4	
Total				
Weighted total				

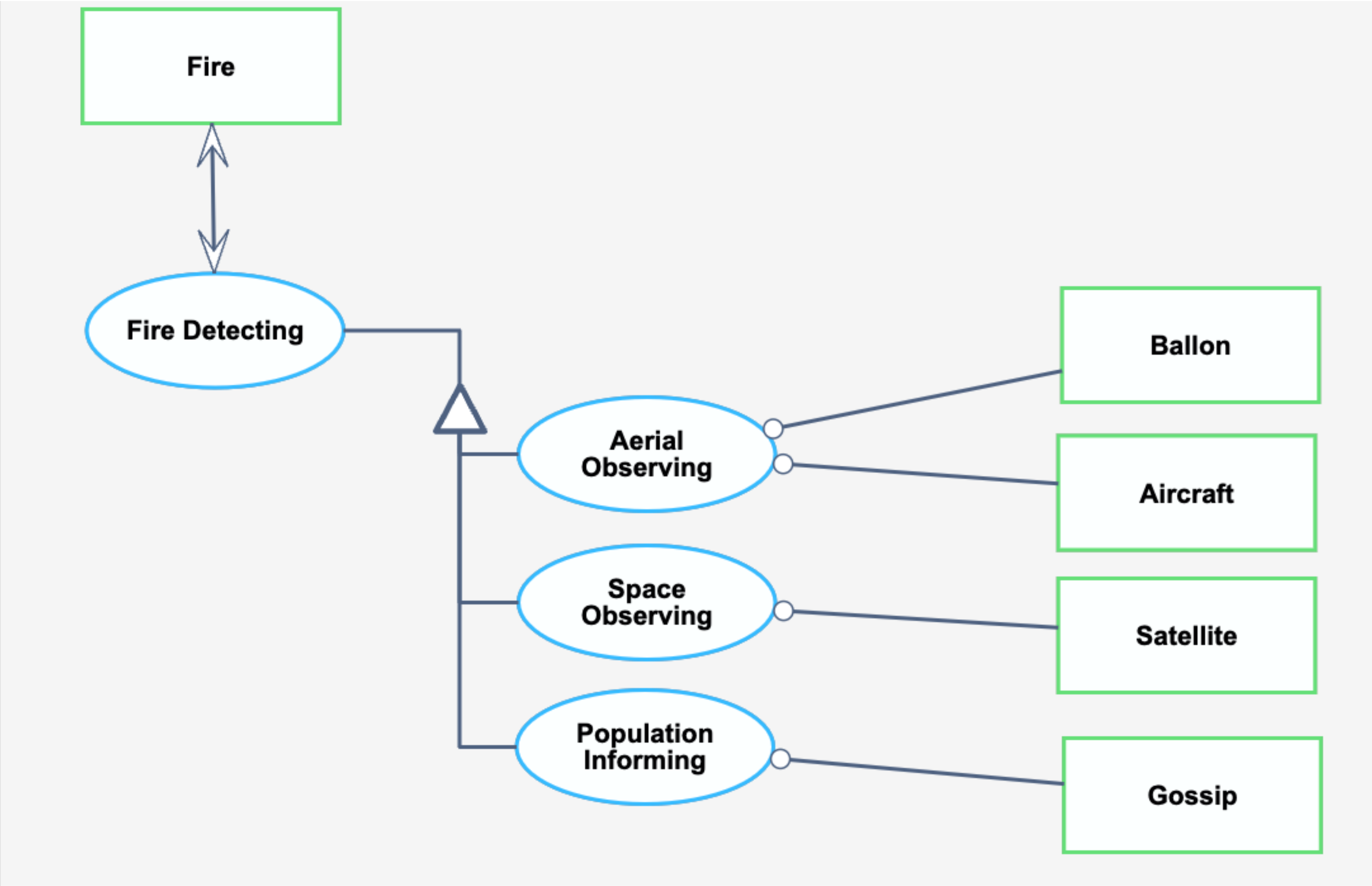
Criteria	Wt	Vendor 1	Vendor 3	Vendor 4
		Alternatives		
Cost	.30	Datum	+	S
Response time	.17		+	+
Training time	.17		-	S
Ease of use	.17		+	+
Strong team	.10		-	-
Team experience	.10		S	-
Pluses	1.0		3	2
Minuses			2	2
Overall total			+1	0
Weighted total			+.37	+.14

Figure 2.3: The two forms of a Decision Matrix.

- Means of scoring each alternative concept on its ability to meet a set of criteria (MoEs).
- By comparing the scores, you develop insight into the best alternatives and the most useful information to make your decision.


DCA-400-6
System Element Requirements
ROP
Conceptual Alternatives
Ideation
Solution Neutral Architecture
Logical Architecture
Concept Simulation
SIL
PIL
Flowed-down Requirements
RTLI
Benchmark

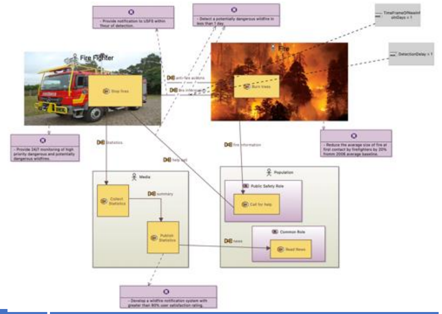
INTENTION EXPLORATION



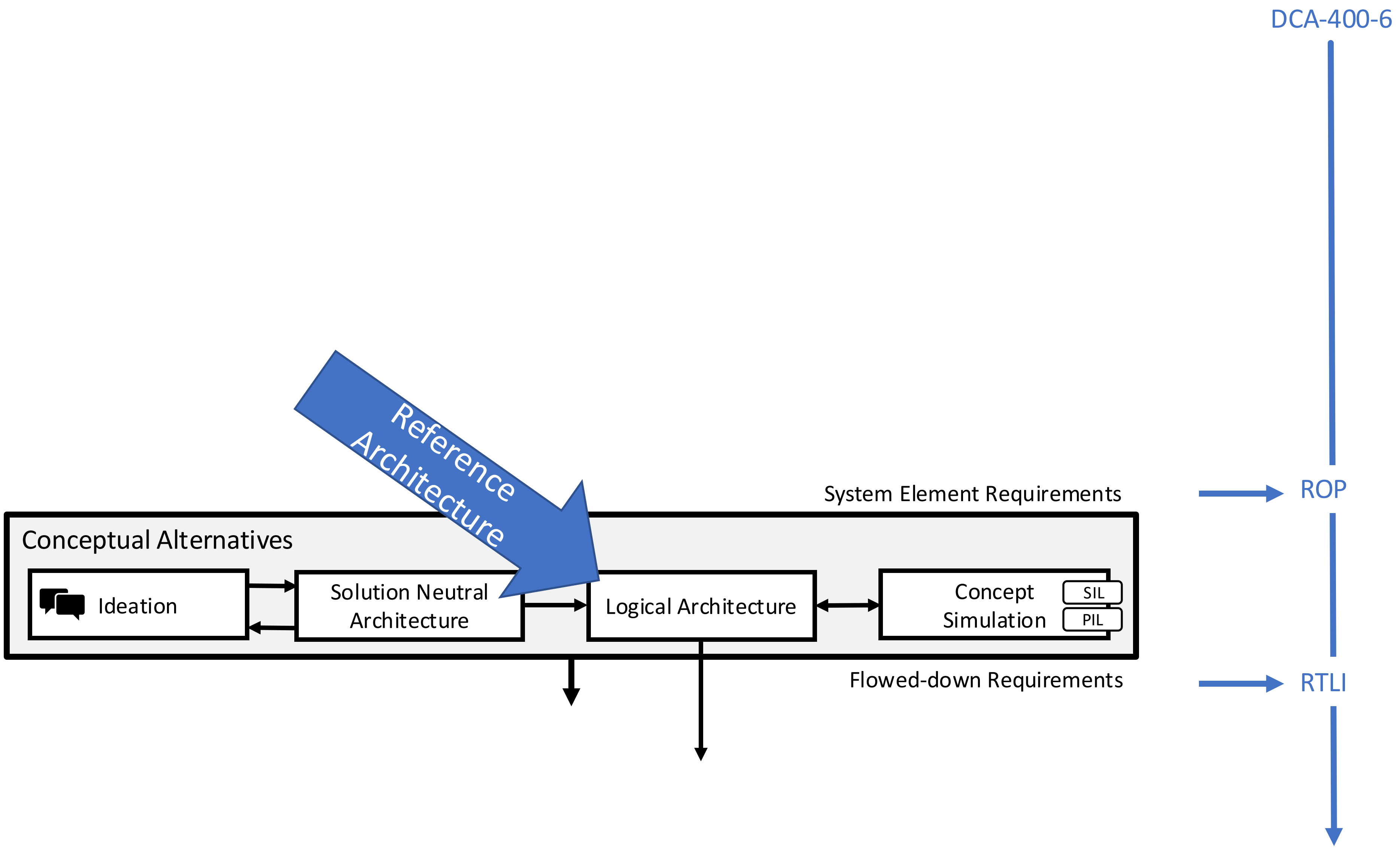
OPL

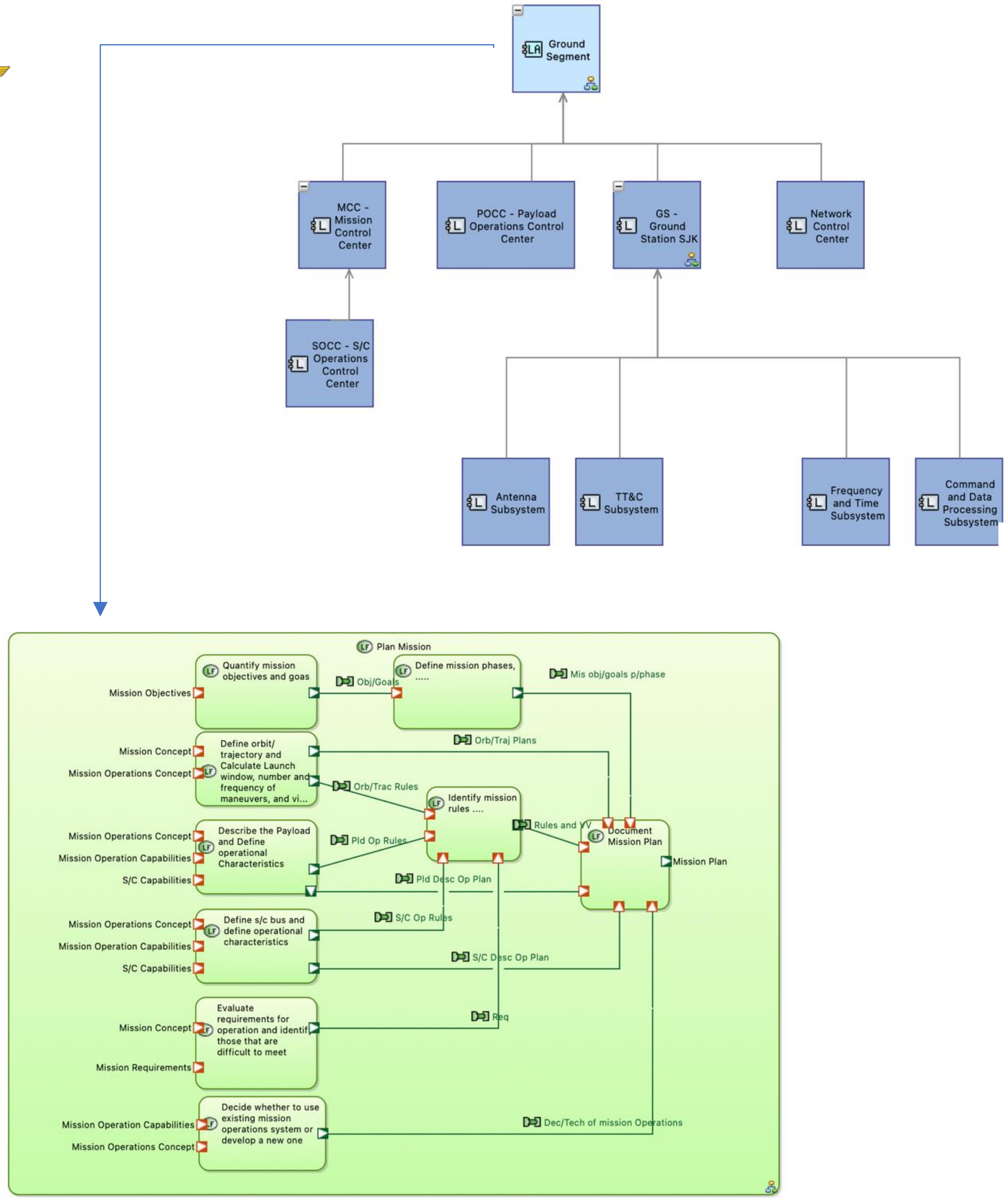
1. Aerial Observing, Population Informing and Space Observing are Fire Detecting.
2. Fire Detecting affects Fire.
3. Aerial Observing requires Aircraft and Ballon.
4. Space Observing requires Satellite.
5. Population Informing requires Gossip.

trading

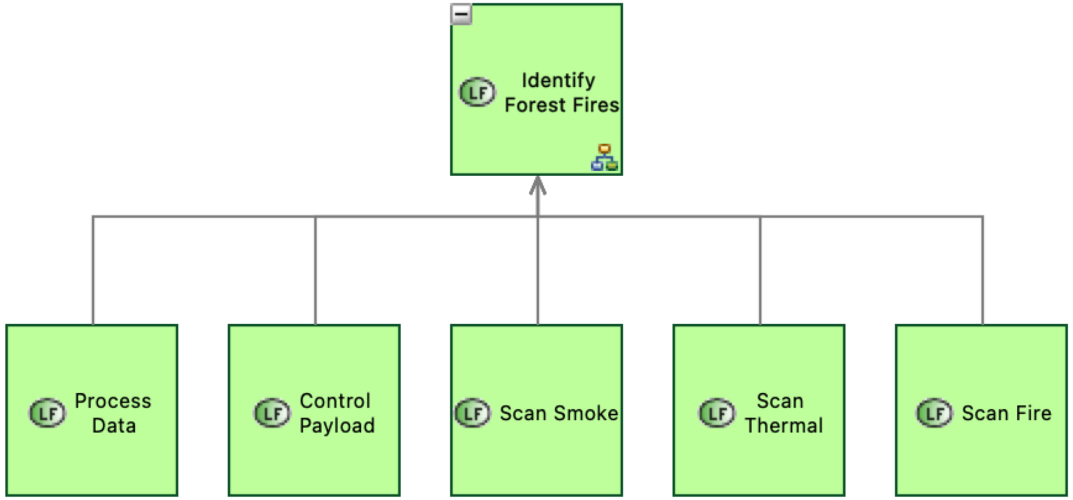


		Ballon	Aircraft	Satellite	Gossip
TimeFrameOfNewInformation	.4	-	+	+	0
DetectionDelay	.6	+	-	+	0
	Total	0	0	2	
	Weighted	.6	.4	1	

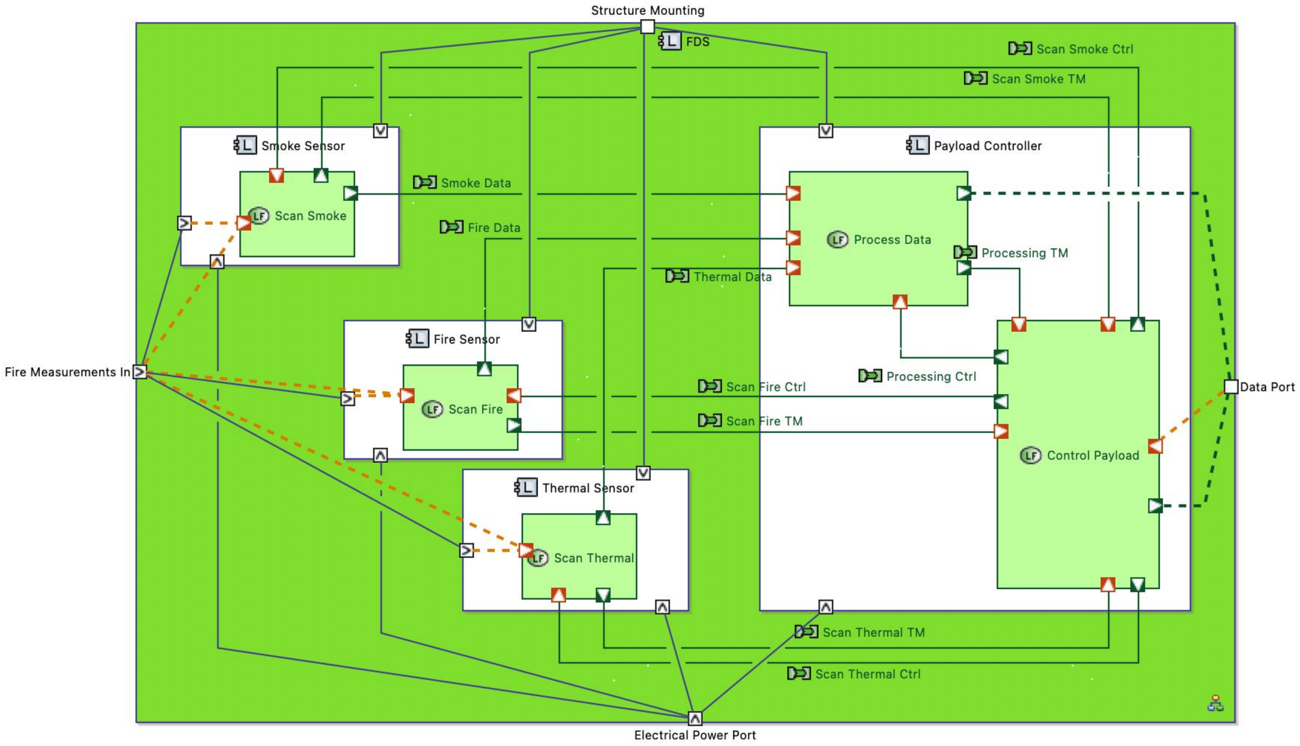

DCA-400-6
Reference Architecture
System Element Requirements
ROP
Conceptual Alternatives
Ideation
Solution Neutral Architecture
Logical Architecture
Concept Simulation
SIL
PIL
Flowed-down Requirements
RTLI



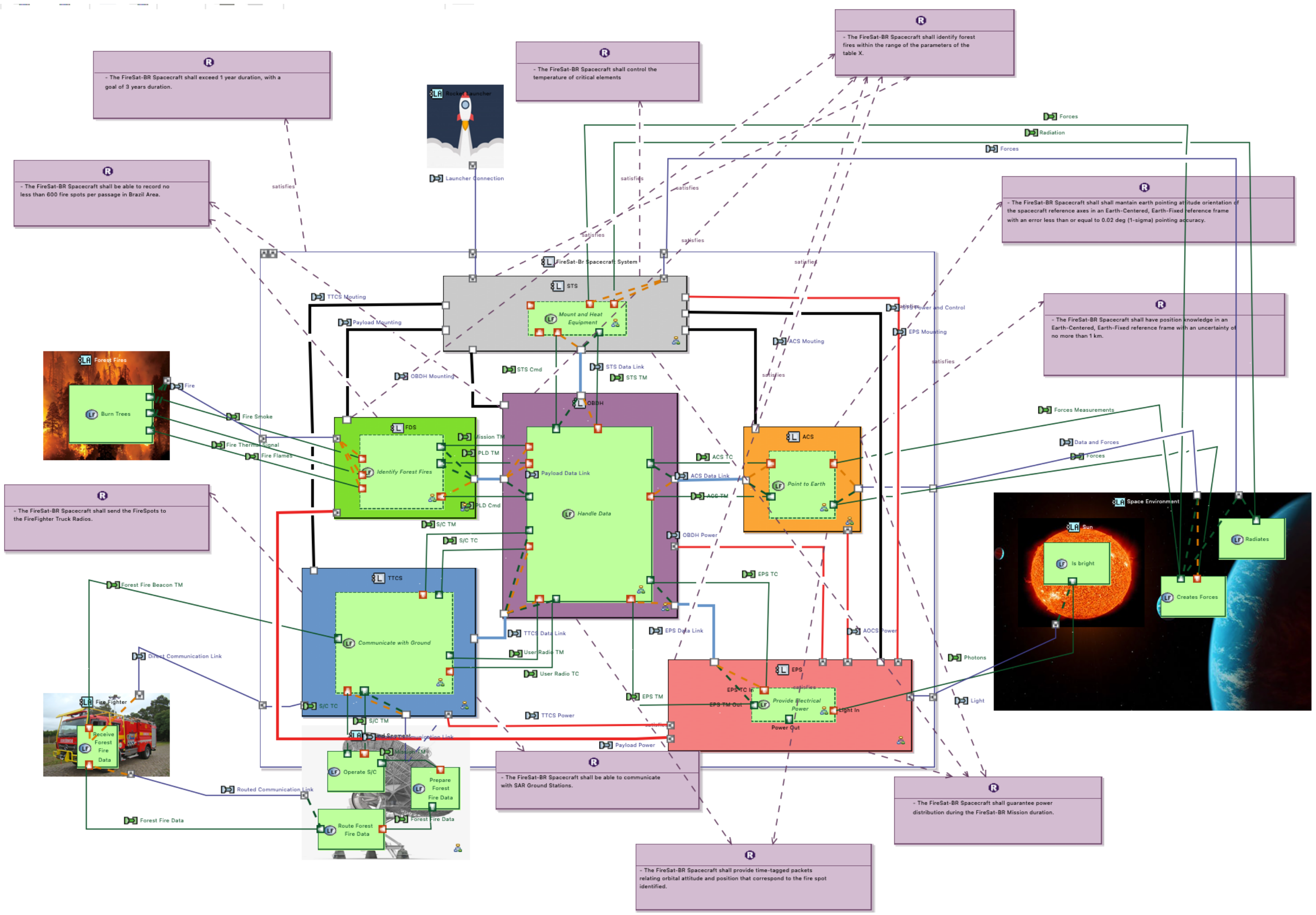
Ground Segment
MCC - Mission Control Center
POCC - Payload Operations Control Center
GS - Ground Station SJK
Network Control Center
SOCC - S/C Operations Control Center
Antenna Subsystem
TT&C Subsystem
Frequency and Time Subsystem
Command and Data Processing Subsystem
Plan Mission
Quantify mission objectives and goas
Define mission phases, ...
Mission Objectives
Obj/Goals
Mis obj/goals p/phase
Orb/Traj Plans
Mission Concept
Mission Operations Concept
Define orbit/ trajectory and Calculate Launch window, number and frequency of maneuvers, and vi...
Orb/Trac Rules
Identify mission rules ....
Rules and VV
Document Mission Plan
Mission Plan
Mission Operation Capabilities
S/C Capabilities
Describe the Payload and Define operational Characteristics
Pld Op Rules
Pld Desc Op Plan
Define s/c bus and define operational characteristics
S/C Op Rules
S/C Desc Op Plan
Evaluate requirements for operation and identify those that are difficult to meet
Req
Mission Requirements
Decide whether to use existing mission operations system or develop a new one
Dec/Tech of mission Operations



Identify Forest Fires
Process Data
Control Payload
Scan Smoke
Scan Thermal
Scan Fire

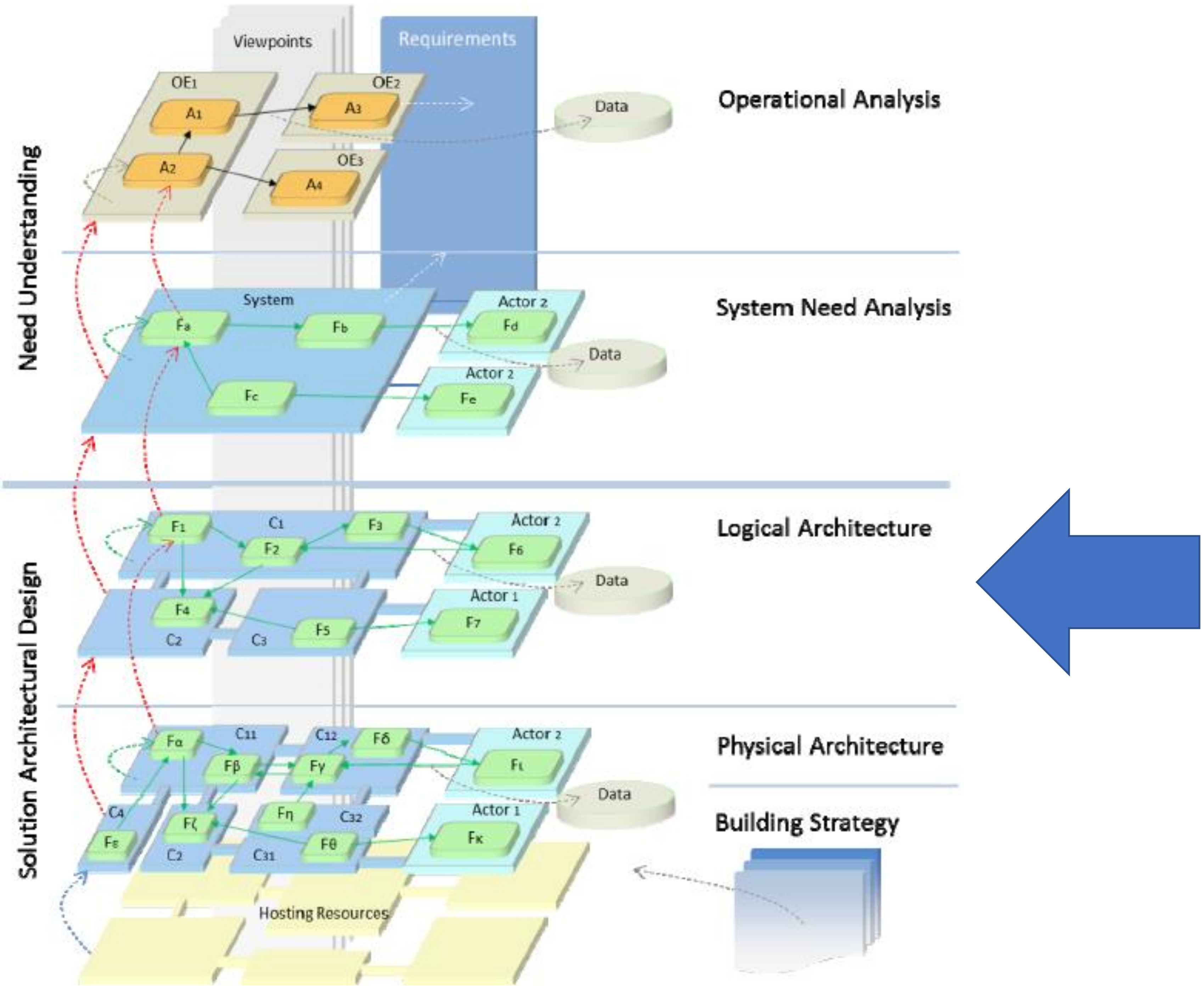


Structure Mounting
FDS
Scan Smoke Ctrl
Scan Smoke TM
Smoke Sensor
Scan Smoke
Smoke Data
Payload Controller
Process Data
Fire Data
Processing TM
Thermal Data
Fire Sensor
Scan Fire
Processing Ctrl
Fire Measurements In
Scan Fire Ctrl
Scan Fire TM
Data Port
Control Payload
Thermal Sensor
Scan Thermal
Scan Thermal TM
Scan Thermal Ctrl
Electrical Power Port



Logical Architecture
- [Capella Module]
  - System Requirements
    - [SYS-XXX] The FireSat-BR Spacecraft shall guarantee powe
      - [IE PUID] SYS-XXX
      - [Rationale] null
      - [VV Method] null
      - [VV Success Criteria] null
      - [VV Phase] null
      - [VV Procedure] null
      - [VV Report] null
    - [SYS-XXX] The FireSat-BR Spacecraft shall shall mantain ea
    - [SYS-XXX] The FireSat-BR Spacecraft shall be able to recor
    - [SYS-XXX] The FireSat-BR Spacecraft shall be able to comr
    - [SYS-XXX] The FireSat-BR Spacecraft shall exceed 1 year d
    - [SYS-XXX] The FireSat-BR Spacecraft shall control the tem
    - [SYS-XXX] The FireSat-BR Spacecraft shall send the FireSp
    - [SYS-XXX] The FireSat-BR Spacecraft shall identify forest f
    - [SYS-XXX] The FireSat-BR Spacecraft shall provide time-ta
    - [SYS-XXX] The FireSat-BR Spacecraft shall have position kı
  - []

LOGICAL ARCHITECTURE


Viewpoints
Requirements
OE1
OE2
OE3
A1
A2
A3
A4
Data
Operational Analysis
Need Understanding
System
Fa
Fb
Fc
Actor 2
Fd
Fe
System Need Analysis
C1
C2
C3
F1
F2
F3
F4
F5
F6
F7
Actor 1
Logical Architecture
Solution Architectural Design
C11
C12
C4
C32
C31
Physical Architecture
Building Strategy
Hosting Resources

WHAT IS IN THE LOGICAL ARCHITECTURE (LA)?

Logical Architecture

"How the system will work to meet expectations"

"How the system will work to fulfill expectations"

- In response to the need expressed by the two previous perspectives, it enables the **first major choices of solution** design, first via an internal functional analysis of the system: it describes the functions to be performed and assembled in order to implement the service functions identified in the previous phase. It continues with the **identification of the operational components** implementing these solution functions, integrating the non-functional constraints that we chose to be addressed at this level.

- The level of Logical Architecture aims to identify **Logical Components inside the System** ("how the system will work to fulfill expectations"), their relations and their content, **independently of any considerations of technology or implementation**.
- Next an **internal functional analysis** of the system must be carried out: **the subfunctions required to carry out the System Functions** chosen during the previous phase must be identified; next, **a split into Logical Components to which these internal subfunctions will be allocated must be determined**, all the while integrating the nonfunctional constraints that have been chosen for processing at this level

- The definition of the LA (an activity often – and wrongly – designated “logical architecture” for convenience) **consists mainly of a comparison between the needs expressed in previous perspectives, a functional analysis** describing the system behavior chosen to satisfy requirements, and a **structural analysis intended to identify the components that will constitute the system**, taking the chosen constraints and structuring principles into account.
- The LA is therefore a **first general vision**, moderately detailed, somehow an abstraction, of what the architecture of the system will be

The main activities for the definition of the logical principle architecture are as follows:

- to define the **factors** impacting the architecture and analysis viewpoints;
- to define the principles underlying the **system behavior**;
- to build **component-based system** structuring alternatives;
- to select the **architecture alternative** offering the best compromise.

Definition of the factors impacting the architecture and analysis viewpoints

- Any **properly designed architecture satisfies several expectations and constraints of various kinds**, which constrain and influence or even direct its definition, and whose satisfaction should be verified as early as possible to minimize possible subsequent resumption costs.

- These factors that constrain the architecture depend largely on each domain, and each profession. As examples we mention: delivered services and costs of course, expected performance, safety of operations, privacy, ease of maintenance, life duration, energy or logistical footprint, availability, product policy, scalability, but also more "aesthetic" considerations such as customer satisfaction.

- **For each factor previously identified**, the associated constraints (especially nonfunctional and performance ones), which can be applied to the needs and the solution, **must be expressed and quantified by metrics**; **each candidate architecture will be analyzed according to this viewpoint**, to verify that good practice is correctly followed.
- These decisions reflect know-how, the craft, in addition to the creativity of the engineering team, and will guide the emergence of different alternatives as well as their comparison.
- **Imposed factors and design choices must be categorized by importance or priority**, in order to be able to arbitrate between them when they result in antagonistic properties, or when certain constraints will have to be released to find an acceptable compromise.

- *In the case of the traffic control system, the* ***first impact factor is obviously the safety of goods and people****. An additional factor involves* ***system operators****, their training and their required skills, the scope of their responsibility and the role that must be assigned to them. We should also take into account factors such as* ***environmental conditions, life duration, constraints on logistics and maintenance****.*
- *In the case of the traffic control system, let us mention the required reliability rate and the system failure probability, the capability to be able to operate in the event of partial failure of certain subsystems; the maximal eligible number of operators; extreme temperature ranges, humidity, resistance to possible salt sprays; etc.*

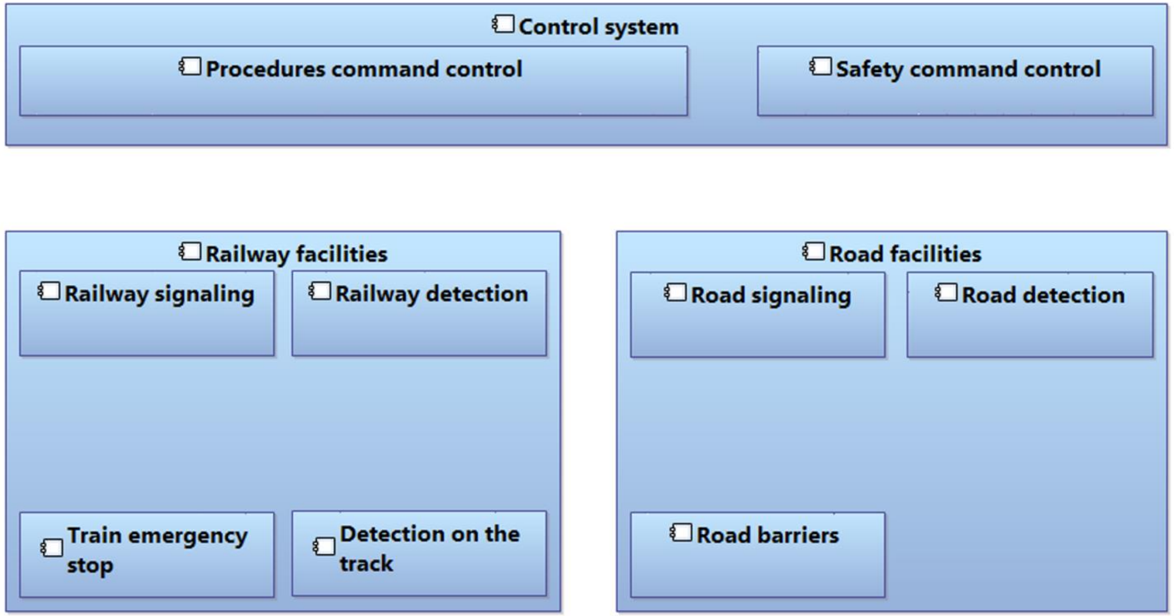
Definition of the behavior principles of the system

- The objective is to formalize the principles of the desired behavior of the system, and to non-functional, to which it has the responsibility to respond during its operation under operational conditions.
- *A common mistake consists of considering the behavior of the solution as a simple refinement of the previous functional expression of need at a finer level of detail. The solution design is much more than that: it is a take into account the constraints, namely "creative" definition effort of a behavior that meets the need (and that does not refine it),* detailing the processes and steps starting from the solicitations of the system, up to the provision of services, results or outputs, taking into account design decisions, mainly guided by the factors and constraints identified previously.

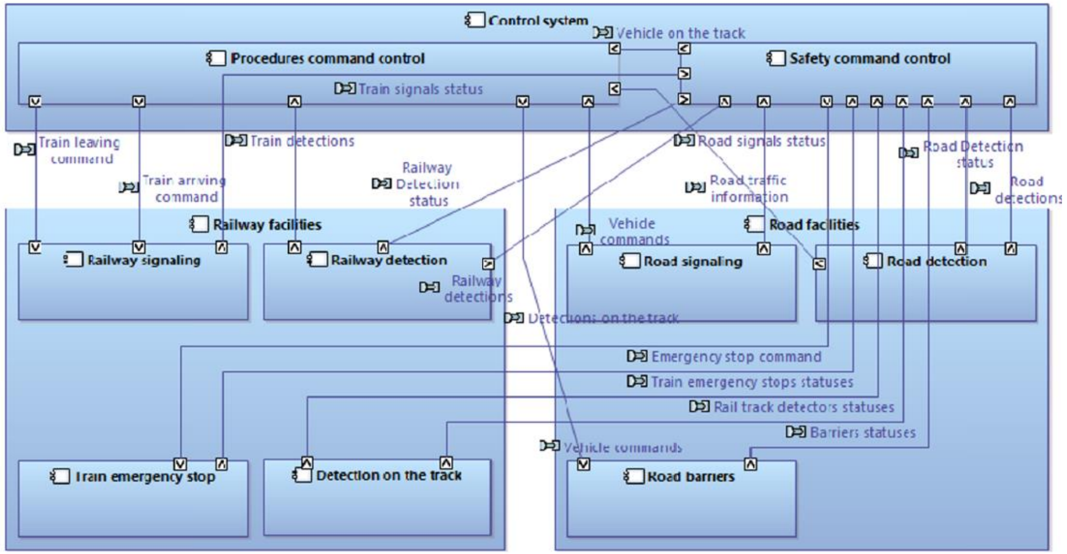
- 1 – identify and formalize **need items** captured. (tracing to SA)
- 2 – search for **possible functions** already in the LA that could also play a role to solve the need. (minimize functions)
- 3 – verify function **boundaries** to achieve what is expected of it.
  - Scenarios / chains will add light to design decisions or to the choice of product line.
- 4 – build a complete and coherent **global description** using the behavioral elements (scenarios/state machines)

Construction of component-based system structuring alternatives

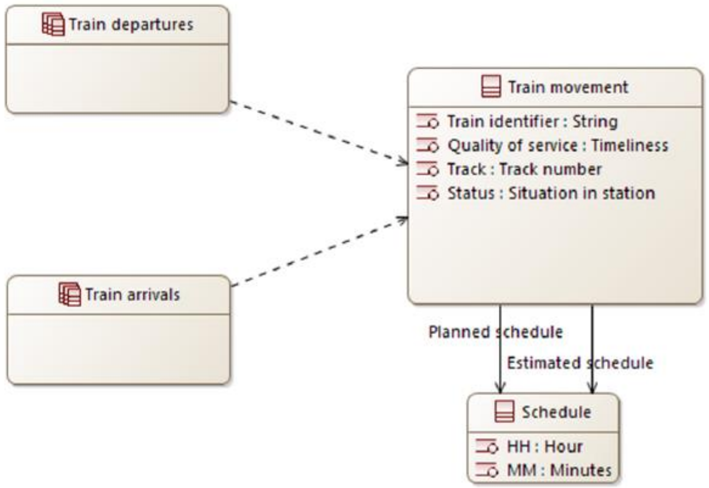
- This step should reveal a number of principle solutions, **describing the preliminary structure of the system**, built on the basis of the previous behavior, incorporating both non-functional associated constraints and the factors and design choices underlying it.
- The system is **broken down into principle components called logical components**. The term "component" is understood here in the general sense, as a constituent of the system at this level; it can later be implemented as a subsystem (or several), equipment, one or more mechanical parts or assemblies, one or more electronic cards, a software program itself eventually distributed or even a human contributor.

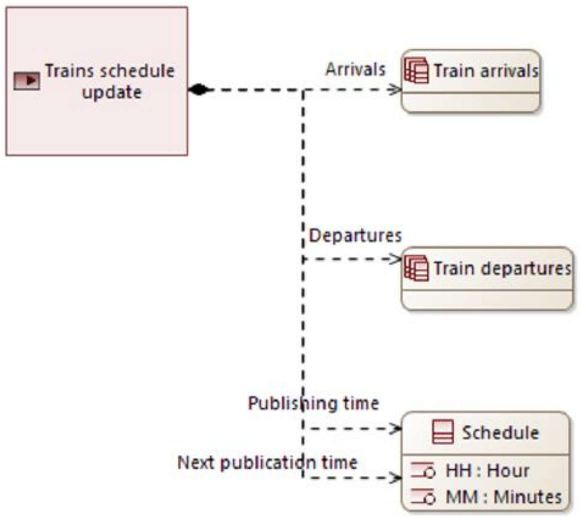


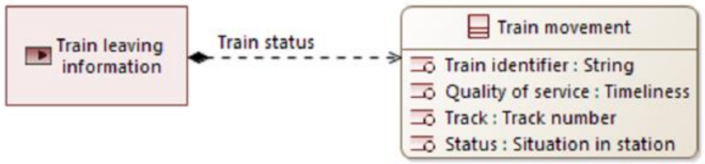
- The component building process consists of **grouping together or segregating the behavior functions** previously defined, according to the constraints and criteria imposed, in grouping sets that thus constitute the components. These latter can themselves be structured by subcomponents, according to the same types of criteria if necessary.
- It is recommended to submit each choice of functional grouping to the **multi-viewpoint analysis**



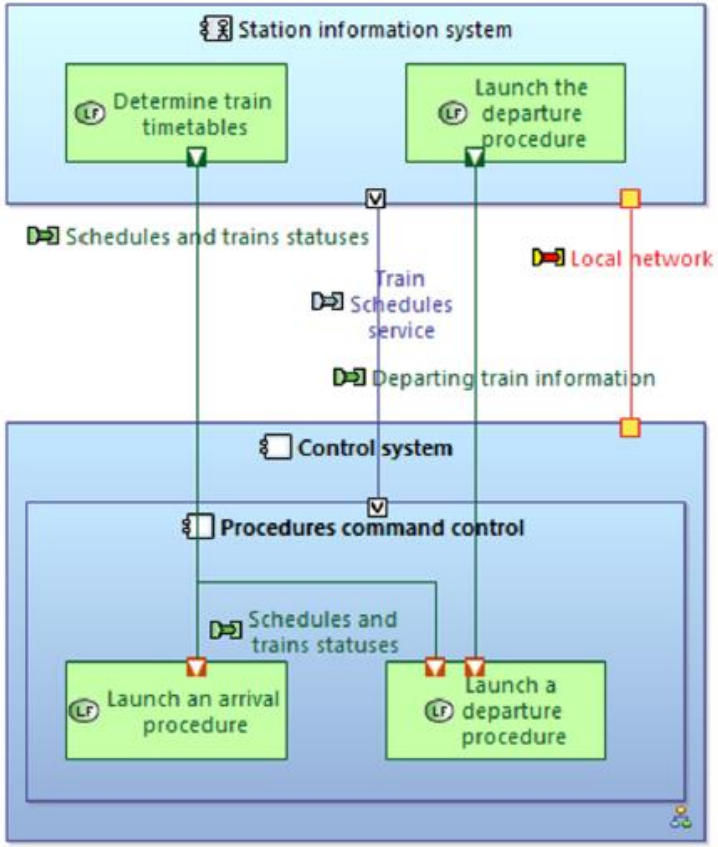
- The (preliminary) definition of interfaces between components (or with external actors) can be done at this level (or be postponed until the definition of the physical architecture): they are built based on the **functional exchanges linking the functions allocated to these components or actors, and exchanges data (and exchange elements) that these exchanges convey**; data and exchanges are mainly grouped according to semantic proximity or usage considerations.
- The actual exchanges between components are also achieved by way of grouping functional exchanges; combined with the capability to hide subcomponents in order to consider those of first level only, this also constitutes a level of synthesis or even of abstraction able to hide the complexity of functional exchanges, and to reason on several levels of detail.

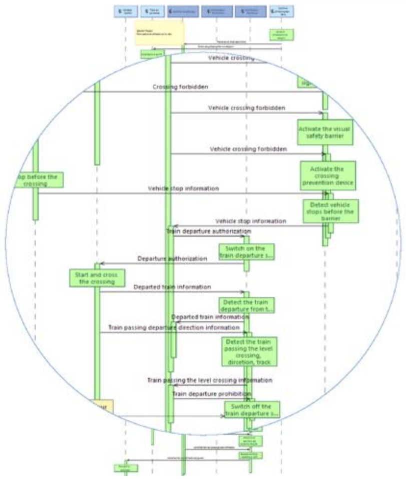


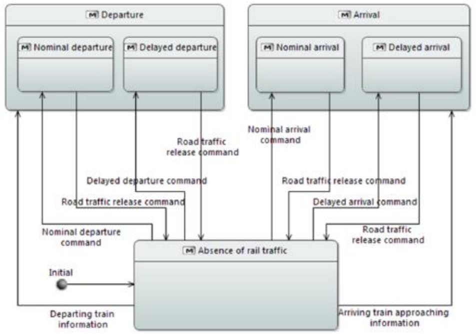




- This static definition of interfaces most often must be accompanied by a **dynamic definition**, by creating scenarios at the boundaries of the components, and if necessary, state and modes machines associated with each contributor to exchanges and managing this dynamics of interfaces.
- Furthermore, **states and modes can be defined** and allocated to components, based on those implemented at the system level in the previous behavioral functional analysis, and consistent with them.





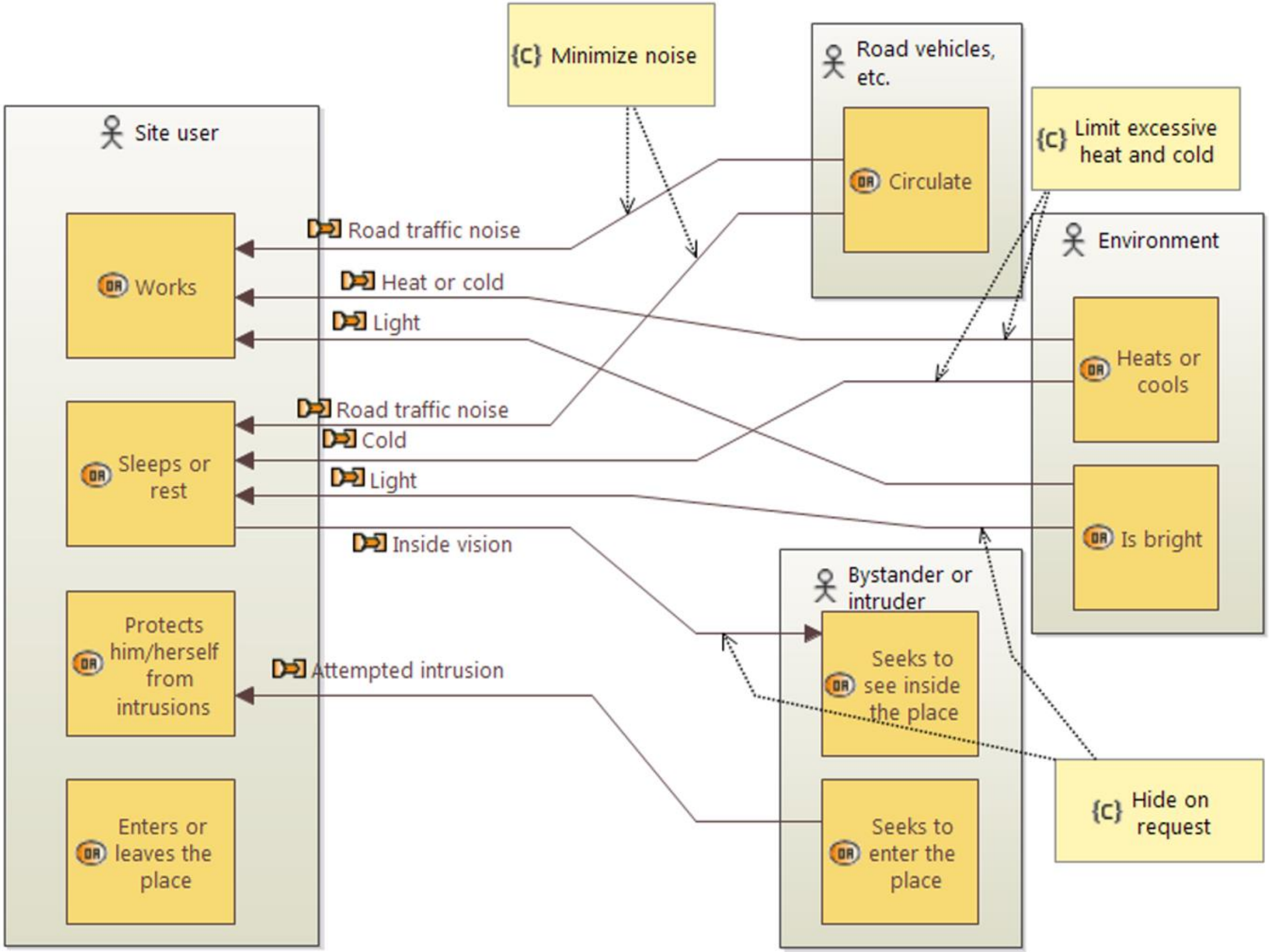


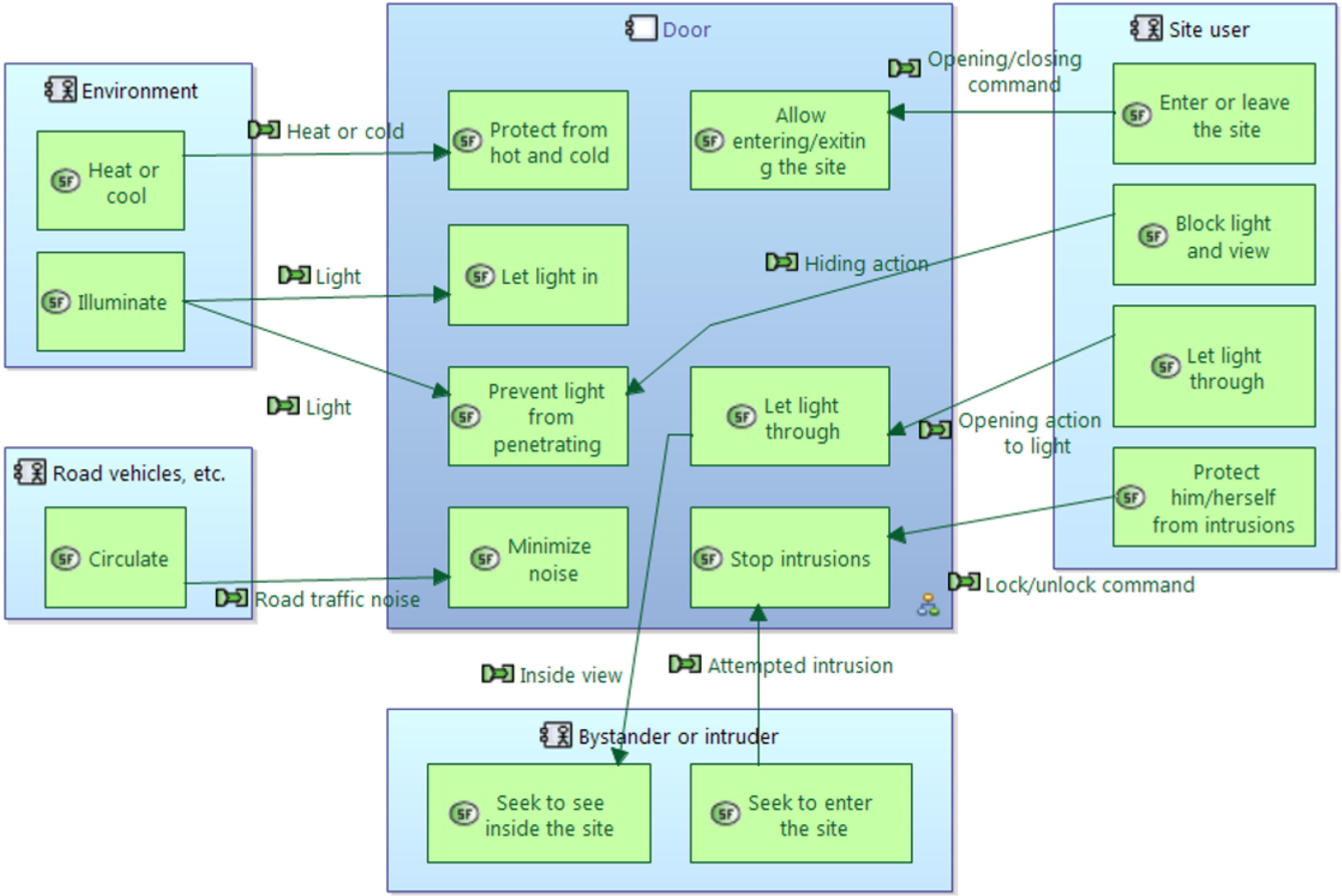
Selection of the architecture alternative offering the best trade-off

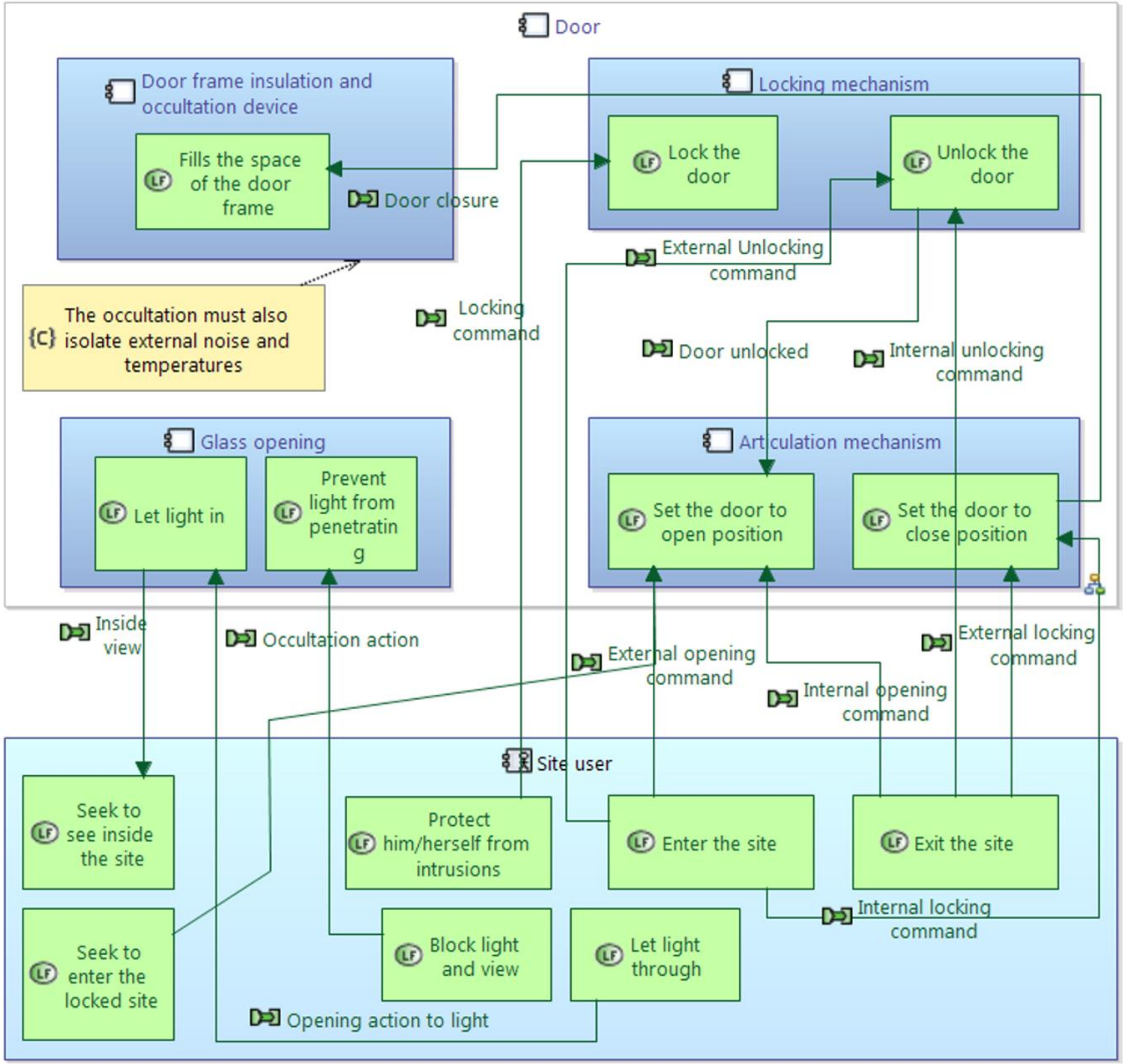
- The purpose of this activity is to find among previous candidate architectures the one that **represents the best trade-off** with respect to all viewpoints under consideration, and to justify its compliance to the need.
- Each alternative has in principle been evaluated based on the major viewpoints impacting it – and their relative importance – during its definition; the inadmissible nonconformities have been eliminated, but as the evaluation is rarely binary, the point is therefore now to **compare the "merits" of each candidate in a multi-criteria quantitative analysis**, of which previously identified viewpoint analyses, priorities and metrics are key elements.

Arcadia method – LOGICAL ARCHITECTURE analysis summary

Define the factors impacting the architecture and analysis viewpoints	satisfies a number of expectations and constraints of various kinds, which constrain and influence or even direct its definition, and whose satisfaction should be verified as early as possible to minimize possible subsequent resumption costs.
Define the principles underlying the system behavior	non-functional, to which it has the responsibility to respond during its operation under operational conditions
Build component-based system structuring alternatives	The system is broken down into principle components called logical components. The term “component” is understood here in the general sense, as a constituent of the system at this level;
Select the architecture alternative offering the best compromise	find among previous candidate architectures the one that represents the best trade-off with respect to all viewpoints under consideration, and to justify its compliance to the need.


{C} Minimize noise
Road vehicles, etc.
Circulate
{C} Limit excessive heat and cold
Site user
Works
Road traffic noise
Heat or cold
Light
Environment
Heats or cools
Sleeps or rest
Road traffic noise
Cold
Light
Is bright
Inside vision
Bystander or intruder
Protects him/herself from intrusions
Attempted intrusion
Seeks to see inside the place
Enters or leaves the place
Seeks to enter the place
{C} Hide on request


Door
Environment
Heat or cool
Illuminate
Heat or cold
Light
Light
Protect from hot and cold
Let light in
Prevent light from penetrating
Minimize noise
Allow entering/exiting the site
Let light through
Stop intrusions
Opening/closing command
Hiding action
Opening action to light
Lock/unlock command
Site user
Enter or leave the site
Block light and view
Let light through
Protect him/herself from intrusions
Road vehicles, etc.
Circulate
Road traffic noise
Inside view
Attempted intrusion
Bystander or intruder
Seek to see inside the site
Seek to enter the site


Door
Door frame insulation and occultation device
Fills the space of the door frame
Door closure
Locking mechanism
Lock the door
Unlock the door
External Unlocking command
The occultation must also isolate external noise and temperatures
Locking command
Door unlocked
Internal unlocking command
Glass opening
Let light in
Prevent light from penetratin g
Articulation mechanism
Set the door to open position
Set the door to close position
Inside view
Occultation action
External opening command
External locking command
Internal opening command
Site user
Seek to see inside the site
Protect him/herself from intrusions
Enter the site
Exit the site
Internal locking command
Seek to enter the locked site
Block light and view
Let light through
Opening action to light

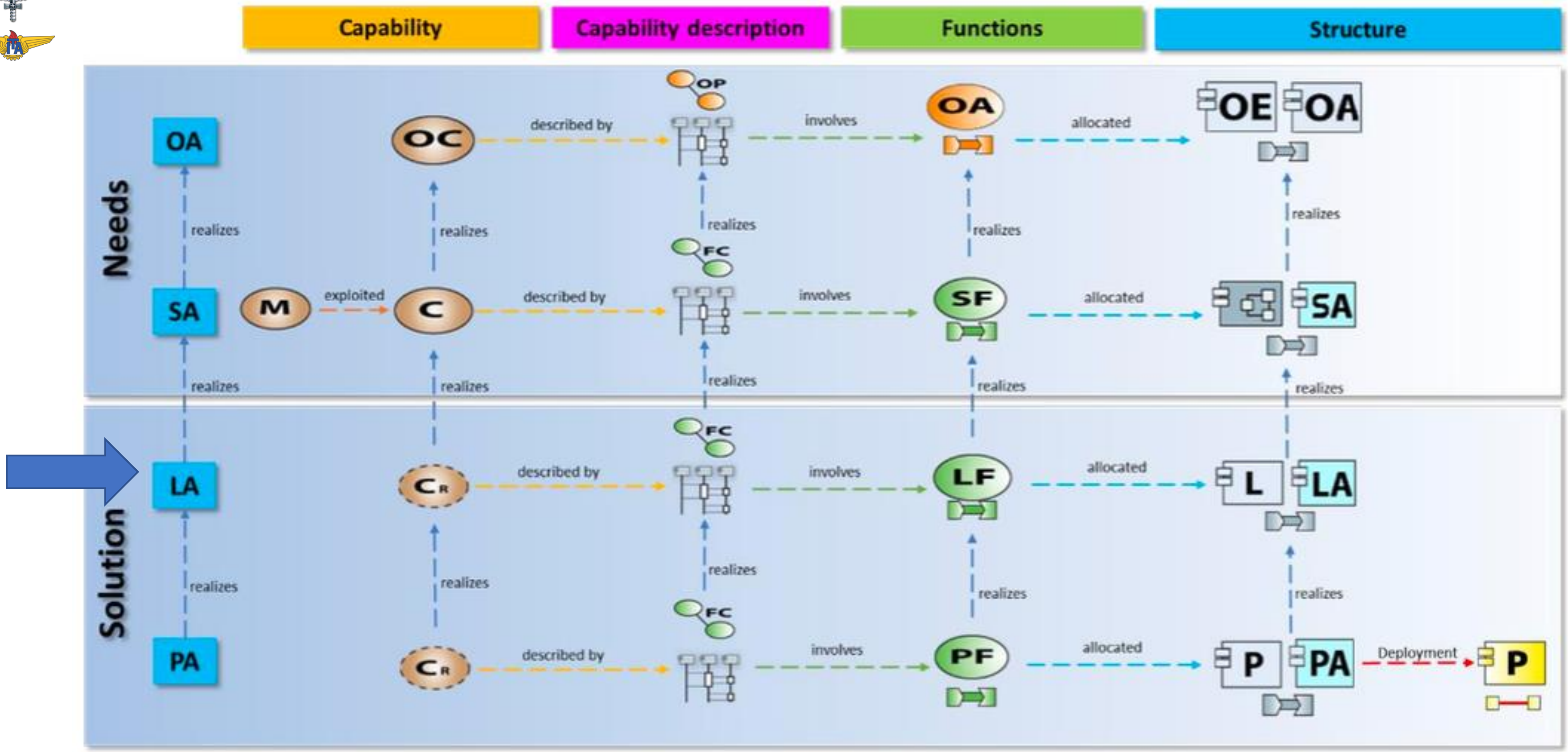


Figure 2.3: Arcadia ontology traceability

https://www.slideshare.net/HelderCastro3/mbse-with-arcadia-methodpdf-256664096

Arcadia layer	Requirements	Capability	Capability description	Functional	Structure	Modes and States	Data	Interfaces
Operational Analysis	**R-OA** Capture stakeholder requirements	**OA1** Define Operational Capabilities	**OA2** Define processes and scenarios	**OA3** Define Operational Activities and interactions	**OA4** Capture Operational Entities and Actors. Allocate Operational Activities to Operational Actors, Entities	**M&S-OA5** Define operational modes and states	**D-OA6** Define operational data model	**I-OA7** Define interfaces and describe interfaces scenarios
System Analysis	**R-SA** Derive Stakeholder requirements and capture System requirements	**SA1** Define System Missions and System Capabilities	**SA2** Define Functional Chains and Scenarios.	**SA3** Define System Functions. Define Functional Exchanges and components	**SA4** Allocate System Functions to System and Actors	**M&S-SA5** Define system modes and states	**D-SA6** Define system data model	**I-SA7** Define interfaces and describe interfaces scenarios Enrich Logical Scenarios.
Logical Architecture	**R-LA** Derive system requirements and Capture components requirements	**LA1** Transition Capabilities Realization from system layer	**LA2** Define Functional Chains and scenarios	**LA3** Derive System Functions and define Logical Functions. Define Functional Exchanges and components.	**LA4** Allocate Logical Functions to Logical Components	**M&S-LA5** Define logical components modes and states	**D-LA6** Define logical data model	**I-LA7** Delegate System Interfaces and create Logical Interfaces. Enrich Logical Scenarios.
Physical Architecture	**R-PA** Derive logical requirements and capture physical requirements	**PA1** Transition Capabilities Realization from logical layer	**PA2** Define Functional Chains, Scenarios, and Physical Path	**PA3** Derive Logical Functions and define Physical Functions. Define Functional Exchanges and components.	**PA4** Define Physical Nodes and refine Behavioural Physical Components. Allocate Behavioural Components.	**M&S-PA5** Define physical nodes modes and states	**D-PA6** Define physical data model	**I-PA7** Delegate Logical Interfaces and create Physical Interface. Enrich Physical Scenarios.

Table 3.2: Arcadia matrix activities

https://www.slideshare.net/HelderCastro3/mbse-with-arcadia-methodpdf-256664096

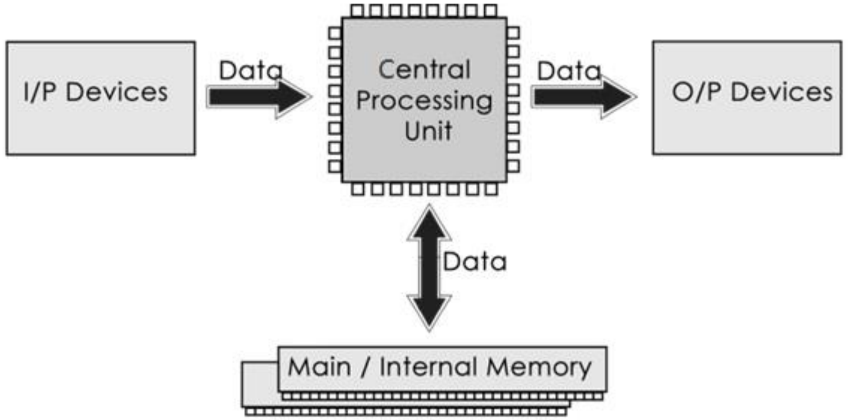
Arcadia layer	Requirements	Capability	Capability description	Functional	Structural	Modes and States	Data	Interfaces
Operational Analysis	**R-OA** No dedicated diagram	**OA1** [OCB] Operational Capabilities	**OA2** [OAS] Operational Activity Scenario [OPD] Operational Process Scenario [OES] Operational Entity Scenario	**OA3** [OABD] Operational Activity Breakdown Diagram [OAIB] Operational Activity Interaction Blank	**OA4** [OEBD] Operational Entities Blank Diagram [ORB] Operational Roles Blank [OAB] Operational Architecture Blank	**M&S-OA5** [MSM] Modes and States	**D-OA6** [CDB] Class Diagram	**I-OA7** [IDB] Interface Definition Blank [CEI] Component External Interfaces [IS] Interface Scenario [CDI] Component Detailed Interface
System Analysis	**R-SA** No dedicated diagram	**SA1** [MCB] Mission and Capabilities Blank [CC] Contextual Capability	**SA2** [FS] System Functional Scenario [ES] System Entity Scenario [SFCD] System Functional Chain Description	**SA3** [SFBD] System Functional Breakdown Diagram [SDFB] System Data Flow Blank	**SA4** [CSA] Contextual System Actor [SAB] System Architecture Blank	**M&S-SA5** [MSM] Modes and States	**D-SA6** [CDB] Class Diagram	**I-SA7** [IDB] Interface Definition Blank [CEI] Component External Interfaces [IS] Interface Scenario [CDI] Component Detailed Interface
Logical Architecture	**R-LA** No dedicated diagram	**LA1** [CRB] Capabilities Realization Blank [CRI] Contextual Capability Realization Involvement	**LA2** [FS] Logical Functional Scenario [ES] Logical Entity Scenario [LFCD] Logical Functional Chain Description	**LA3** [LFBD] Logical Functional Breakdown Diagram [LDFB] Logical Data Flow Blank	**LA4** [LCBD] Logical Component Breakdown Diagram [LAB] Logical Architecture Blank	**M&S-LA5** [MSM] Modes and States	**D-LA6** [CDB] Class Diagram	**I-LA7** [IDB] Interface Definition Blank [CEI] Component External Interfaces [IS] Interface Scenario [CDI] Component Detailed Interface
Physical Architecture	**R-PA** No dedicated diagram	**PA1** [CRB] Capabilities Realization Blank [CRI] Contextual Capability Realization Involvement	**PA2** [FS] Physical Functional Scenario [ES] Physical Entity Scenario [PFCD] Physical Functional Chain Description	**PA3** [PFBD] Physical Functional Breakdown Diagram [PDFB] Physical Data Flow Blank	**PA4** [PCBD] Physical Component Breakdown Diagram [PAB] Physical Architecture Blank	**M&S-PA5** [MSM] Modes and States	**D-PA6** [CDB] Class Diagram	**I-PA7** [IDB] Interface Definition Blank [CEI] Component External Interfaces [IS] Interface Scenario [CDI] Component Detailed Interface

Table 3.3: Arcadia diagrams matrix

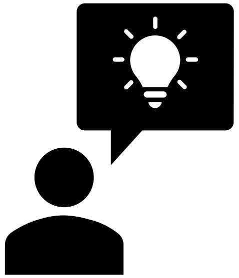
https://www.slideshare.net/HelderCastro3/mbse-with-arcadia-methodpdf-256664096

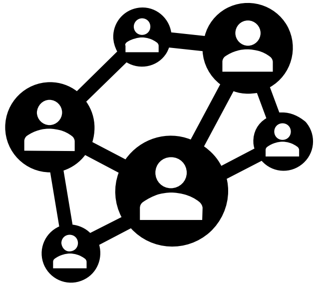
LOGICAL ARCHITECTURE CONCEPTS

- **Logical Component**: structural element within the System, with **structural Ports** to interact with the other Logical Components and the external Actors. **A Logical Component can have one or more Logical Functions**. It can also be subdivided into Logical subcomponents;

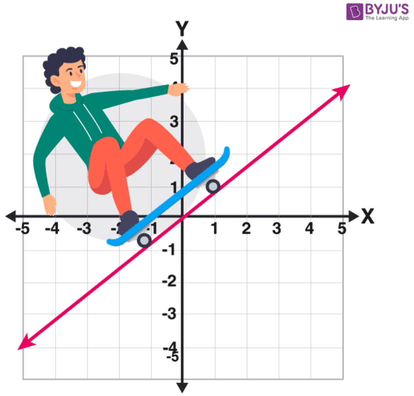


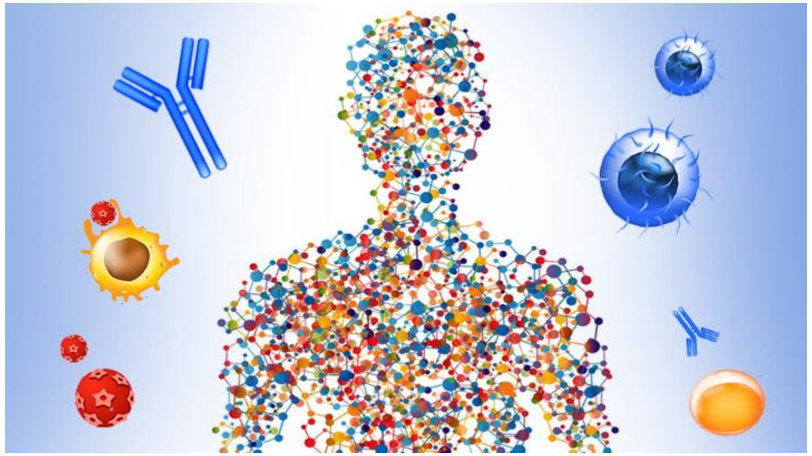
- **Logical Actor**: any element that **is external to the System** (human or non-human) and that interacts with it (for example Pilot, Maintenance operator, etc.).

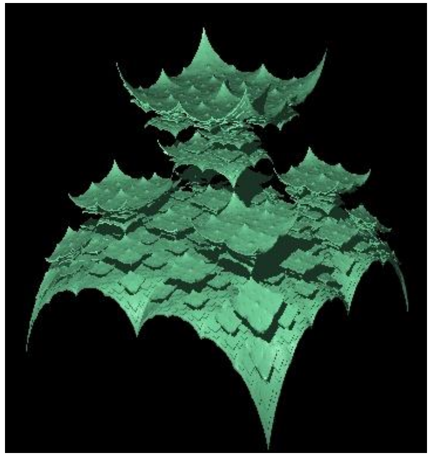





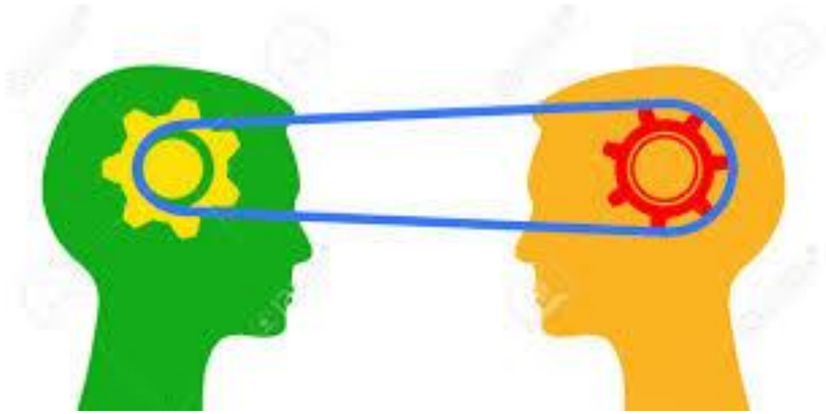

- **Logical Function**: behavior or service provided by a Logical Component or by a Logical Actor. A **Logical Function has Function Ports that allow it to communicate with the other Logical Functions**. A Logical Function can be subdivided into Logical subfunctions;

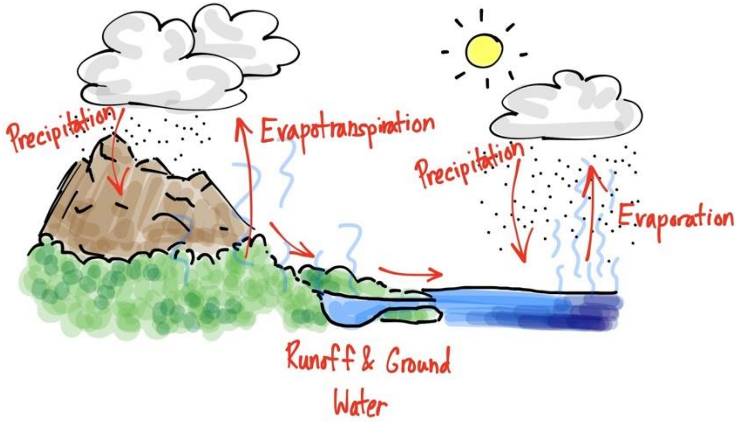




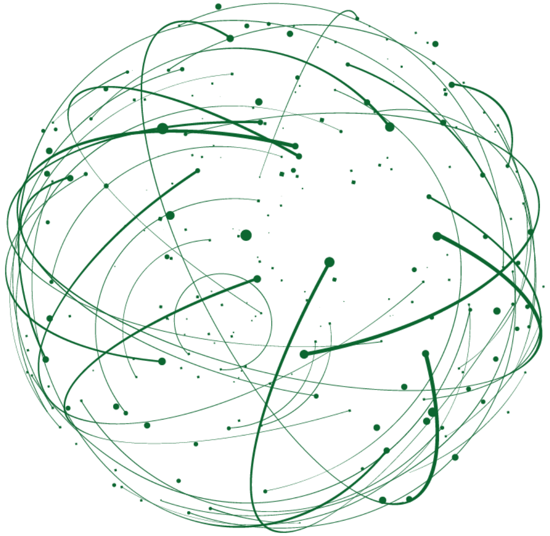


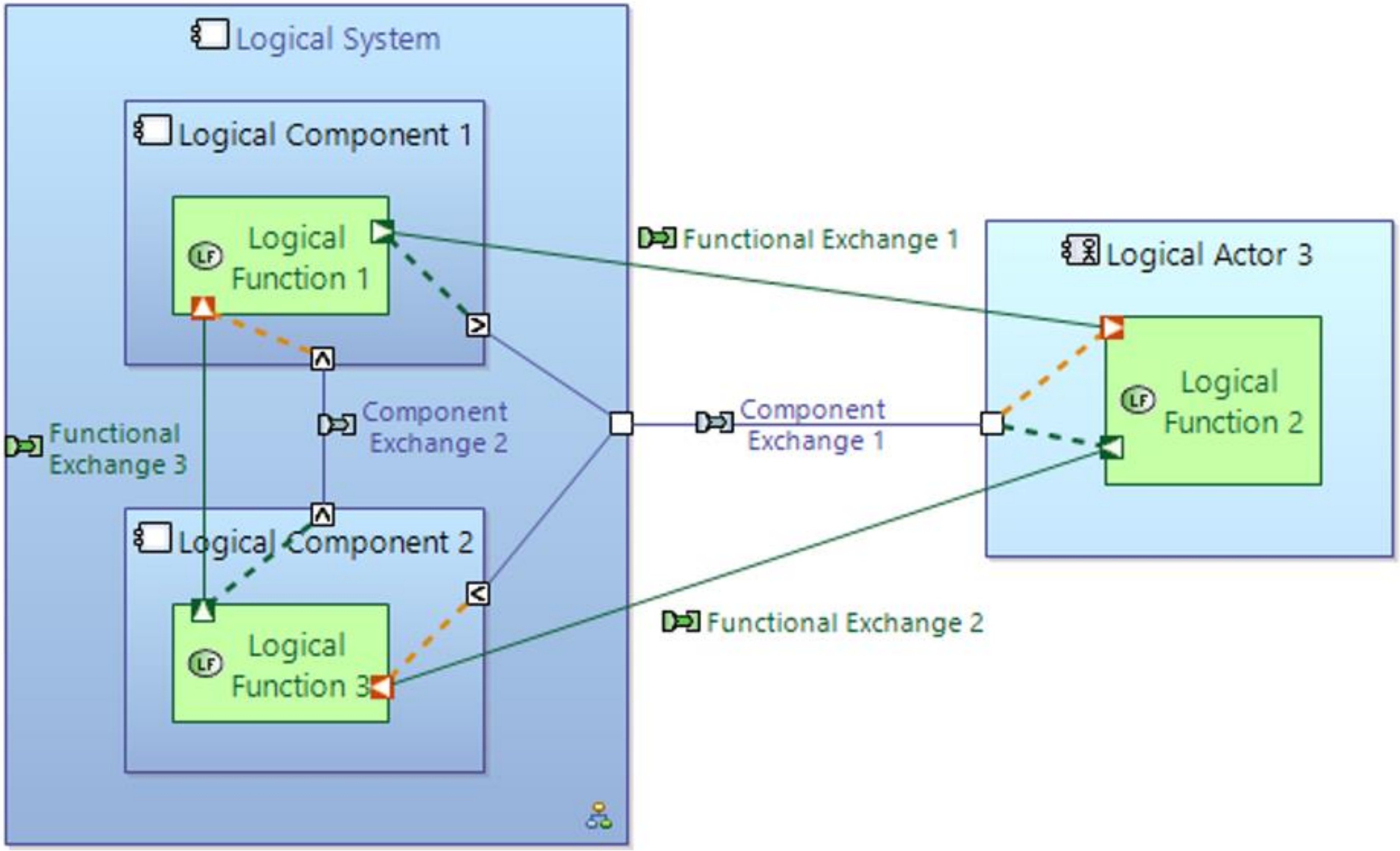
- **Functional Exchange**: a ***unidirectional*** exchange of information or matter between two Logical Functions, linking two Function Ports;





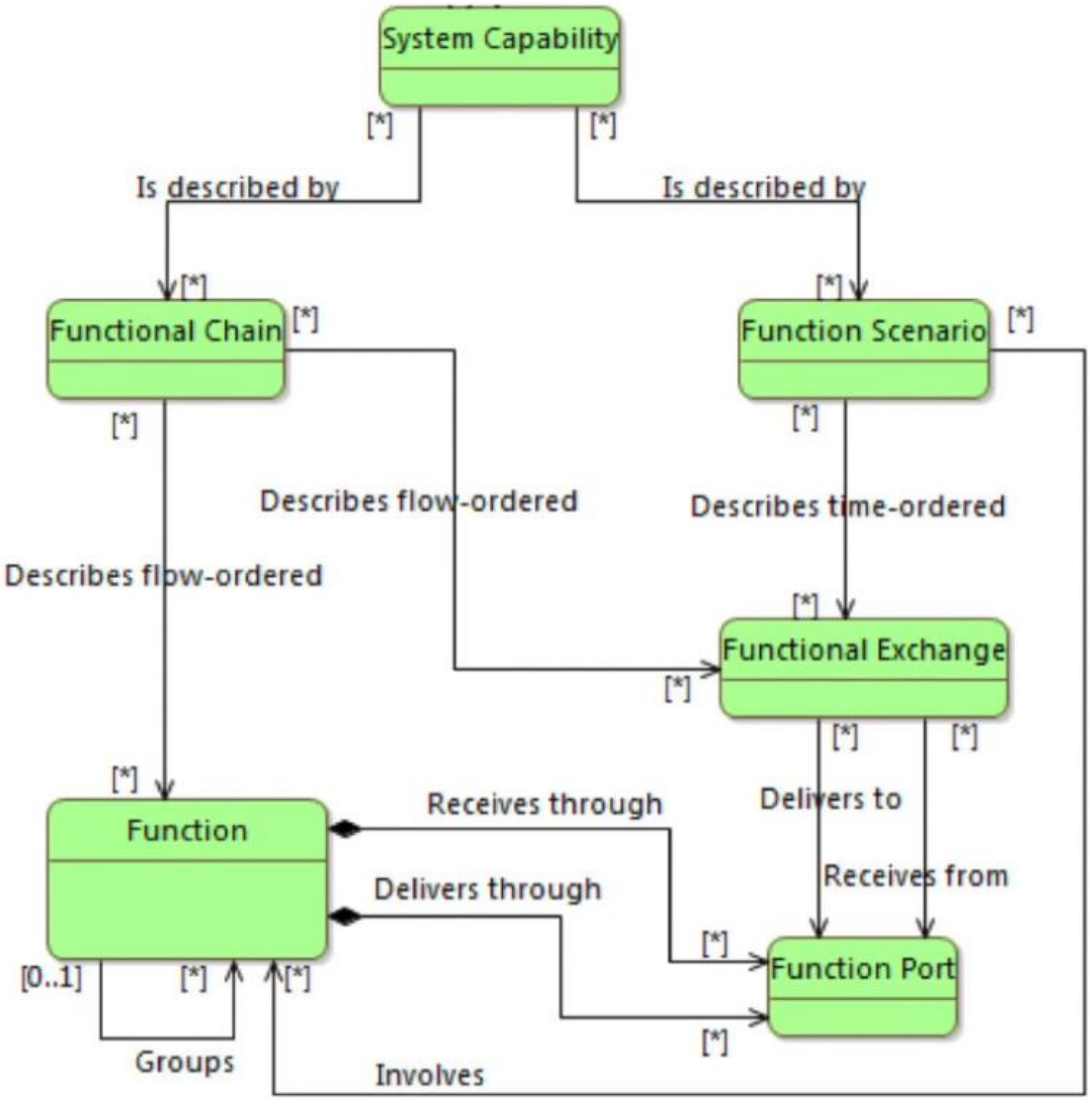
- **Component Exchange**: **connection between the Logical Components** and/or the Logical Actors, allowing **circulation of the Functional Exchanges**;




Logical System
Logical Component 1
Logical Function 1
Functional Exchange 3
Component Exchange 2
Logical Component 2
Logical Function 3
Functional Exchange 1
Component Exchange 1
Functional Exchange 2
Logical Actor 3
Logical Function 2

- **Logical Scenario**: **dynamic occurrence describing the interactions between Logical Components and Logical Actors** in the context of a Capability. It is commonly represented as a sequence diagram, with the vertical axis representing the time axis;

- **Functional Chain**: element of the model that enables a **specific path to be designated among all possible paths** (using certain Functions and Functional Exchanges). This is particularly useful for assigning constraints (latency, criticality, etc.), as well as organizing tests;



Arcadia conceptual metamodel

Functional chains with Arcadia and Capella_ Concepts and exploitation

LOGICAL ARCHITECTURE DIAGRAMS

Transition from System Functions

Perform an automated transition of System Functions

Create Traceability Matrix

Initialization and automated update of the logical functions according to the system functions

The transition tools create a first 1-1 traceability mapping between Logical Architecture and System Analysis. Use dedicated traceability matrices to modify the traceability relationships.

Refine Logical Functions, describe Functional Exchanges

[LFBD] Create a new Functional Breakdown diagram

[LDFB] Create a new Functional Dataflow Blank diagram

[FS] Create a new Functional Scenario

Enrich and details the functional breakdown with new logical functions.

Describe the data flows between logical functions and identify specific functional chains.

Define Logical Components and Actors

Perform an automated transition of System Actors

[LCBD] Create a new Logical Component Breakdown diagram

[LAB] Create a new Logical Architecture diagram

The initialization and automated updated of the logical actors can be performed according to system actors.

Use an architecture or breakdown diagram to describe the system internal building blocks from a logical point of view.

Logical components are intended to interact with each other to achieve the functional goals of the system.

Allocate Logical Functions to Logical Components

[LAB] Create a new Logical Architecture diagram

[ES] Create a new Exchange Scenario

Create a new allocation Logical Component / Logical Function Matrix

The logical components are responsible for implementing the logical functions. Manage these allocations using an architecture diagram and deduce component exchanges implementing the functional exchanges.
Create dataflows scenarios to illustrate functional exchanges between the components.

Delegate System Interfaces and create Logical Interfaces

[CII] Create a new Contextual Internal Interface diagram on the Logical System Component

Use the automated synchronization tool to initialize the root logical system according to the interfaces defined in System Analysis.

Delegate each system interface to one or more logical components. Create internal interfaces between subcomponents.

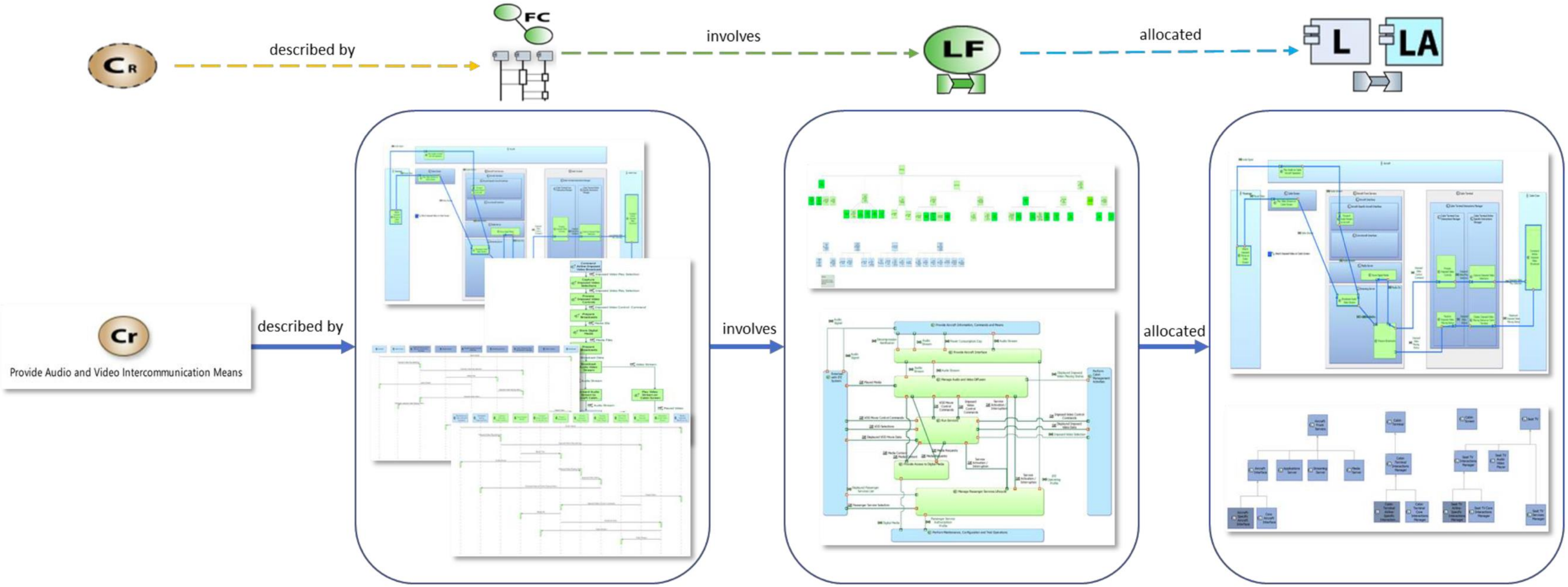
Enrich Logical Scenarios

Perform an automated transition of System Analysis Capabilities

[IS] Create a new Interface Scenario

Specify the dynamical behavior of the logical components by completing the interaction sequences coming from the System Analysis. The enrichment of the interaction sequences and the identification of the logical interfaces are two very tight and iterative activities.

The scenario refinement process is iterative, each update on a source can be automatically propagated to the target.



Capability diagrams	Capability description diagrams	Functional diagram	Structural diagrams
[CRB] Capabilities Realization Blank [CRI] Contextual Capability Realization Involvement	[FS] Logical Functional Scenario [ES] Entity Scenario [LFCD] Logical Functional Chain Description	[LFBD] Logical Functional Breakdown Diagram [LDFB] Logical Data Flow Blank	[LCBD] Logical Component Breakdown Diagram [LAB] Logical Architecture Blank

Figure 6.7: Logical Architecture model elements traceability and diagrams relationship

Final Considerations

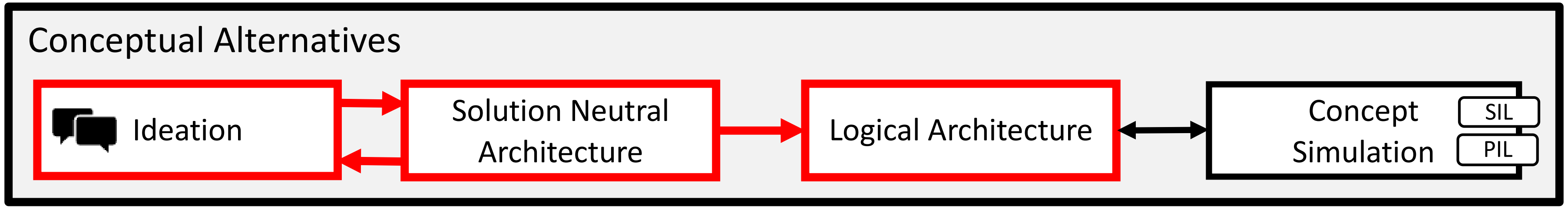
SOME THOUGHTS

- Logical Architecture answers:
  - ***"How** the system will work to fulfill expectations"*
- In the logical architecture we unfold the system in functional logical aggregators = components
- We must decide how to arrange the functions and the components and trade
- Maps the internal functions of the system.

Atividades para a próxima aula

- Fazer a etapa da formalização do modelo funcional
- Apresentar o modelo da arquitetura funcional:
  - Características mínimas: desdobrar em 3 subcomponentes, das funções de fronteira quebrar/juntar em no mínimo 10 subfunções, mostrar análise de coesão-acoplamento dos subcomponentes, fazer a máquina de estado de cada subcomponente (min 3), fazer o diagrama de interfaces internas, escrever 10 requisitos (8 funcionais e 2 não funcionais) desdobrados dos requisitos da intervenção sistêmica.

ROP



RTLI

[EXTRA] O PODER DO REUSO ☺

ACELERANDO AS COISAS

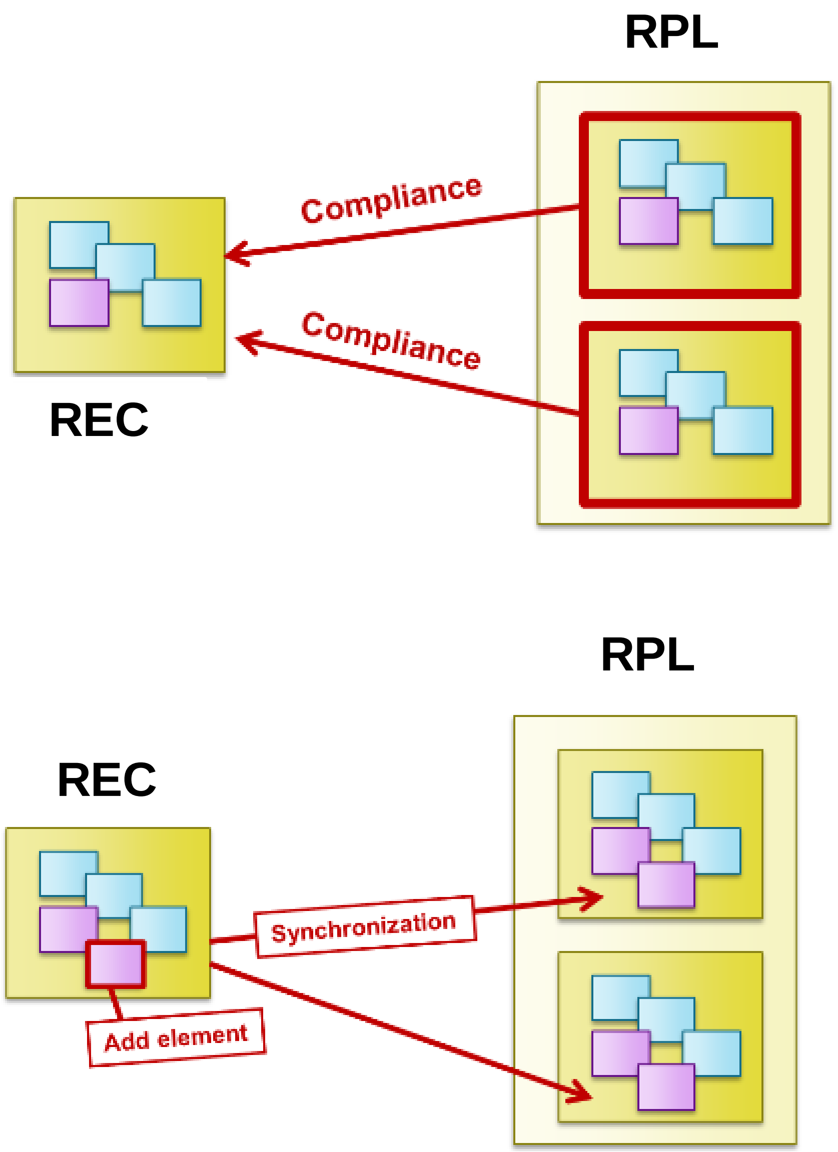
- **Uma grande vantagem** do “MB” é a capacidade de **reuso dos modelos**.
- Se todo projeto tiver que construir todo o modelo todas as vezes, não será diferente de fazer tudo baseado em documento, **na verdade será até mais demorado**.
  - Por isso temos que se beneficiar dos mecanismos de reuso das ferramentas.
- No Capella temos “dois mecanismos”:
  1. Coleções e Replicas
  2. Bibliotecas

Coleção de Elementos Replicáveis (REC) e Replicas (RPL)

Replicable Elements Collection (REC) e Replicas (RPL)

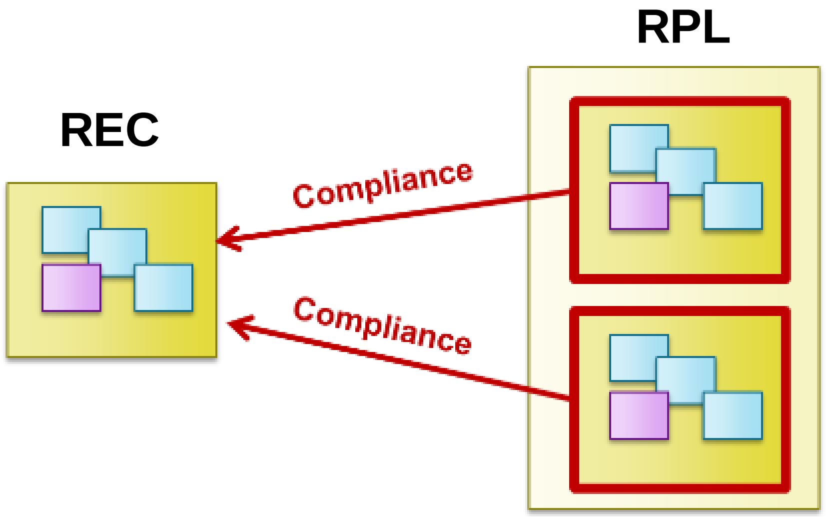
Escrito por Mateus S. Venturini

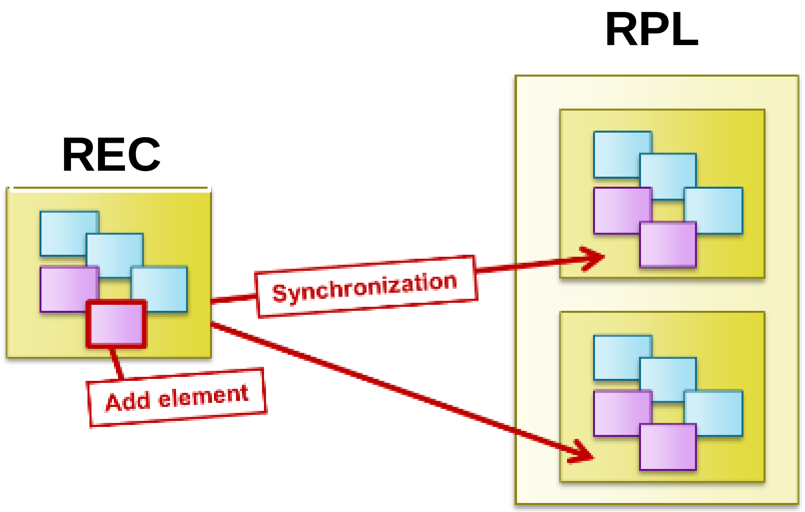
Definição



- Uma **Coleção de Elementos Replicáveis (REC)** é um conjunto de elementos de modelo, identificados como sendo um **padrão (um modelo no sentido comum do termo)** para a **construção de Réplicas (RPLs)** que mantêm conformidade com ele.

Definição



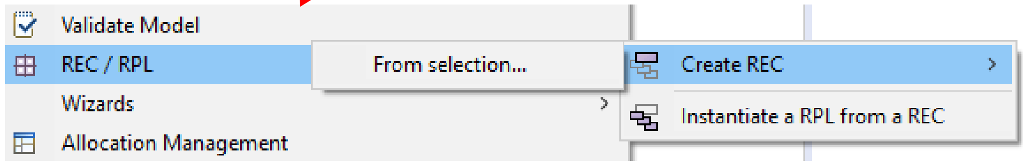


- A REC can be viewed as a “contract” to which all its RPLs must comply. REC can embed RPLs of other RECs.
- REC and RPL are located in Catalogs. Technically, REC and RPL are technical objects pointing towards the list of the elements they embed.
- Capella provides tooling to manage the creation of REC and their instantiation, as well as update mechanisms (from REC to RPL and from RPL to REC) and validation rules.
- Different kinds of conformance are possible between a RPL and its REC. Capella defines three default kinds of conformance, but end-user can define their own ones.
  - Blackbox: No modification is allowed on the Replica.
  - Constrained Reuse: Internal elements can be added inside a RPL, but constraints and Interfaces (Function and Component Ports for example) defined in REC cannot be modified.
  - Inheritance: Any element can be added in the RPL, including new Interfaces.

cRIANDO UMA REC

2. From a diagram, select a consistent set of elements (here, a Component and the Functions it is performing).

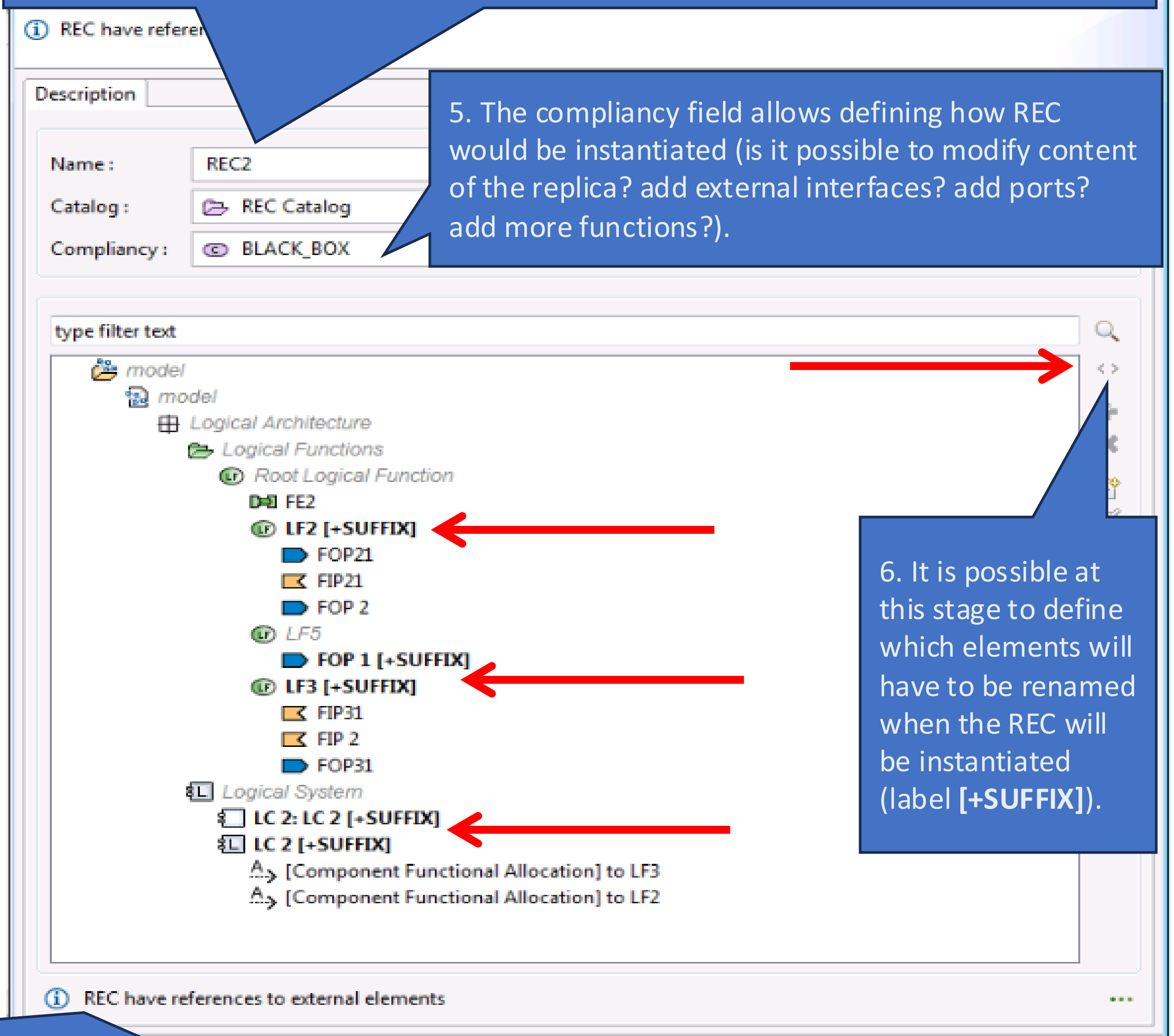
1. From the contextual menu, select "REC/RPL->Create REC->From selection..."



3. The scope (content) of the REC is displayed. This dialog helps modifying this scope (for example adding or removing elements). By default, the tool applies a set of business rules to include elements (for example, allocations between Component and Functions, children of an element, Etc.). Note here that despite Functional Exchange "fe23" is carrying Exchange Items, these Exchange Items are not included by default in the REC. In most of the cases, they shoud not be, as references are kept.

4. The REC creation dialog appears. A name shall be given to the REC. The Catalog field allows to select in which catalog this REC should be created. When working with Libraries, the Catalog is most likely located in a Library. In a library, an additional action "With whole library content..." is shown in the REC creation menu. If that action is chosen, the new REC will be initialized with the entire contents of the library.

5. The compliancy field allows defining how REC would be instantiated (is it possible to modify content of the replica? add external interfaces? add ports? add more functions?).

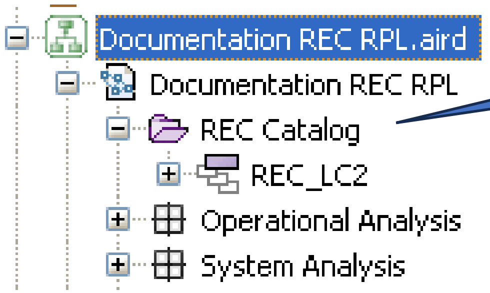


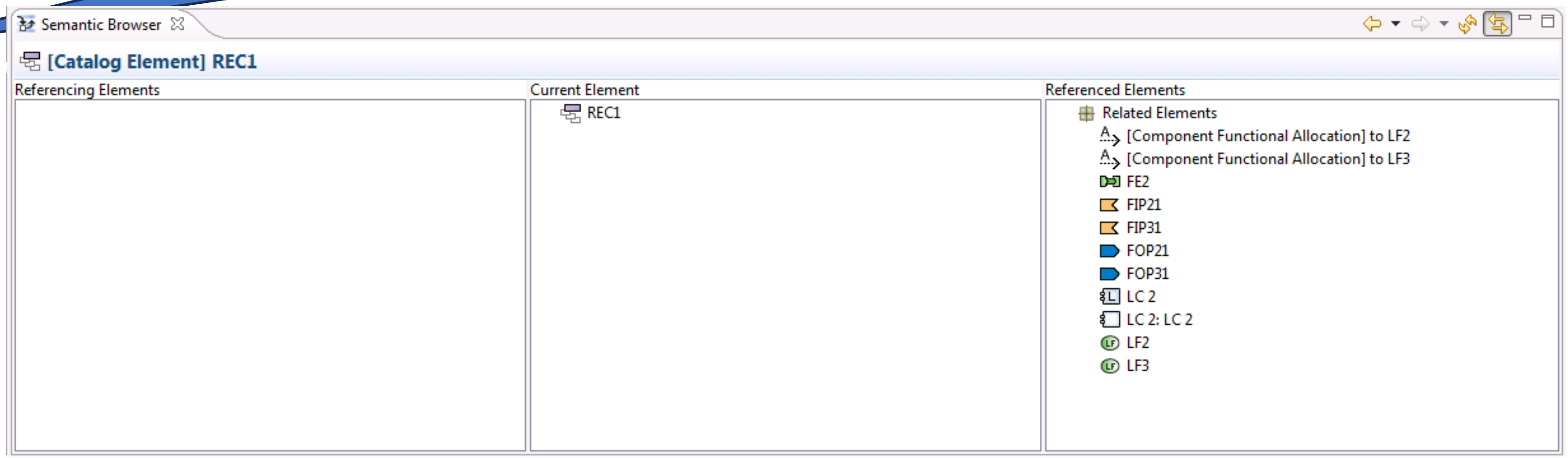
6. It is possible at this stage to define which elements will have to be renamed when the REC will be instantiated (label **[+SUFFIX]**).

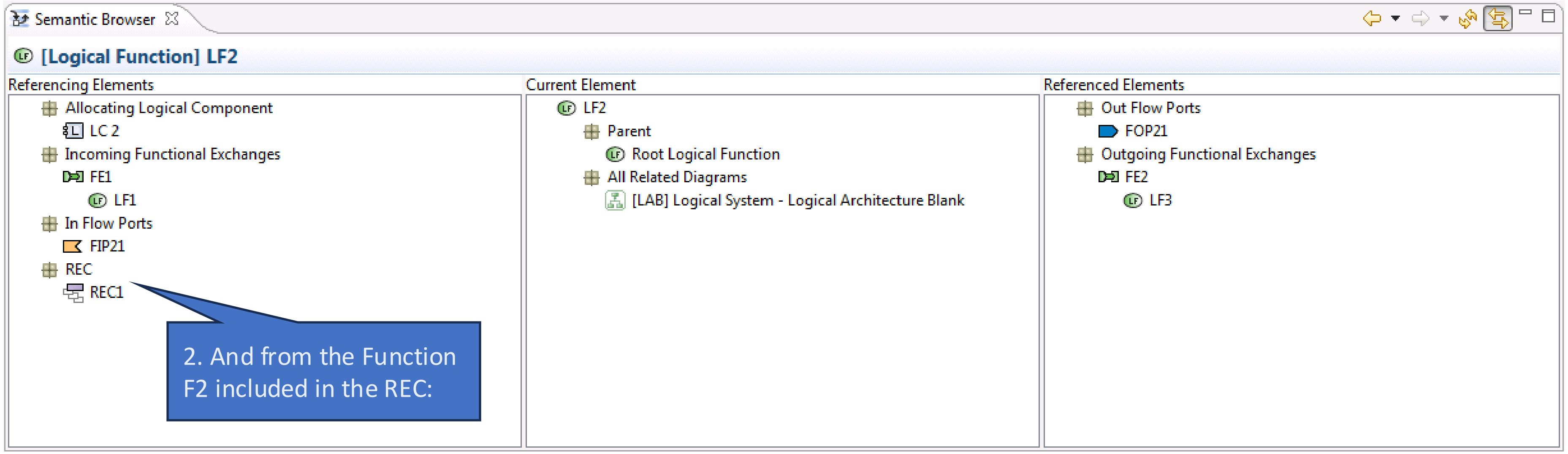
7. Notice the message at the bottom of the dialog, selected elements are linked to some elements which are not included in the REC (many exchange items, visible by clicking on the browse button on the right). When the REC will be instanciated, elements of the newly RPL will be linked to these exchange items too.

Criando...

1. Close the dialog and check the result.
In the Project Explorer, the newly created REC appears. The Semantic Browser also show REC-related information:



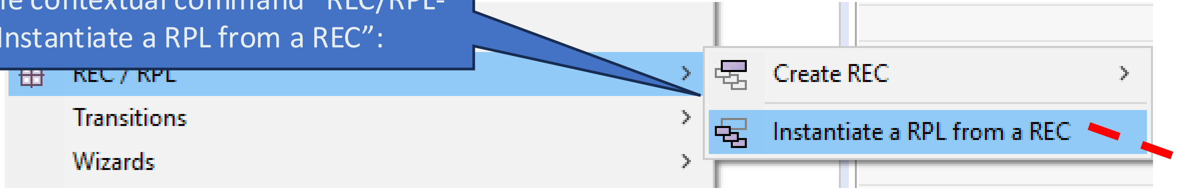




2. And from the Function F2 included in the REC:

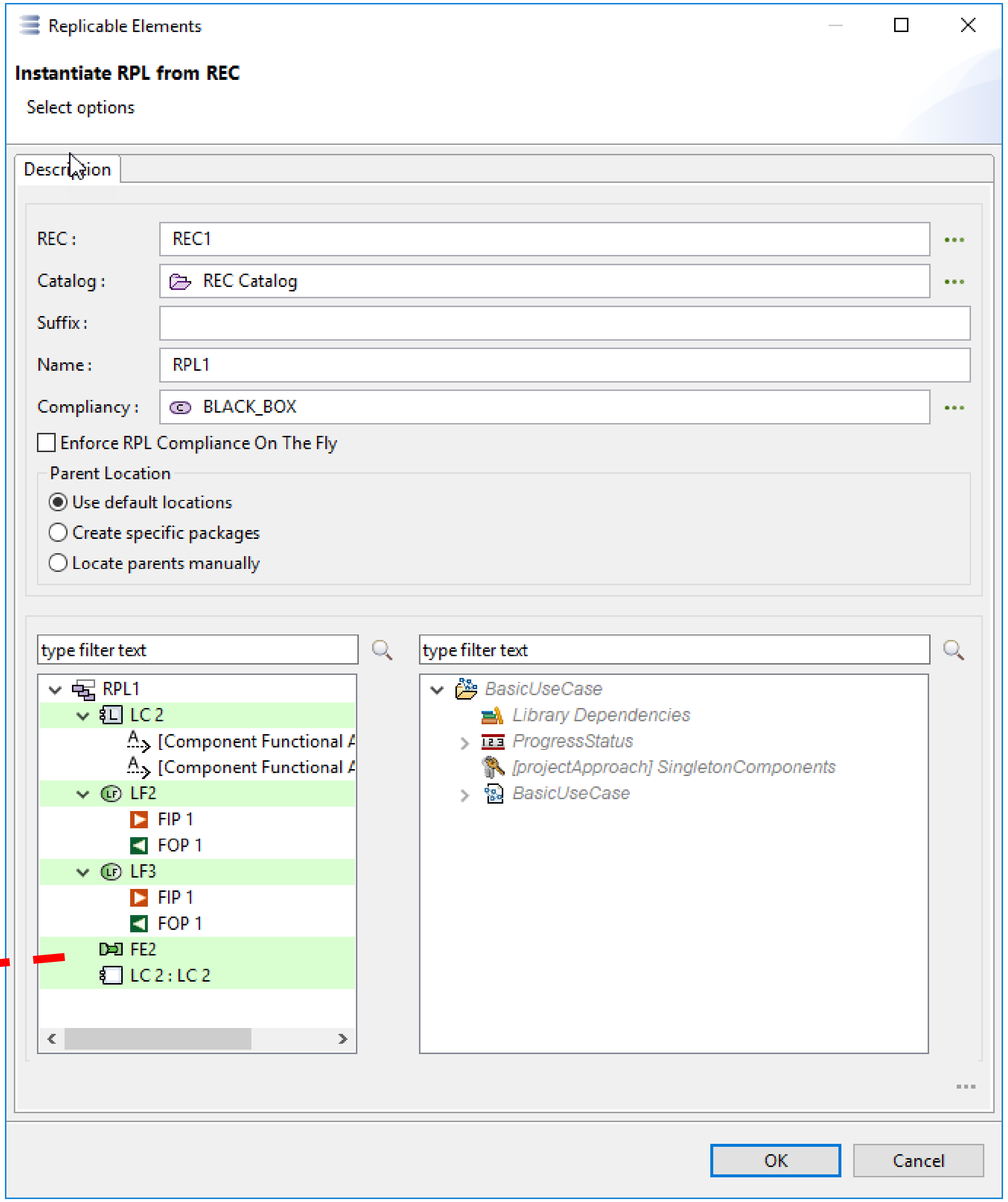
Usando uma rec EM UMA RPL

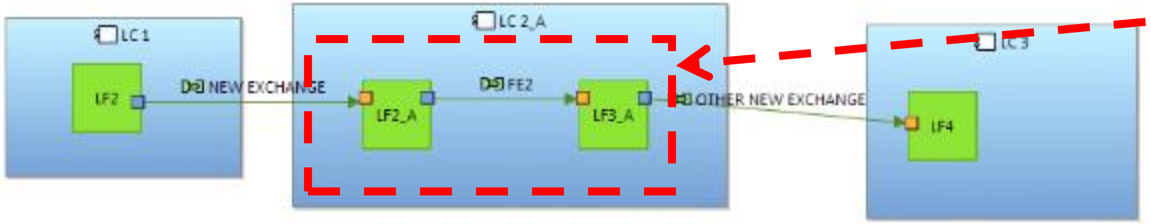
1. From anywhere in the model, use the contextual command “REC/RPL->Instantiate a RPL from a REC”:



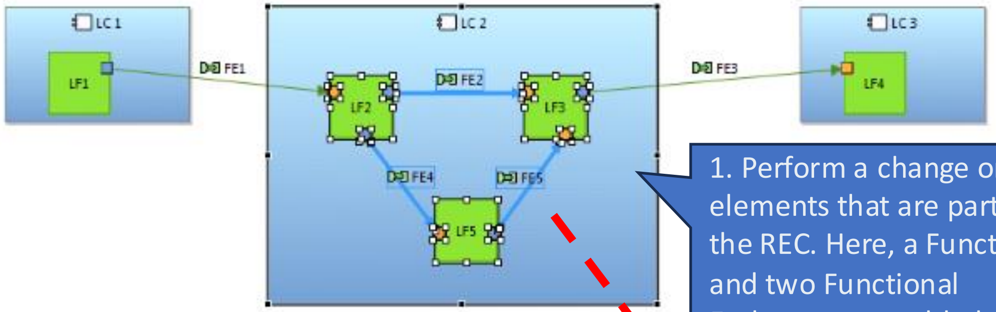
2. This dialog allows:
- The selection of the REC to instantiate (click on “Browse”)
- The definition of a target container (Catalog) for the RPL going to be created.
- The definition of a suffix for each element of the REC that was marked as having to be renamed.
- The compliancy field allows defining how RPL can be modified according to its REC (is it possible to modify content of the replica? add external interfaces? add ports? add more functions?) See the RPL Validation part for further description of any kind of compliancy **(This feature is not fully available yet)**
- To enable live compliancy validation for this RPL select “Enforce RPL Compliance on the fly”.

All RPL elements corresponding to a REC element with the suffix tag [+SUFFIX] will have the RPL suffix.
The parent locator options exist to specify where the RPL elements will be located:
- Use default locations: Elements will be located in standard containers in the model
- Create specific packages: For each element type, a RPL specific package will be created. Elements of the corresponding type will be stored in that package. Some elements, e.g. Parts do not get a specific package and are located just as if the default locations option would be selected.
- Locate parents manually: A location has to be found manually for the root elements of the RPL. The elements for which a location still has to be found are marked in Orange. The definition of a new location is performed using drag and drop between the two trees:

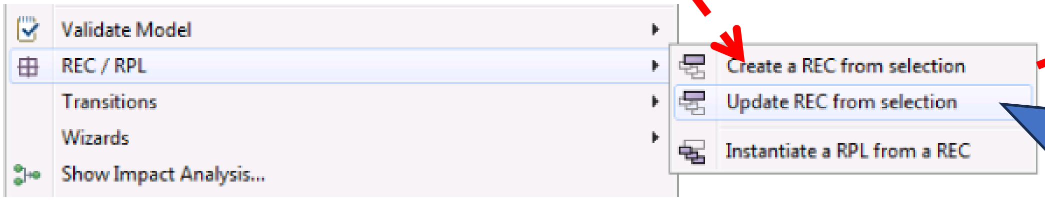




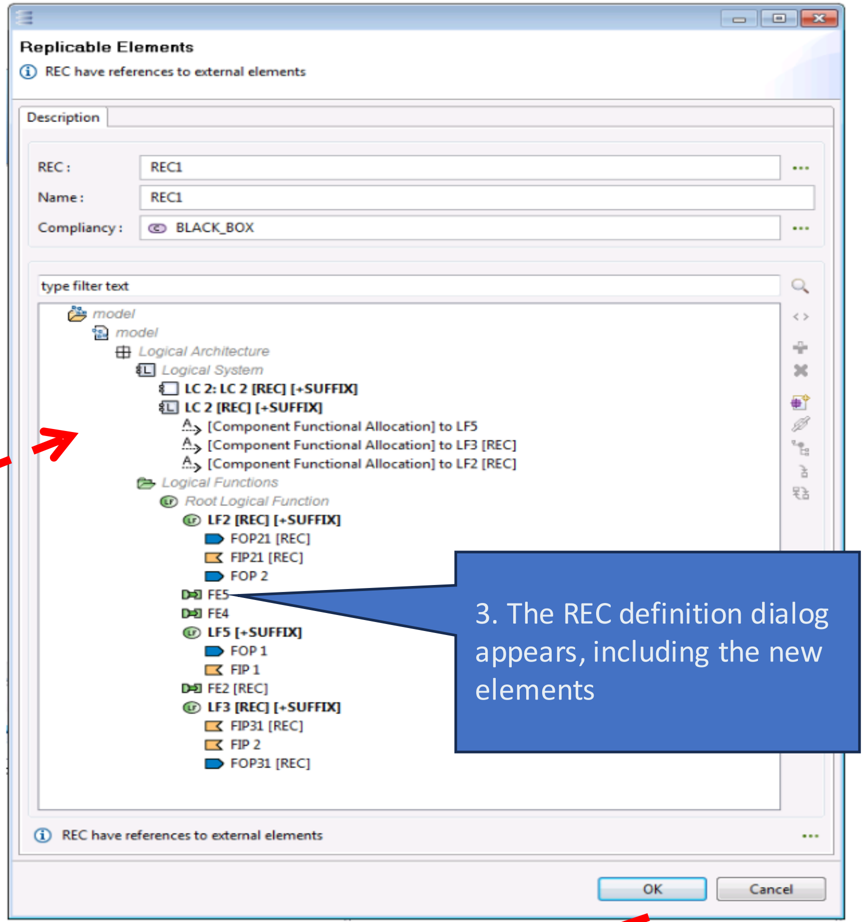
UPDATE REC from the RPL



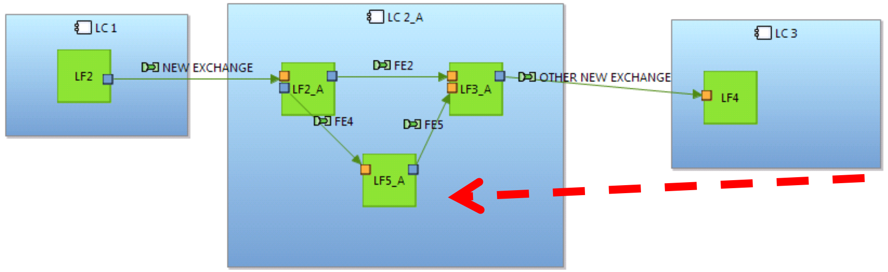
1. Perform a change on elements that are part of the REC. Here, a Function and two Functional Exchanges are added.

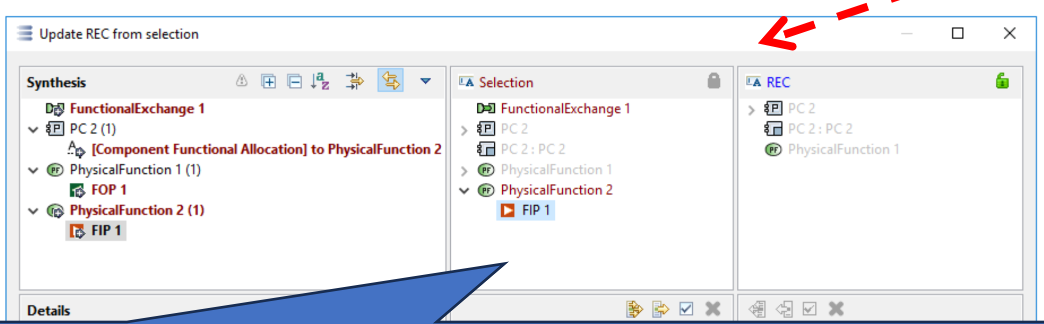


2. Select at least one original element of the REC and from the contextual menu, choose 'Update REC from selection"



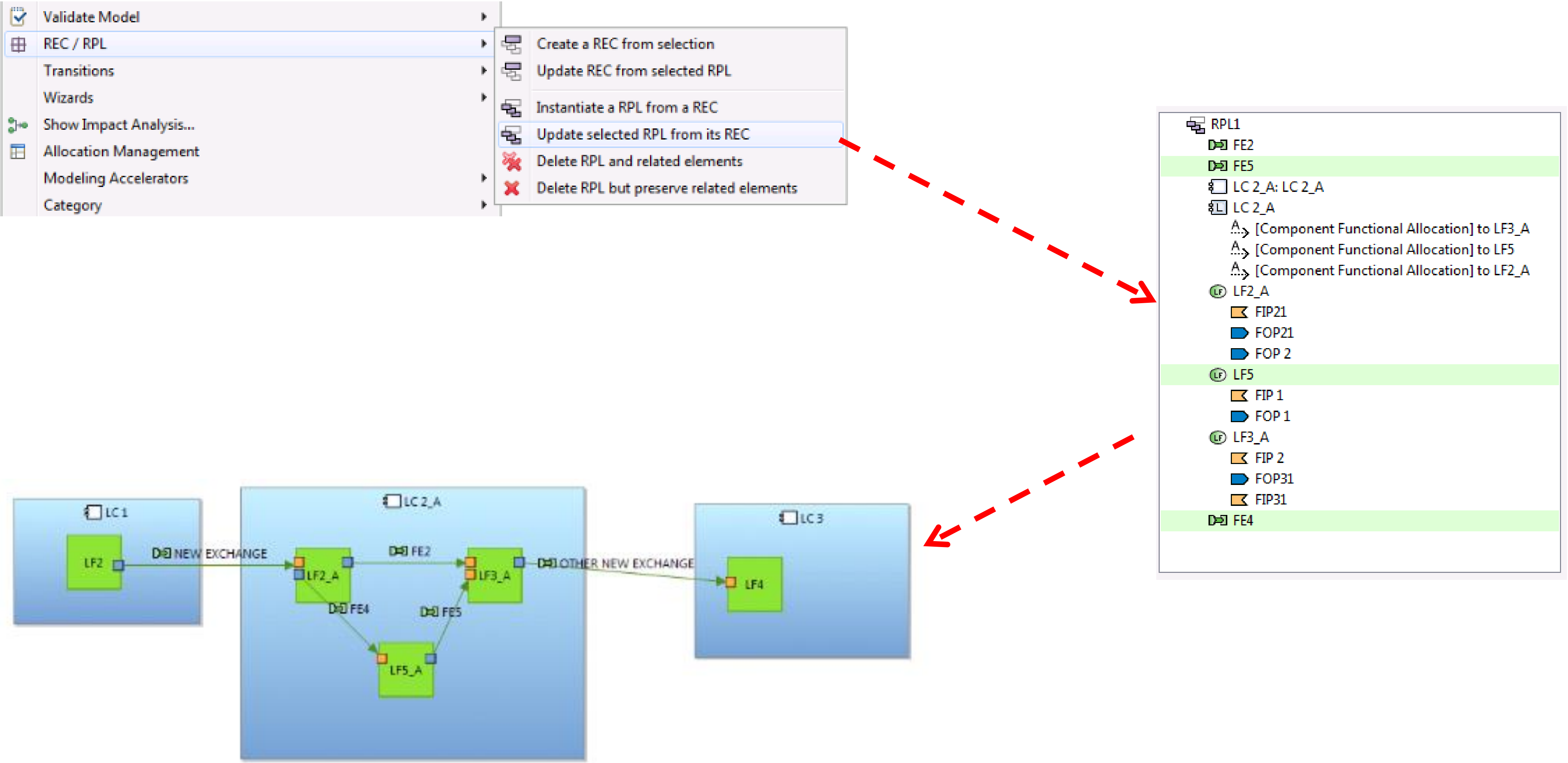
3. The REC definition dialog appears, including the new elements





4. Click on OK. This will open the DiffMerge view. Press 'Apply' to update the rec to include all additional changes to the REC (additional information about this dialog is available in the Model DiffMerge section) .

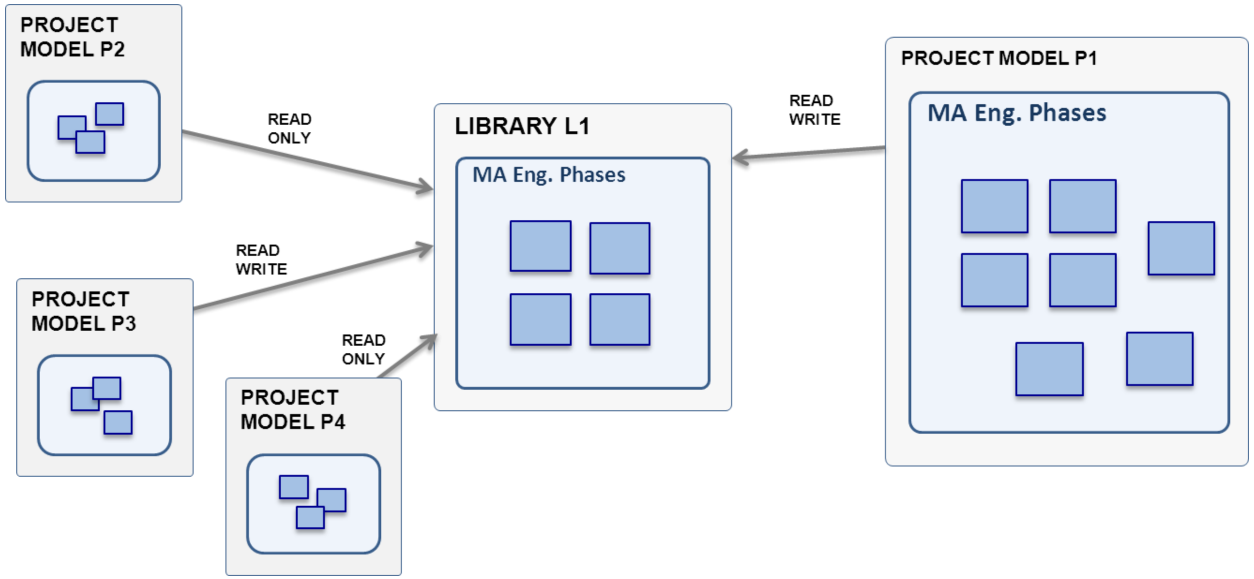
Update RPL from the REC



Bibliotecas

definição

- Biblioteca é um modelo Capella destinado a ser compartilhado entre vários projetos.
- Um projeto pode fazer referência a uma biblioteca com READ ou READ/WRITE. Neste último caso, isso significa que o conteúdo da Biblioteca pode ser modificado a partir do próprio Projeto, sem ter que abrir especificamente a Biblioteca.
- Uma biblioteca pode ter referências a outras bibliotecas, mas uma biblioteca não pode ter uma referência a um projeto.

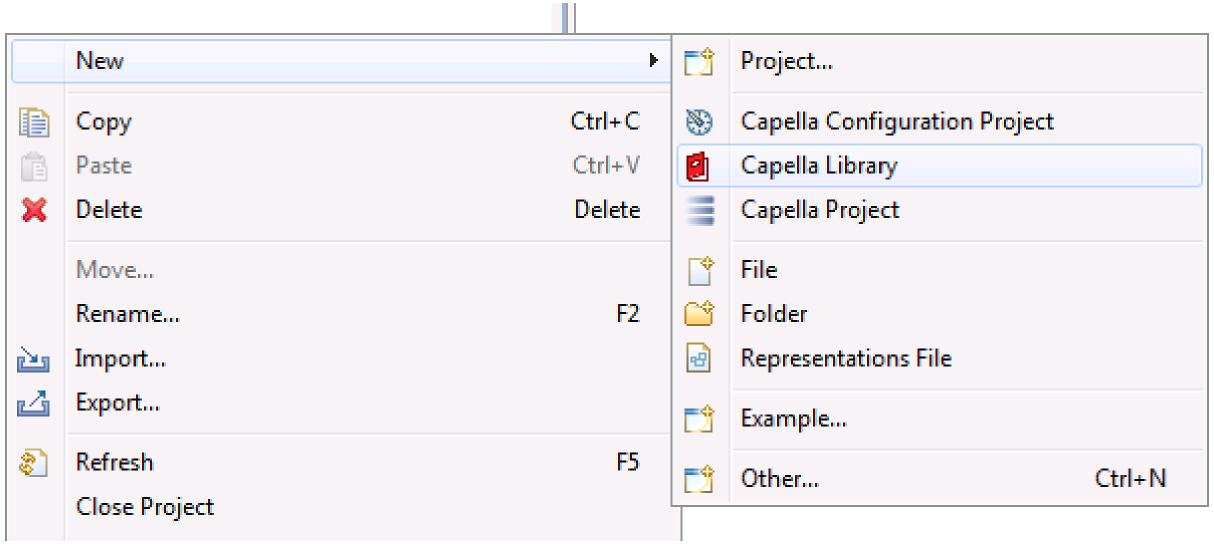


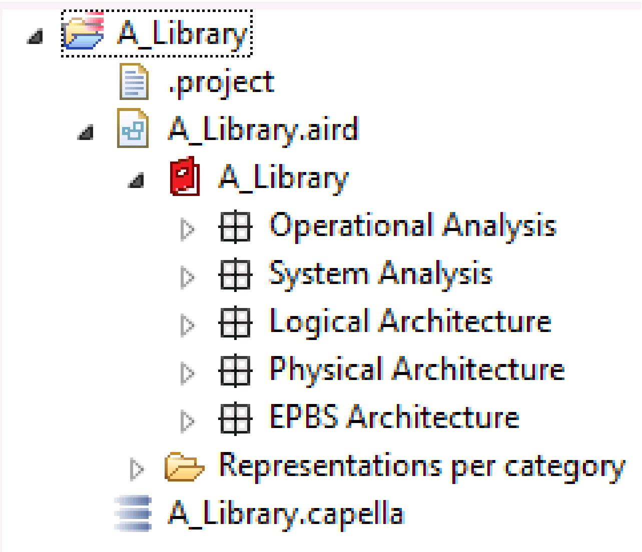
A que se destinam as Bibliotecas?

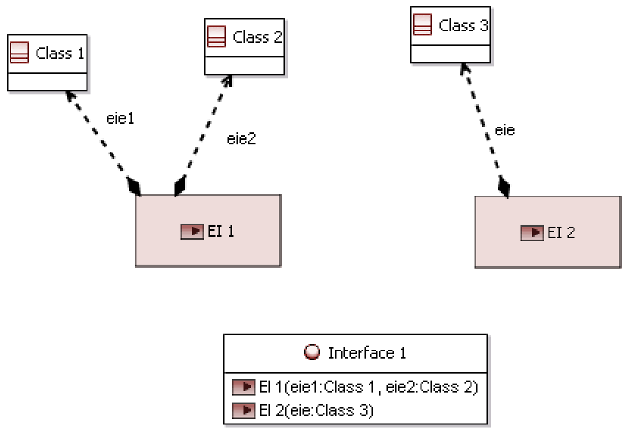
- Permitir a reutilização de elementos de modelo em modelos diferentes (por exemplo, vários projetos em um domínio geralmente precisam compartilhar o mesmo modelo de dados).
- Melhorar a organização (evitar duplicação e referências entre modelos)
- Catálogos de elementos replicáveis
- As bibliotecas se beneficiam das mesmas ferramentas que os modelos.
    - Edição do conteúdo da biblioteca através de diagramas e editores
    - Navegador semântico
    - Regras de validação e correções rápidas

Criação de bibliotecas

- As bibliotecas são criadas da mesma forma que os projetos Capella padrão.
- Do Project Explorer, criar uma nova biblioteca usando o menu contextual

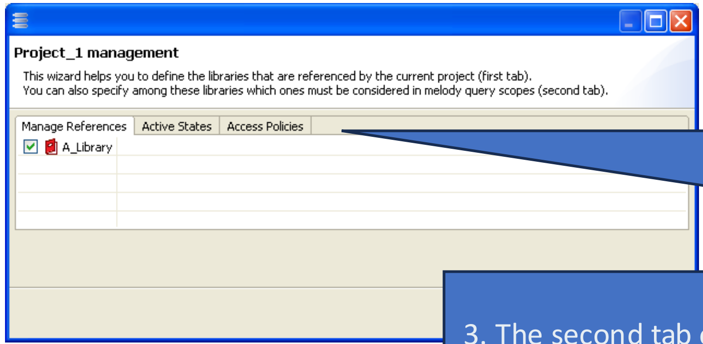






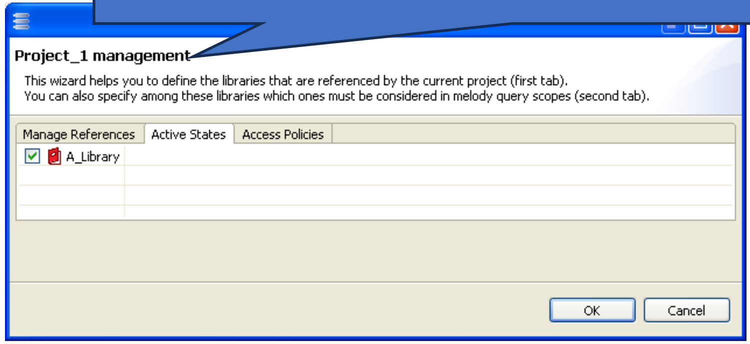
Referenciando uma biblioteca

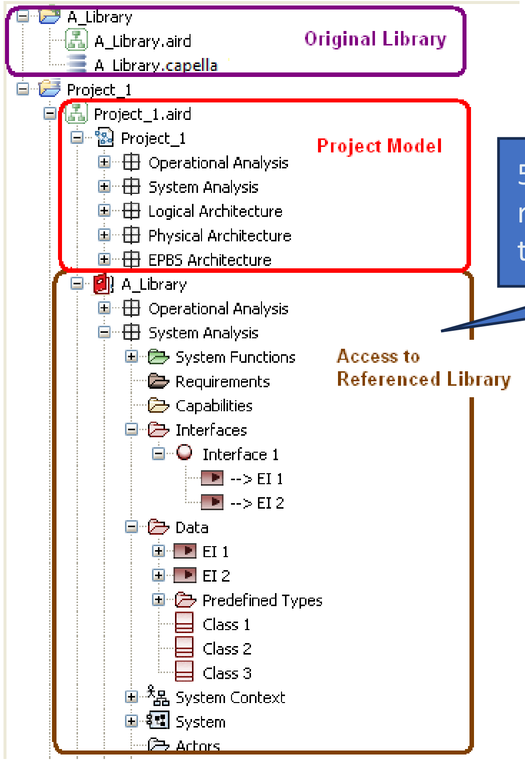
1. Select the “Libraries | Manage References” item in the contextual menu on the “aird” file of a standard Project Model.



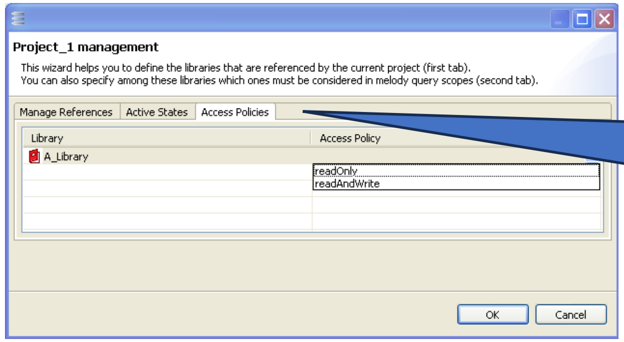
2. The first tab of this dialog displays all the accessible Libraries in the current workspace (A Library in a closed Eclipse Project will not be proposed).

3. The second tab of the dialog displays which Library is currently active. When a Library is not active, queries in Editors for example will not display the content located in the Library.





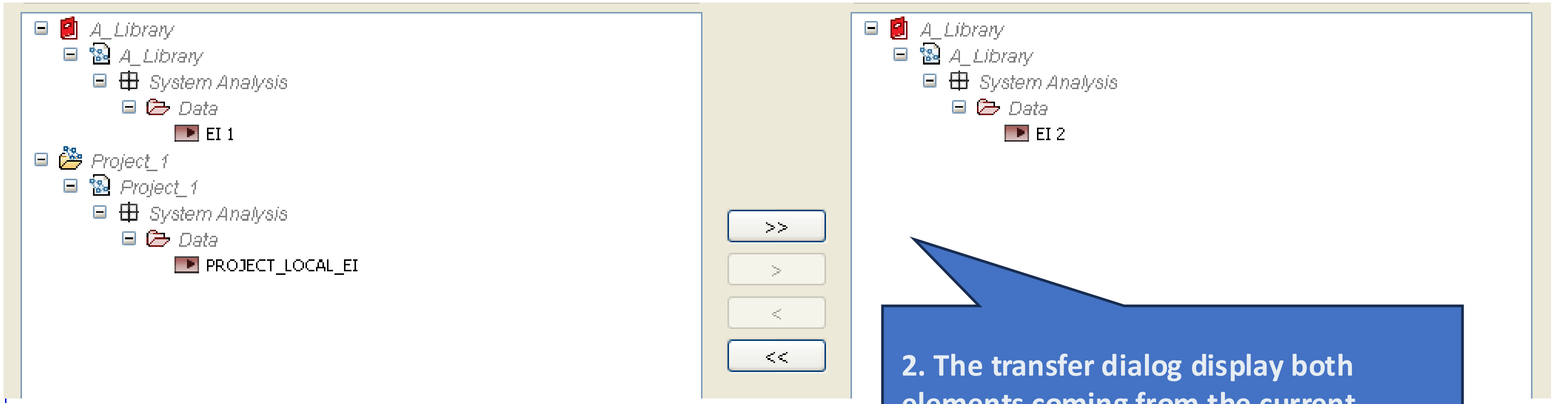
5. Once the Project Model is opened, the referenced Library can be seen directly from the Project itself.



4. The third tab allows specifying whether the content of a referenced Library can be directly modified from the Project itself. The default is “Read only”.

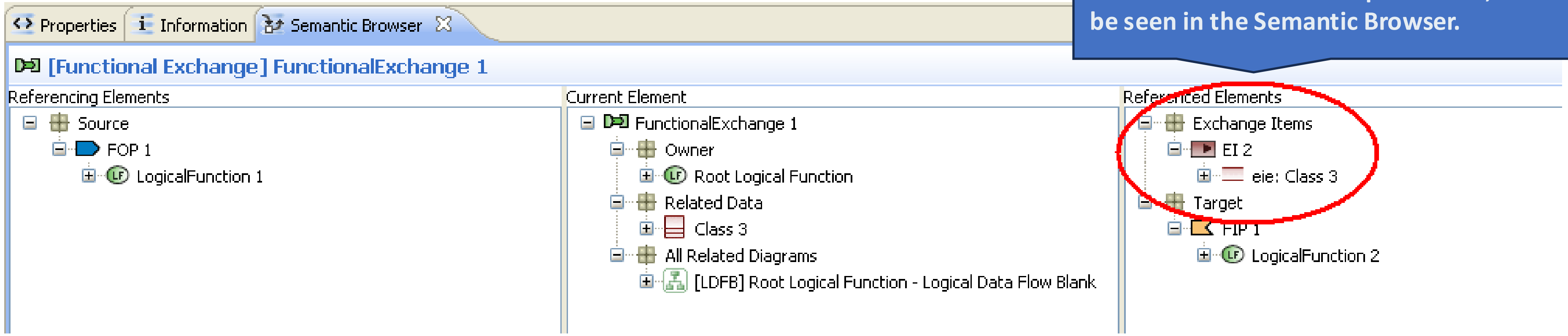
Usando: allocation of Exchange Items to a Functional Exchange

1. Example with the allocation of Exchange Items to a Functional Exchange

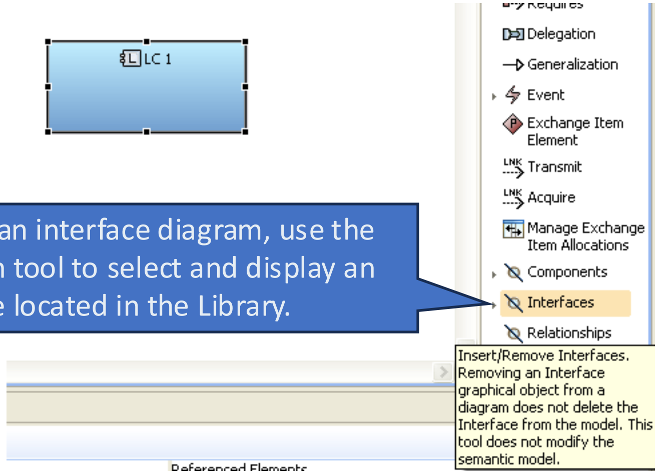


2. The transfer dialog display both elements coming from the current Project and elements from the Library.

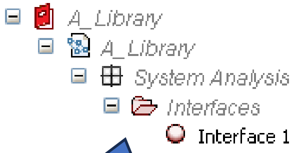
3. Once the allocation is performed, the result can be seen in the Semantic Browser.



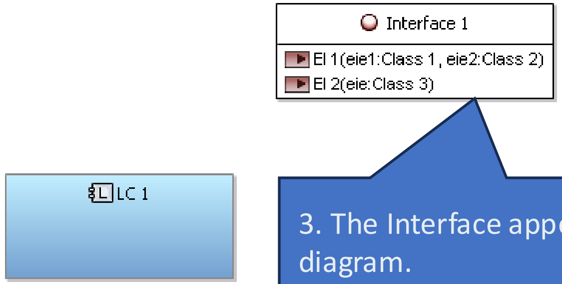
Usando: Components and Interfaces



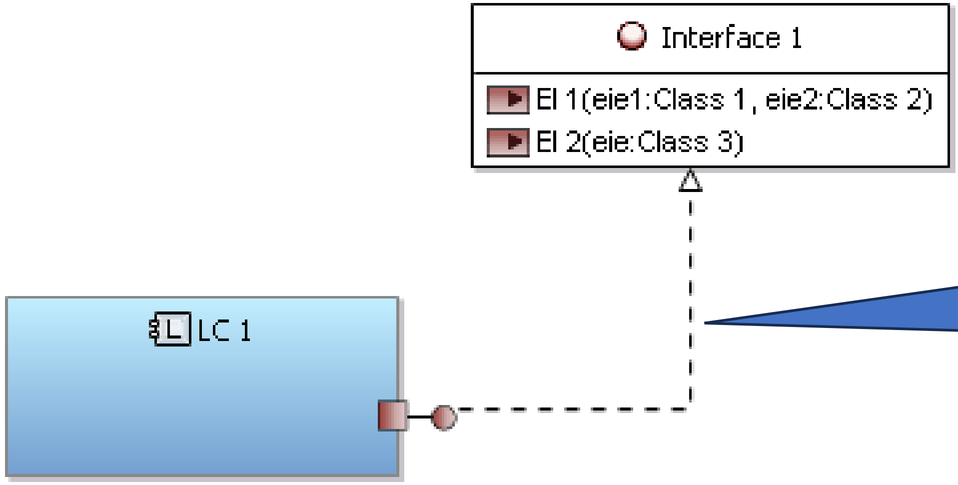
1. From an interface diagram, use the Insertion tool to select and display an Interface located in the Library.



2. The selection wizard proposes the Interface located in the Library.

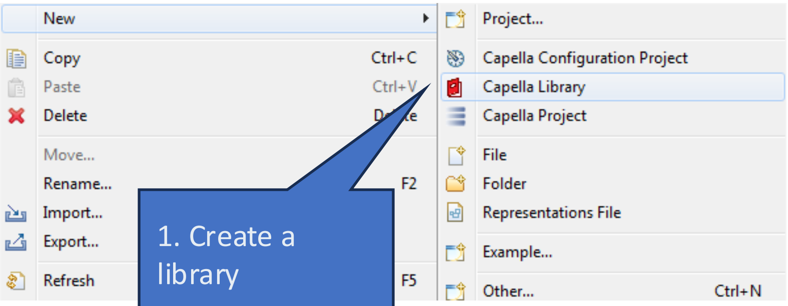


3. The Interface appears on the diagram.

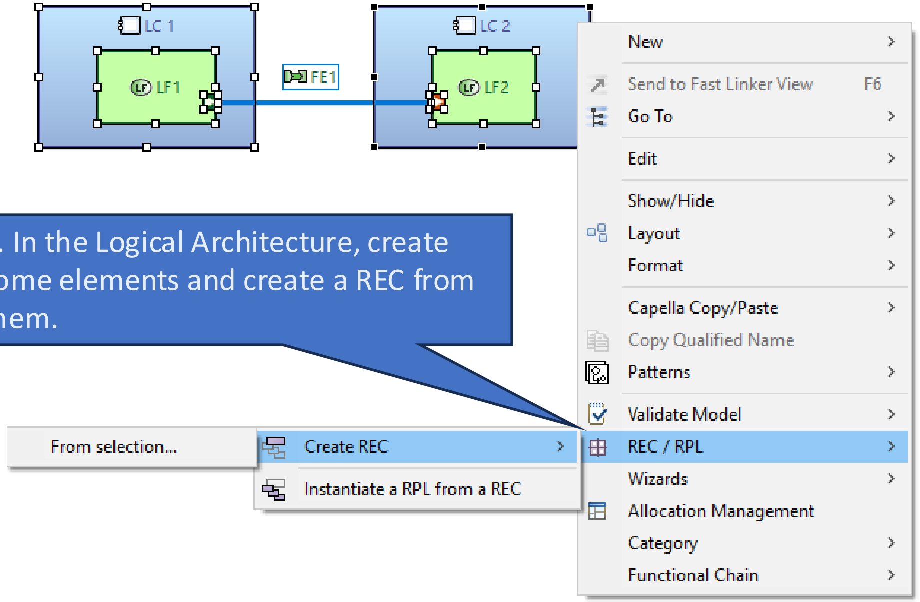


4. The Component in the Project Model can now “provide” the Interface located in the Library.

USANDO COM RPL/REC

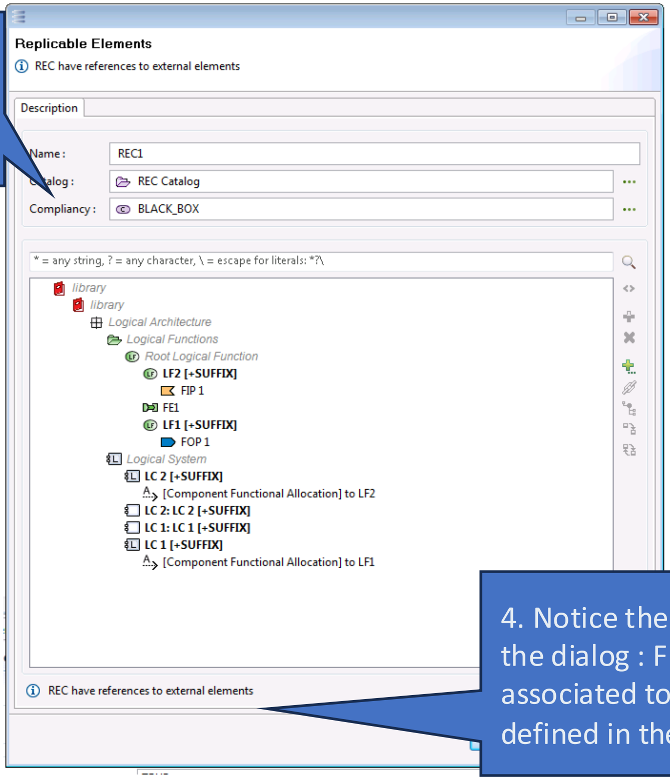


1. Create a library

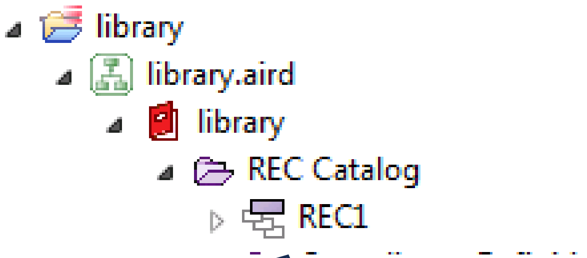


2. In the Logical Architecture, create some elements and create a REC from them.

3. The dialog displays the content of the REC.



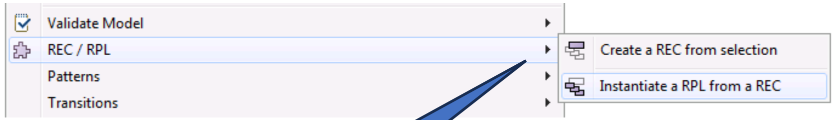
4. Notice the message at the end of the dialog : Functional Exchange is associated to some Exchange Items defined in the library:



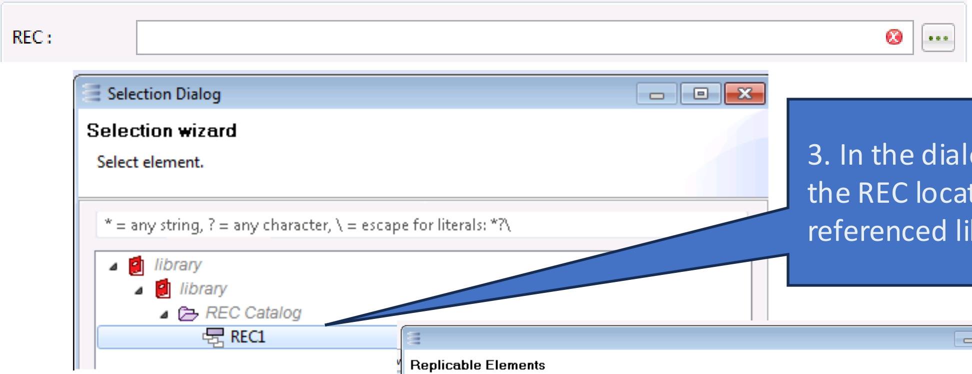
5. After creation, the REC is available In the library.

Ensure your library is saved.

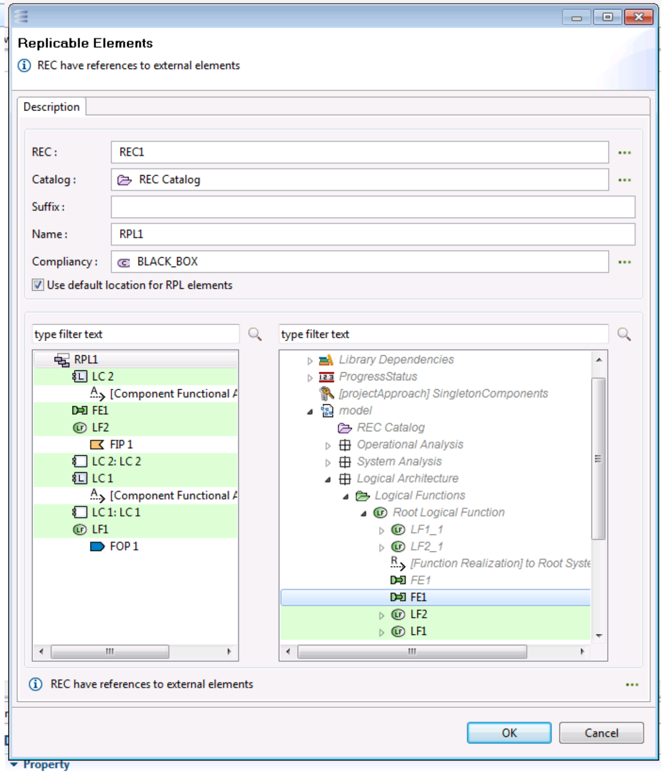
ADICIONANDO A RPL DA LIB NO PROJETO



1. In a Logical Architecture Blank diagram, instantiate a new RPL



3. In the dialog, select the REC located in the referenced library:



2. Display all elements of the RPL in the diagram

REFERÊNCIAS sobre REC->RPL / libs

- **[HOW TO] Replicate model elements in Capella (4’25”)**
- https://www.youtube.com/watch?v=h-ax61eVlxM

- **Webinar - Strategies and tools for model reuse with Capella (58’23”)**
- https://www.youtube.com/watch?v=I28EhAXe-i8

- **In-Flight Entertainment System (IFE) – Example**
- https://download.eclipse.org/capella/samples/1.3.1/InFlightEntertainmentSystem.zip

- **Capella Help – Replicable Elements**