The ISE&PPOOA MBSE Methodology. A practical approach to apply model-based systems engineering.

José L. Fernández
MBSE methodologist
Madrid (Spain)

https://www.linkedin.com/in/jose-luis-fernandez-sanchez-4838425

jose.fernandez@incose.org

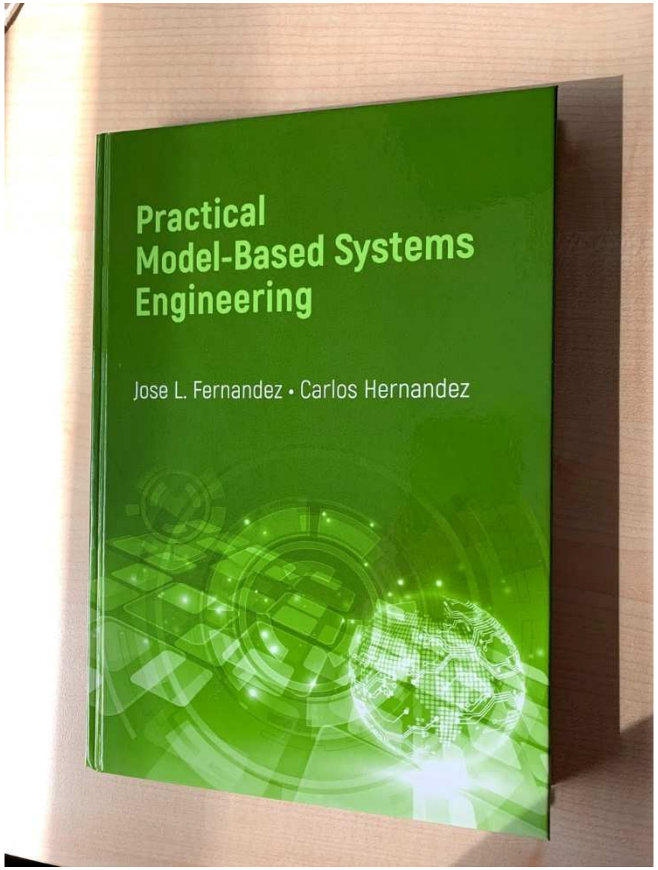
Practical Model-Based Systems Engineering

Authors: Jose L. Fernandez and Carlos Hernández

Publisher: Artech House, Norwood, MA. July 31,2019

ISBN-13:978-1-63081-579-0

ISE&PPOOA wiki: https://www.omgwiki.org/MBSE/doku.php?id=mbse:ppooa



Contents

- What do we mean by a MBSE methodology?
- The ISE&PPOOA methodology
  - A requirements driven methodology
  - The use of trees, flows and bridges
  - Conceptual model of ISE&PPOOA
- The ISE&PPOOA dimensions and process
- Software intensive subsystems
- Application example- Unmanned Aerial Vehicle

What do we mean by a MBSE methodology?

MBSE methodology definition

- A **Model-Based Systems Engineering Methodology** can be characterized as the collection of related processes, methods and tools used to support the discipline of systems engineering in a model-based or model-driven context[1]

[1]Stefan et al., "Survey of Model-Based Systems Engineering (MBSE) Methodologies". INCOSE-TD-2007-003-01)Version/Revision: B, 10 June 2008. INCOSE, Seattle (WA),2008.

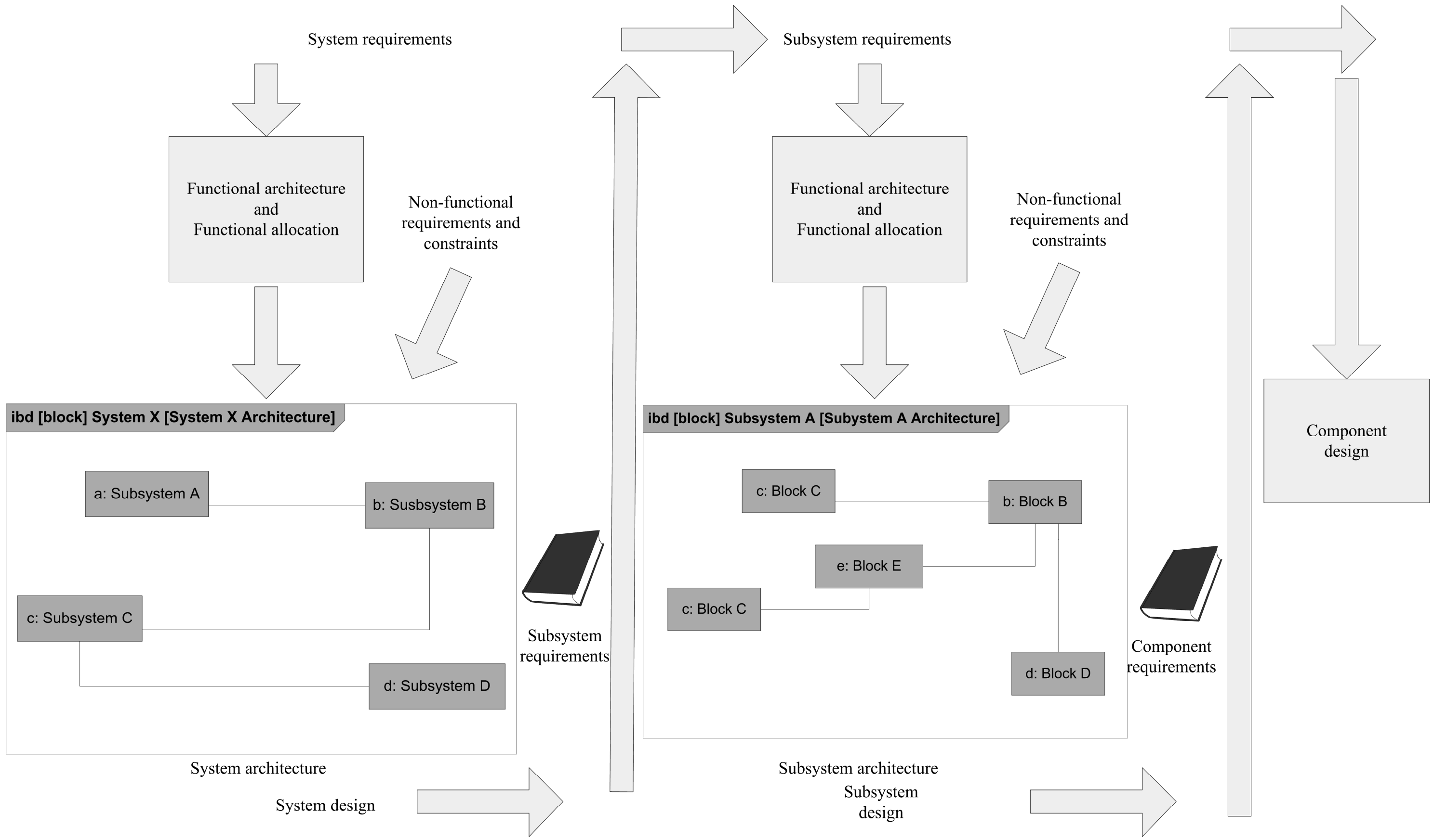
- **Methodology**= **processes** + **methods** + **tools**

Integrated Systems Engineering and Pipelines of Processes in Object Oriented Architectures (ISE&PPOOA) methodology

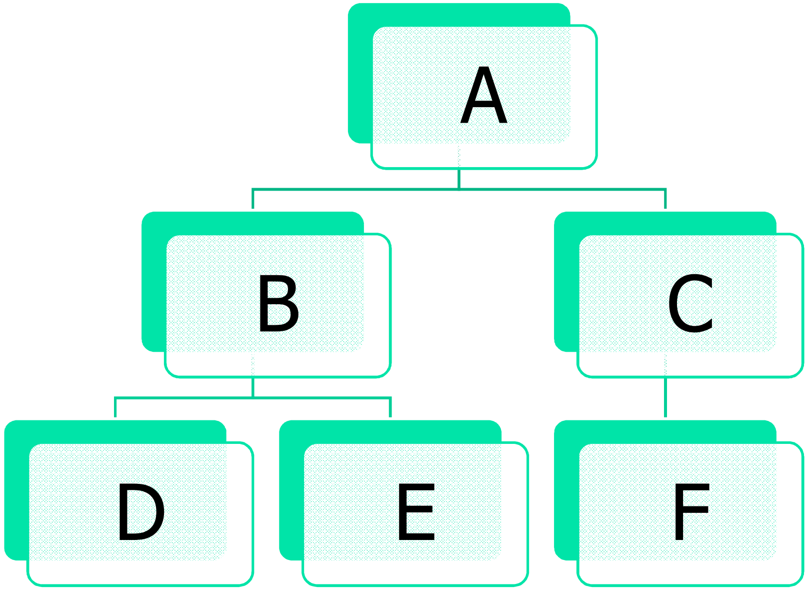
ISE&PPOOA

- **ISE&PPOOA** is a requirements-driven, model based systems engineering approach where the main outcomes are the functional and physical architectures of the product, system or service to be developed
- We are proposing a **way of thinking consistently** to solve an engineering problem, where identifying the functions and quality attributes of the product to be developed is a main issue to synthesize the solution

ISE&PPOOA is requirements driven



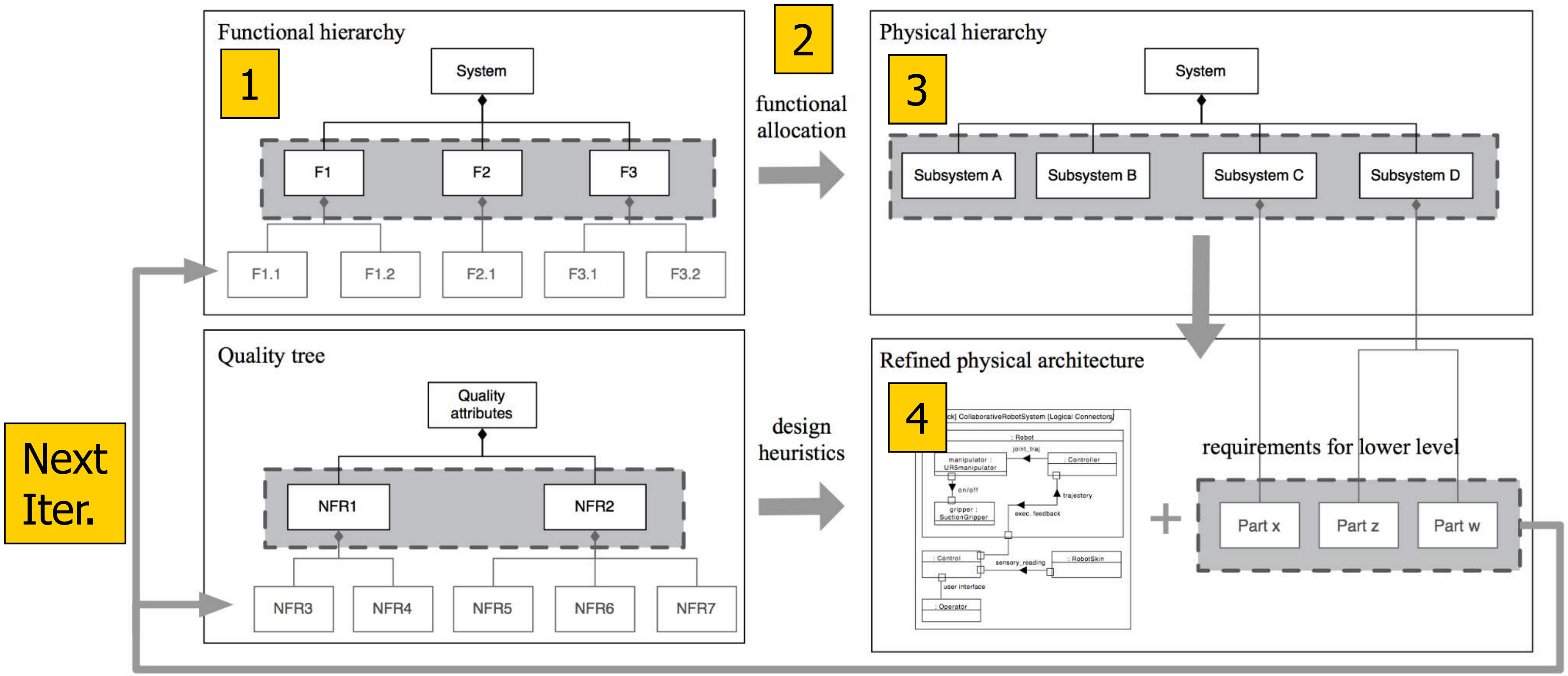
Trees, flows and bridges



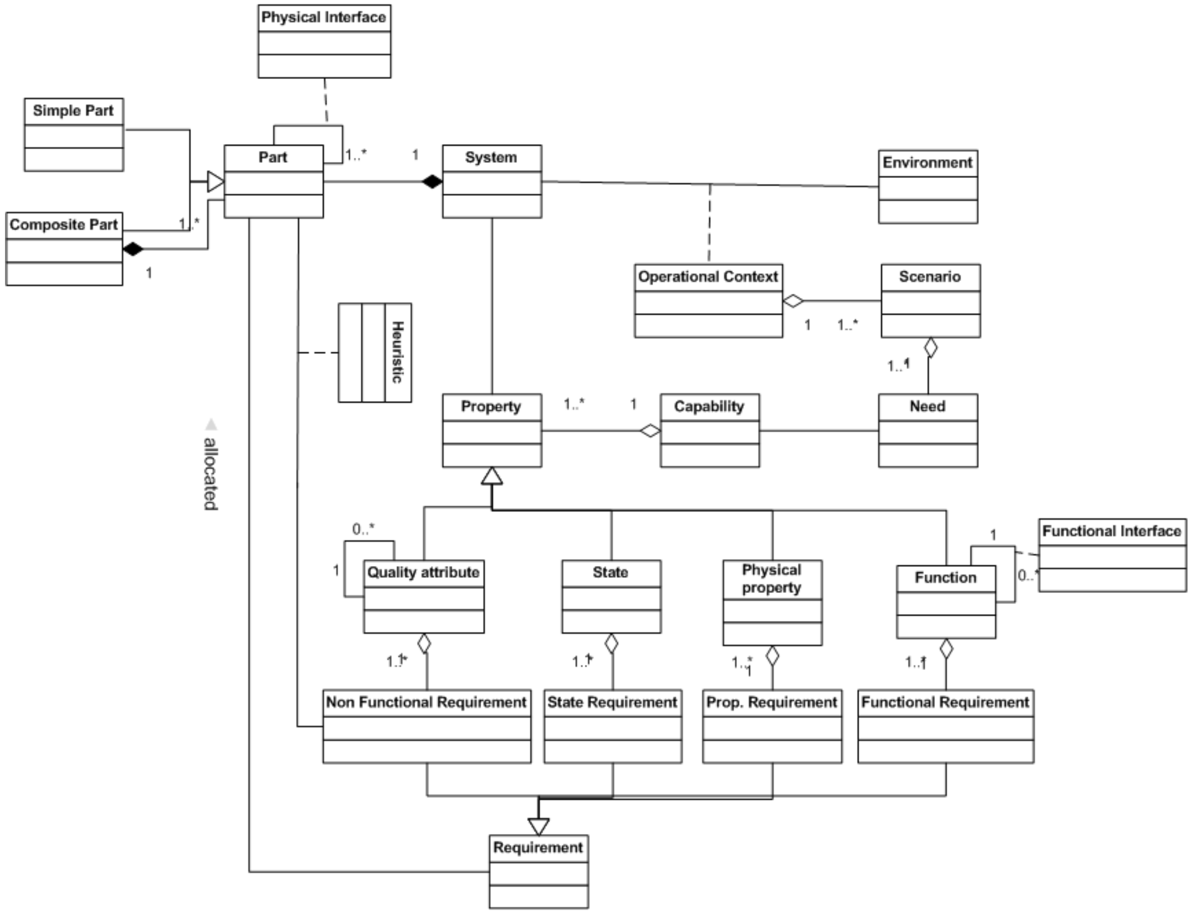
Trees, flows and bridges

- **Trees** are the diagrams used for **hierarchical** representations, for example: functional, quality models, physical and requirements diagrams.
- **Flows** are the diagrams used for representing **behavior** that is functional flows that describe system behavior represented here as SysML activity diagrams.
- **Bridges** are the way to **cross over the semantic gap** between two engineering areas. For example "**heuristics**" bridge nonfunctional requirements and the system refined architecture. The "**domain model**" bridges the system architecture and the software architecture

Functional, Physical and Quality Trees



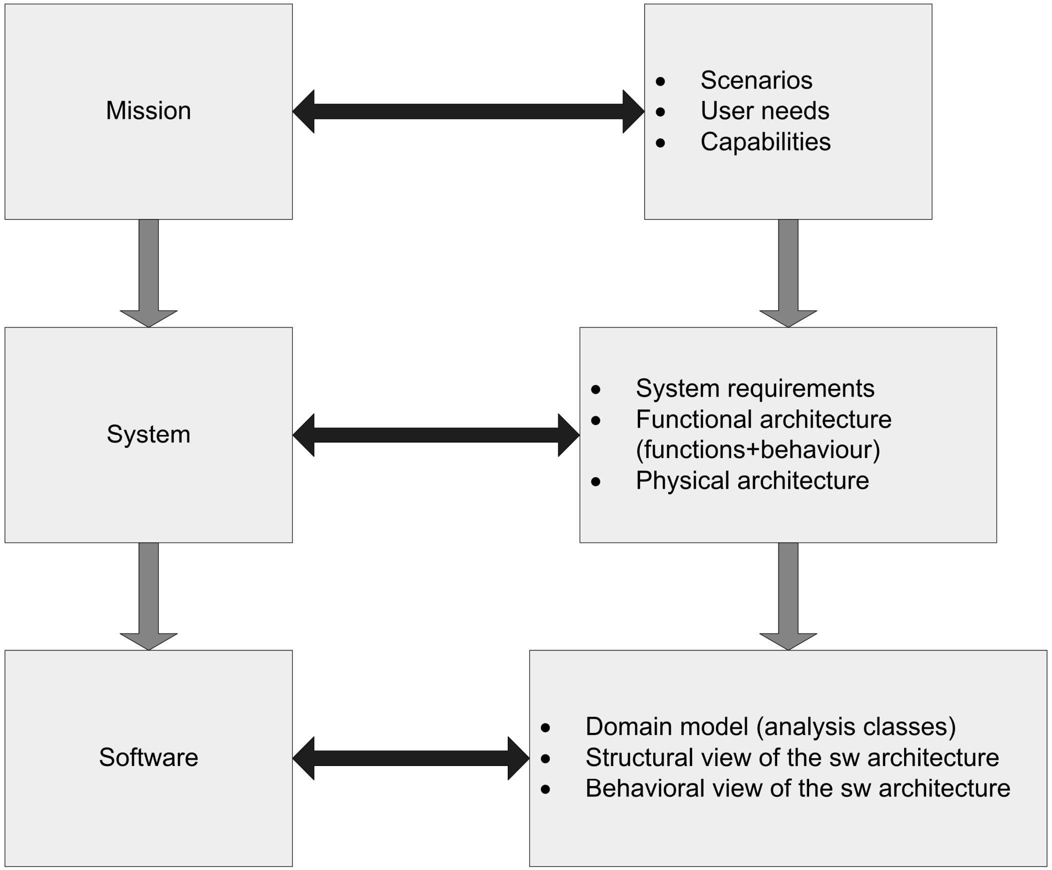
ISE&PPOA conceptual model



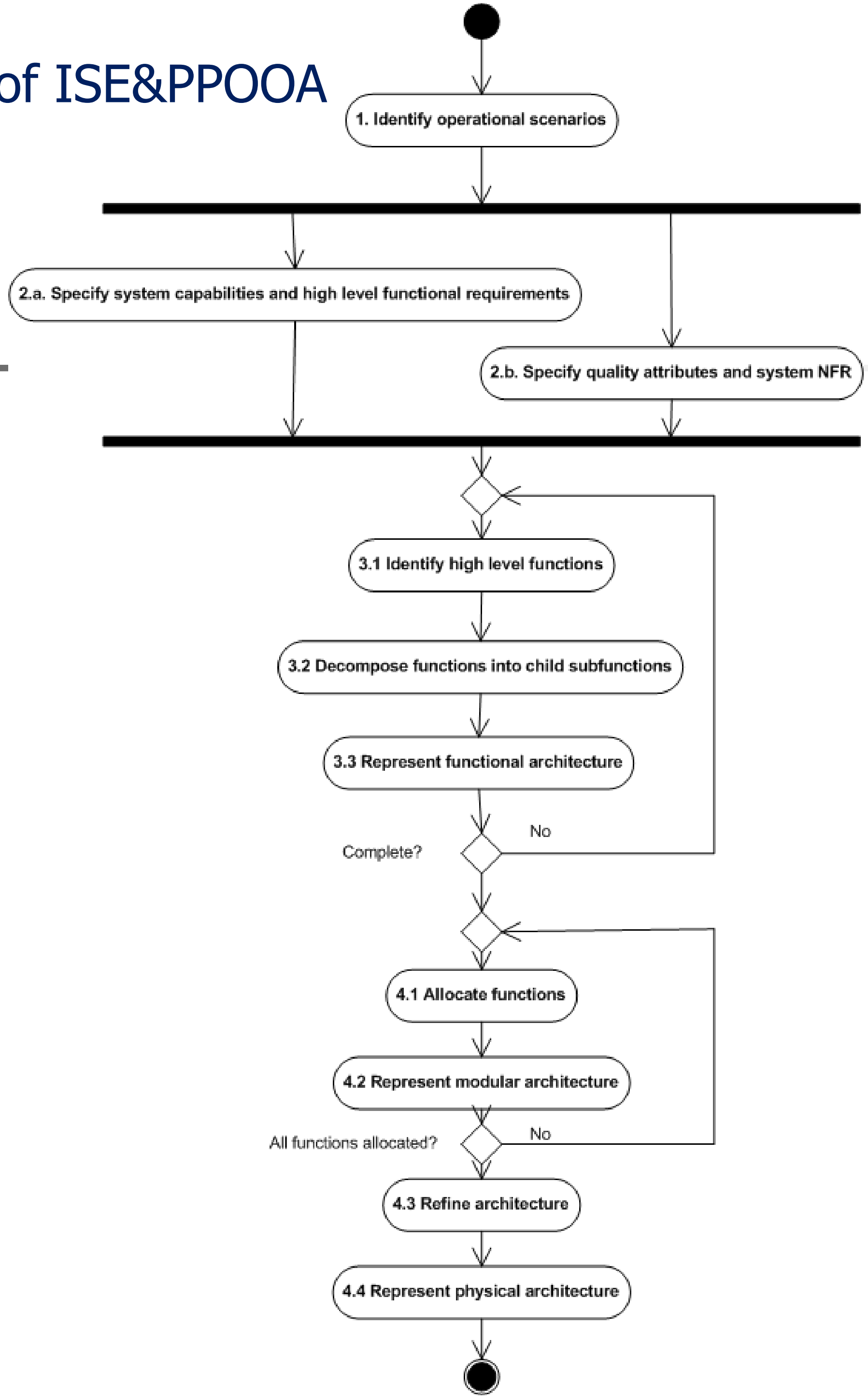
ISE&PPOOA dimensions and process

The three dimensions of ISE&PPOOA and the steps to apply the methodology iteratively to derive the low level requirements and create the architecture of the system

The three dimensions of ISE&PPOA



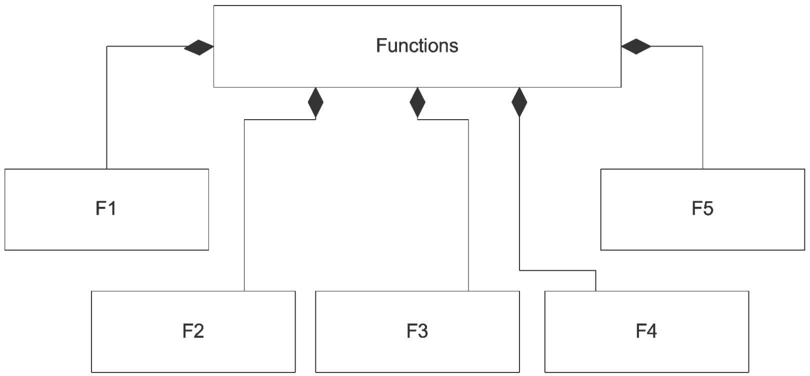
The ISE subprocess of ISE&PPOOA



Step 3.3. Represent the functional architecture of the system

- **Goal:** Represent the functional architecture identifying the **functional hierarchy, functional flows or behavior and functional interfaces**
- **Deliverable**: The deliverable is the functional architecture representing the functional hierarchy using a SysML block definition diagram. This diagram is complemented with **activity diagrams** for the main system functional flows to represent behavior. The **N^2 chart** is used as an interface description where the main functional interfaces are identified. A **textual description** of the system functions is provided as well
- **Profiles:** Systems engineers with the collaboration of customers and users

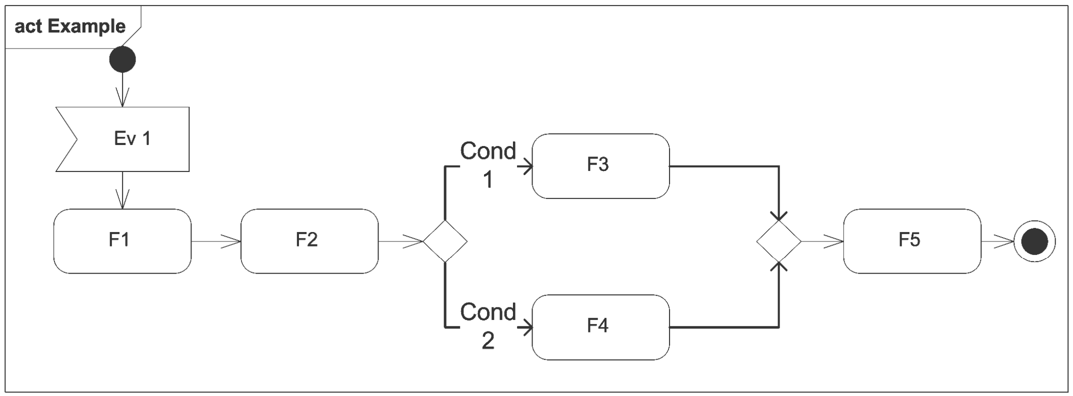
System functional architecture



Functional hierarchy

item1				item 6	
F1	item 2				
	F2	item 3			
		F3	item 4		
			F4	item 5	
				F5	item 7

N square chart



Functional flow

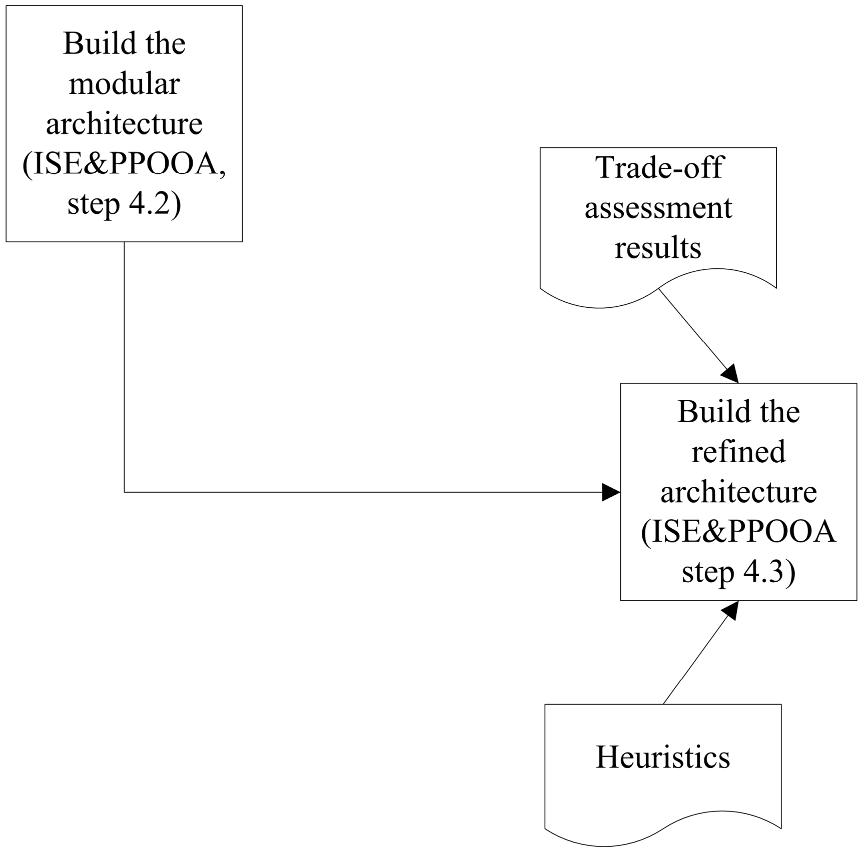
Functions description
(text and tabular format)

Functions description

Step 4.2. Represent modular architecture

- **Goal:** The selection of the solution is based mainly on the clustering of functions to obtain a modular architecture
- **Deliverable**: The deliverable is the first version of the physical architecture. **The modular architecture** is represented by the system decomposition into **subsystems** and **parts** using a SysML block definition diagram. This diagram is complemented with SysML internal block diagrams representing the system physical blocks with either logical or physical connectors for each subsystem identified, and activity and state diagrams for behavioral description as needed. A textual description of the system blocks is provided as well
- **Profiles:** Systems architects.

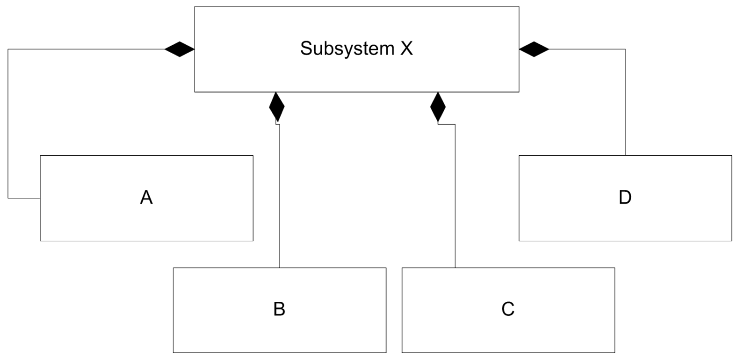
Heuristics and tradeoff



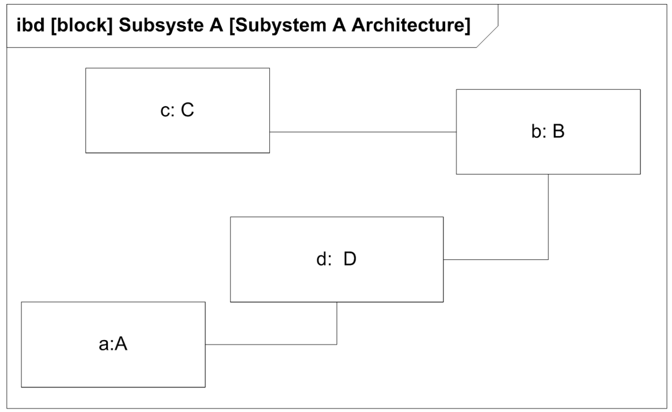
Step 4.3. Refine the architecture

- **Goal:** The modular architecture of the previous step is **refined** considering the implementation of nonfunctional or quality attributes requirements. **Design heuristics** are used for taking into account the system nonfunctional requirements. So the heuristics are the means of satisfying a nonfunctional requirement by manipulating some aspect of a quality attribute model through design decisions. Currently the heuristics presented in Chapter 6 are related to **maintainability, safety, efficiency and those related to resilience**
- **Deliverable**: Each used heuristic may be documented with the template presented in the book. The heuristics collection is an asset that will be updated with the experience of the projects closed. Trade studies may be performed to select the preferred physical architecture that optimizes the measures of effectiveness that may be defined in step 2.a. See Chapter 11 of the book for trade studies
- **Profiles:** Systems architects with the collaboration of customers and experts in some quality domains for example security or safety.

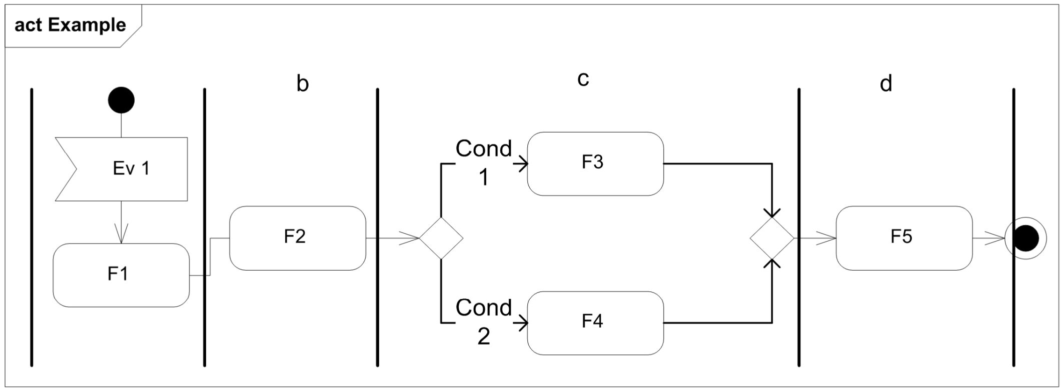
System physical architecture



Physical hierarchy



IBD



Parts Descriptions
(text and tabular format)

Allocated Functional Flow

What about software intensive subsystems?

We propose to use the Pipelines of Processes in Object Oriented Architecture (PPOOA) architectural framework.

The integration between the systems engineering modeling process **ISE** and the **PPOOA** software architecture modeling process is achieved by using a **responsibility** driven software analysis approach supported by **CRC** (Class Responsibility Collaborator) **cards**, a technique proposed by the object oriented community.

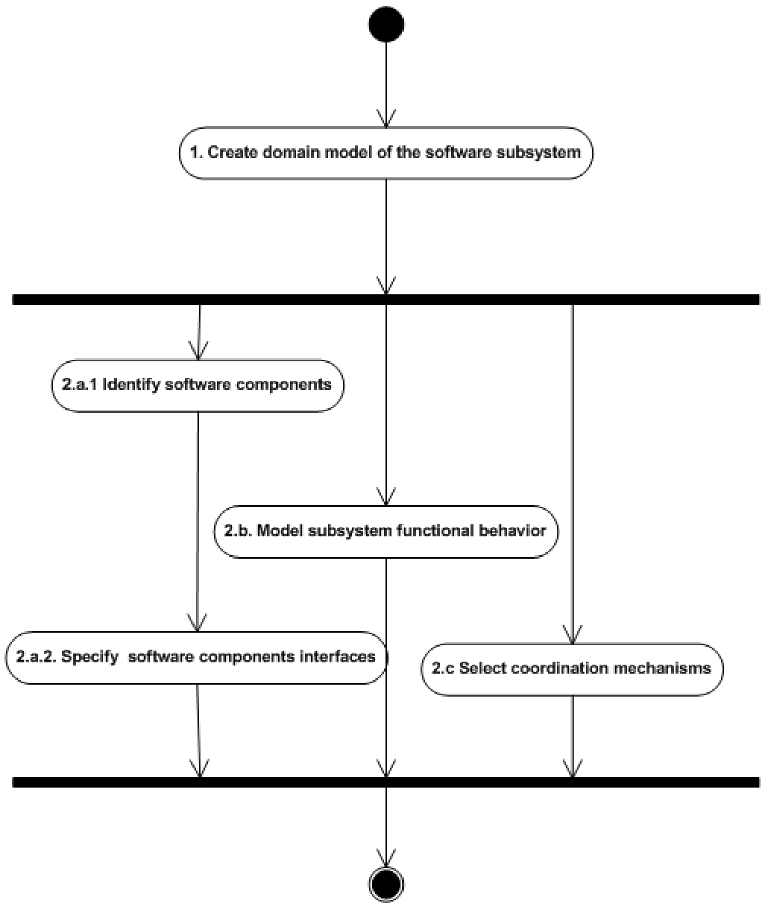
PPOOA

- PPOOA is more than a process, it is an **architecture framework** oriented to design the software of real-time systems.
- PPOOA uses two viewpoints: **structural**, using UML class diagrams extended with PPOOA stereotypes, and **behavioral**, supported by UML activity diagrams

PPOOA vocabulary

- The PPOOA vocabulary of building elements is composed of **Components** and **Coordination mechanisms**
- A **component** is a conceptual computation entity that performs some responsibility and may provide and require interfaces to other components
- A **coordination mechanism** provides the capabilities to synchronize or communicate components of the software architecture

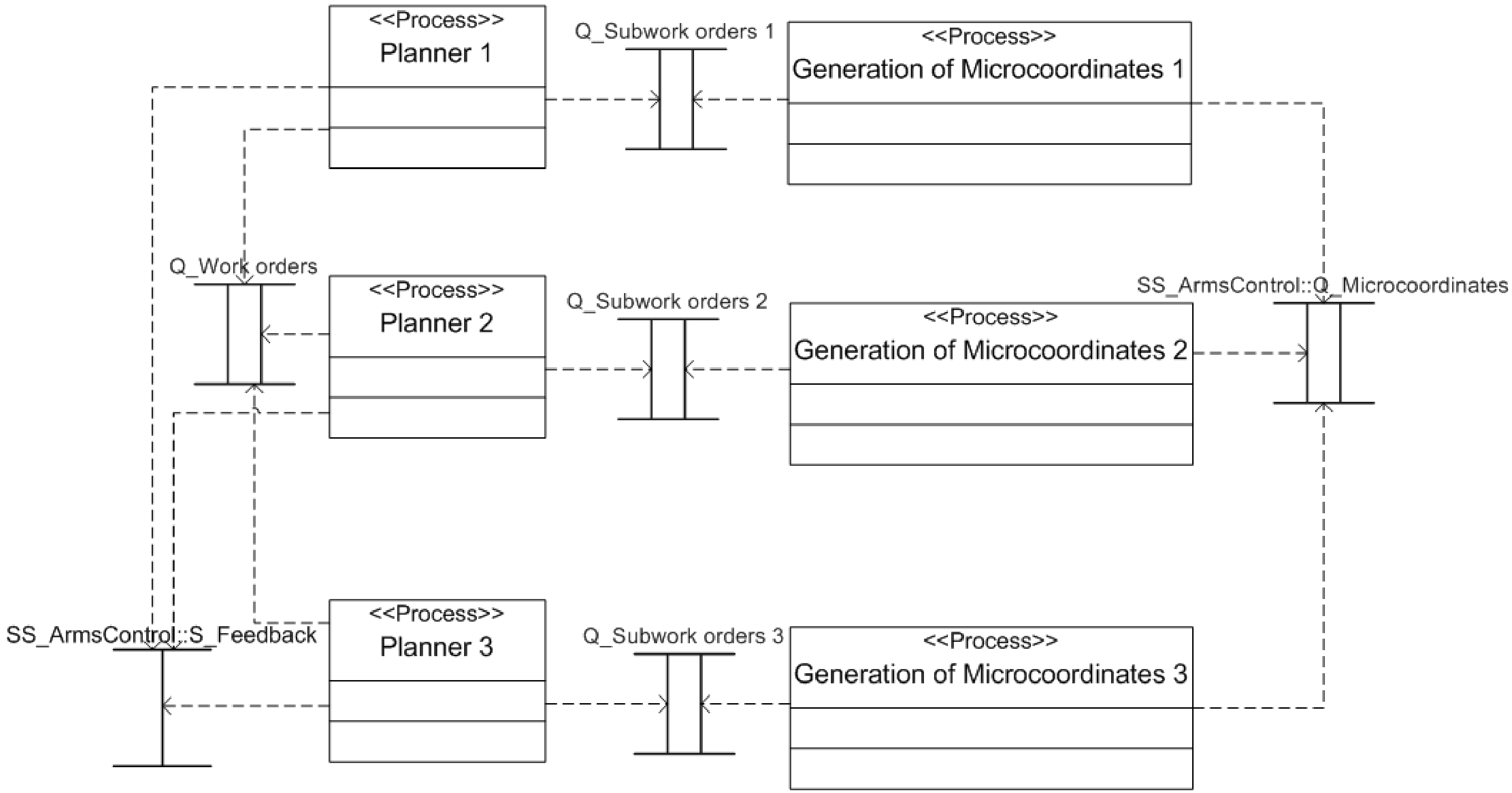
PPOOA architecting subprocess of ISE&PPOOA



Step1. Create a domain model

- A **domain model** yields a more precise specification of software requirements than project team has in the results from earlier system requirements
- A domain model is described using more formalism than textual descriptions, for example UML class diagrams
- A domain model is an essential input when the subsystem is shaped in software architecture, design and implementation
- **CRC cards** are index cards, one for each domain model class, upon which the responsibilities of the class are briefly documented and a list of classes collaborated with to achieve those responsibilities

Structural view of the SW Architecture



Unmanned Aerial Vehicle-Electrical subsystem

Example overview

- The following slides illustrate how the methodology has been used to engineer the Seeker UAV (Unmanned Aircraft Vehicle) by Aurea Avionics
- It is a fixed-wind lightweight UAS (Unmanned Aircraft System) designed to support and cover ISR (Intelligence, Surveillance and Reconnaissance) missions
- This example describes the electrical subsystem design, which is not software intensive

Seeker

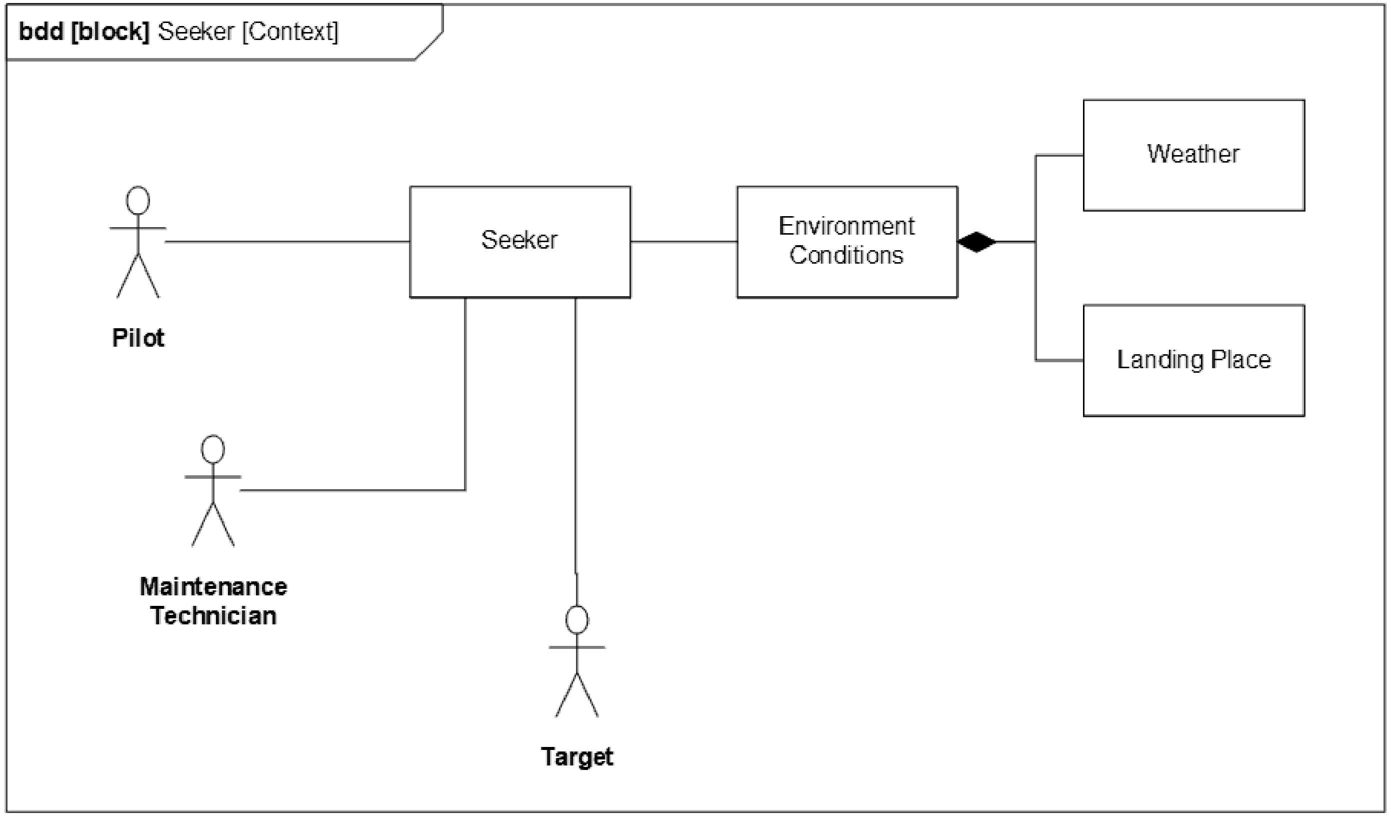
Endurance:	90 minutes
Weight:	3.5 kg
Span:	2.0 m
Length:	1.2 m
Takeoff:	Hand-Launched
Landing:	Belly landing
Link range:	15km LOS
Cruise speed:	60km/h
Operating altitude:	100m-400m

Unmanned Aerial Vehicle

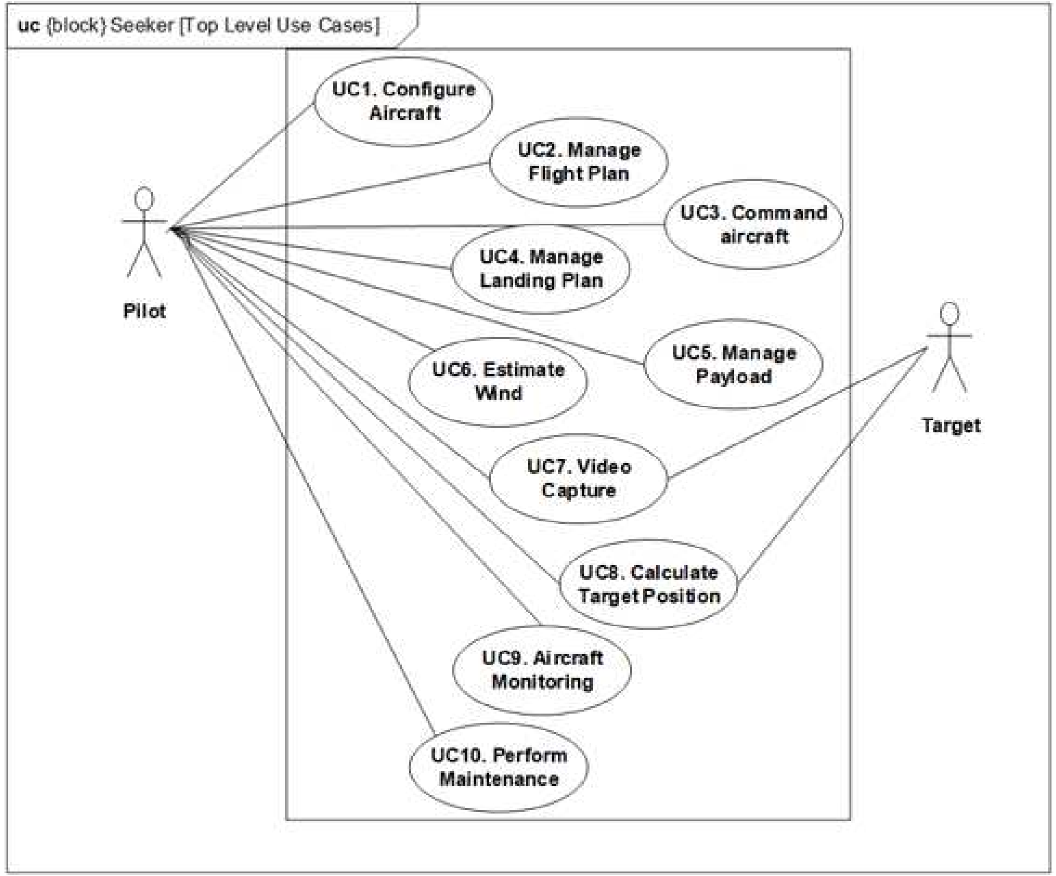
Mission dimension

Scenarios, needs and capabilities

Context diagram



Use cases diagram



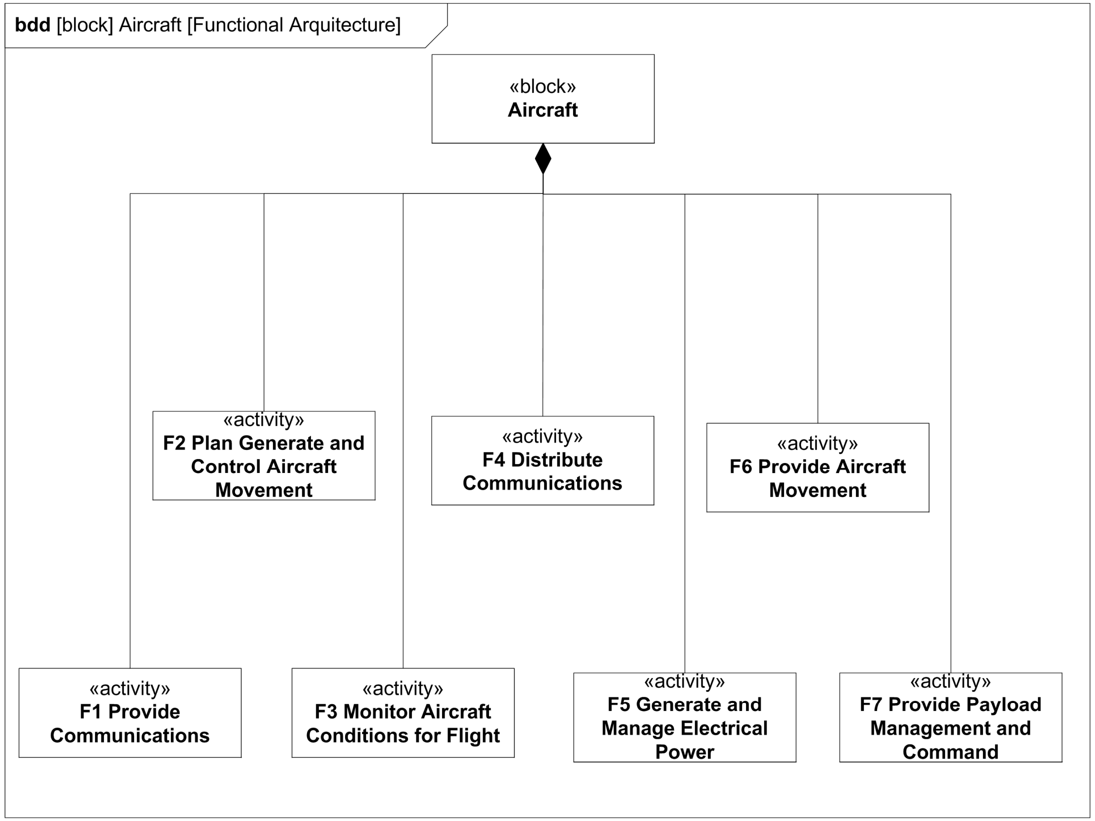
Capabilities

Id	Capability	Description
C1	Easy aircraft configuration	The ground control software shall be configured in order to set the parameters and variables of the aircraft.
C2	Guide to system verification	Before the flight, it's necessary to verify the behavior of the sensors, the structural integrity, communications, etc. The system software must provide a guide to the operator to perform the verification process.
C3	Flight planning management	It's necessary to define and to upload the flight plan prior to the flight and to change it during the flight.
C4	Autonomous flight	The aircraft shall flight in an autonomous mode following the flight plan.
C5	Send and receive communications	The aircraft shall send telemetry and receive commands.
C6	Command payload	The payload (gimbal) shall be pointed and perform zoom.
C7	Send video	The aircraft shall send real time video continuously.
C8	Support wind estimation	The aircraft shall estimate wind and send the estimation to the ground station.
C9	Long endurance	The endurance of the aircraft shall be higher than an hour.
C10	Range	The range of the aircraft shall be higher than 10 km.
C11	Easy assembly	The aircraft shall be assembled for flight in less than 5 minutes.
C12	Easy to transport	The whole system shall be transported in a box or in two medium size bags.
C13	Safe during GPS signal outages	The aircraft shall return to base in case of GPS signal outage.
C14	Low noise footprint	The aircraft shall fly in a silence mode.

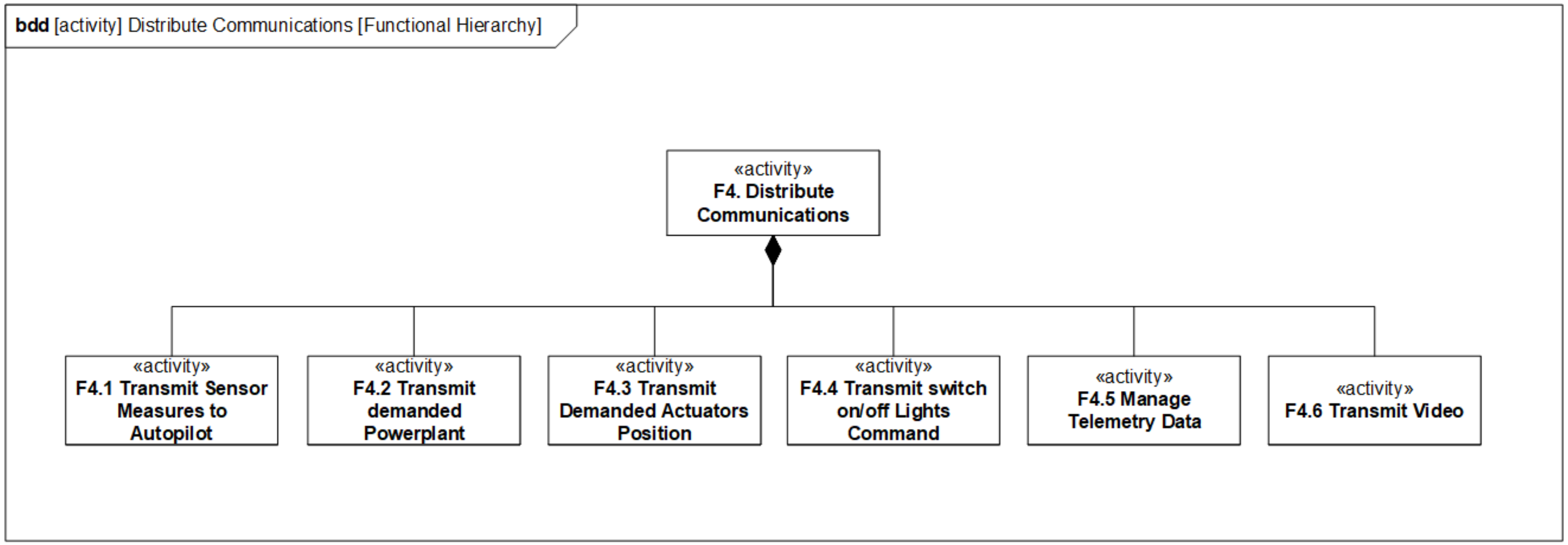
System dimension

Functional architecture (functions+behavior), system requirements and physical architecture

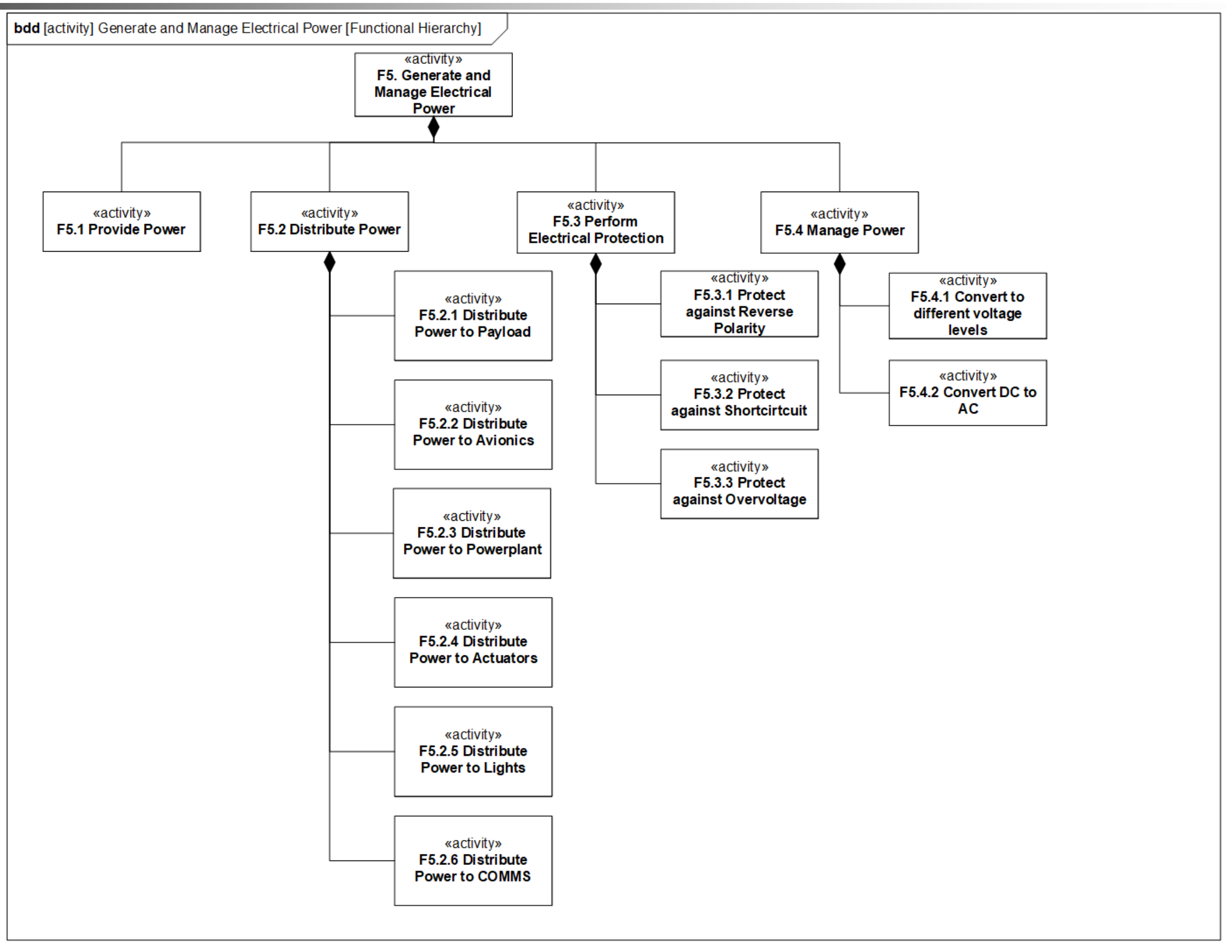
Top level functions



F4.Distribute communications



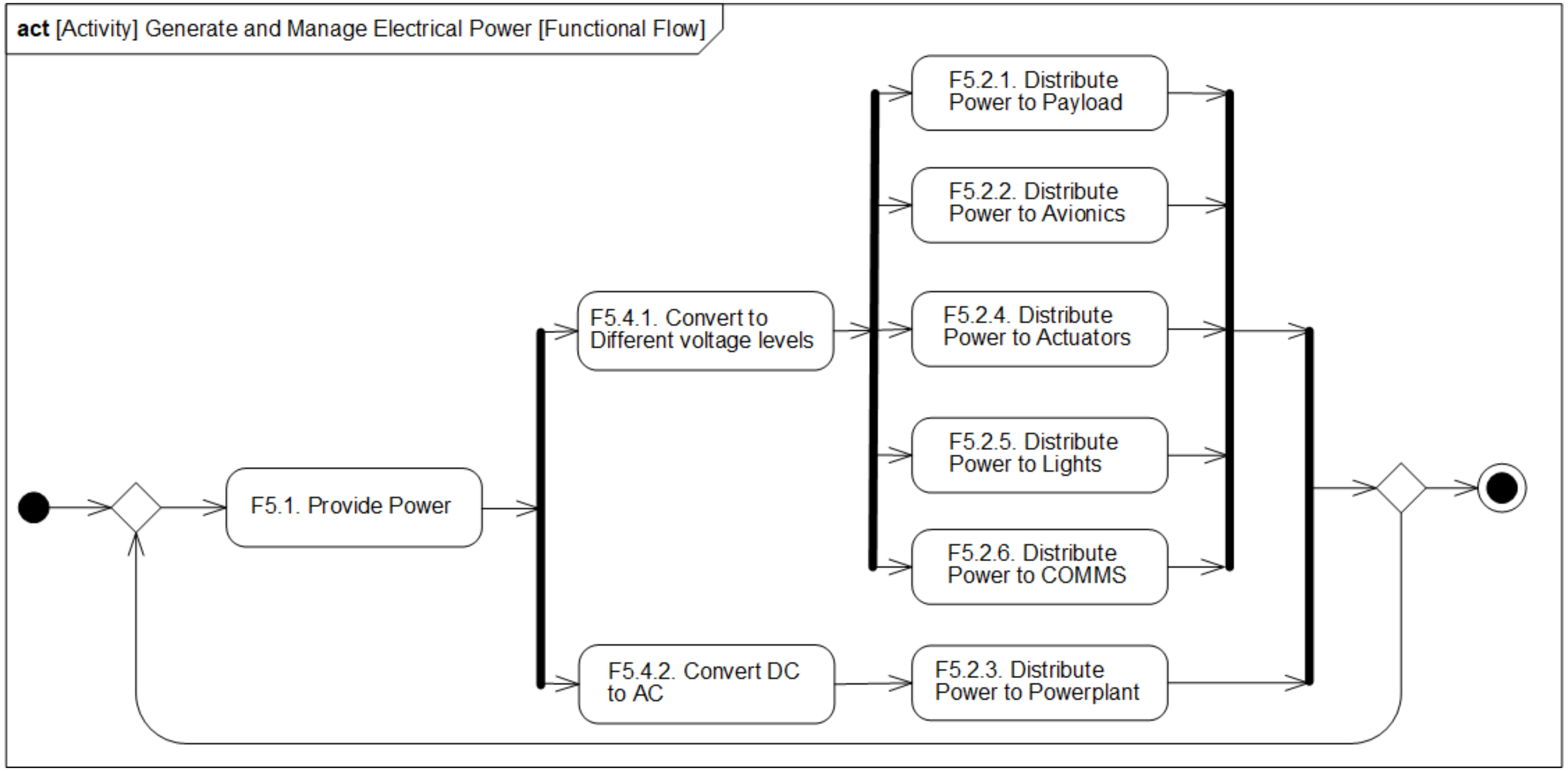
F5. Generate and manage electrical power



Example of function textual description

Function:	**F5.2 Distribute Power.**
Description:	Power distribution to the aircraft subsystems.
Inputs:	Conditioned power
Outputs:	Electrical power to systems.
Parent function:	F5.
Children functions:	Distribute Power to Payload, Distribute Power to Avionics, Distribute Power to power plant, Distribute Power to Actuators, Distribute Power to Lights, Distribute Power to COMMS.

Example of functional flow



Functional interfaces-N^2 chart

stored power from battery												
F5.1. Provide Power							electrical power	electrical power	electrical power			
	F5.2.1. Distribute Power to Payload											power to payload
		F5.2.2. Distribute Power to Avionics										power to avionics
			F5.2.3. Distribute Power to Power Plant									power to power plant
				F5.2.4. Distribute Power to Actuators								power to actuators
					F5.2.5. Distribute Power to Lights							power to lights
						F5.2.6 Distribute Power to COMMS						power to comms
							F5.3.1. Protect against Reverse Polarity			reverse polarity protected electrical power	reverse polarity protected electrical power	
								F5.3.2. Protect against Shortcircuit		shortcircuit protected electrical power	shortcircuit protected electrical power	
									F5.3.3. Protect against Overvoltage	overvoltage protected electrical power	overvoltage protected electrical power	
	conditioned DC power	conditioned DC power		conditioned DC power	conditioned DC power	conditioned DC power				**F5.4.1. Convert to DC voltage levels**		
			conditioned AC power								**F5.4.2. Convert DC to AC**	

Some system requirements/functional

Function: Provide Power

- FR_5.1 The function Provide Power shall provide electrical power to the aircraft subsystems autonomously.

Function: Distribute Power

- FR_5.2 The function Distribute Power shall distribute electrical power to all the electrical equipment.

Rationale: The electrical equipment will depend on the final electrical subsystem solution.

Function: Perform Electrical Protection

- FR_5.3a The function Perform Electrical Protection shall provide protection against reverse polarity.
- FR_5.3b The function Perform Electrical Protection shall provide protection against short-circuits.
- FR_5.3c The function Perform Electrical Protection shall provide protection against overvoltage.

Function: Manage Power

- FR_5.4a The function Manage Power shall provide electrical power to different DC levels .

Rationale: DC levels will depend on the final electrical subsystems selected. The DC levels shall be determined during the detailed design.

- FR_5.4b The function Manage Power shall provide electrical AC power.

Some system requirements/nonfunctional

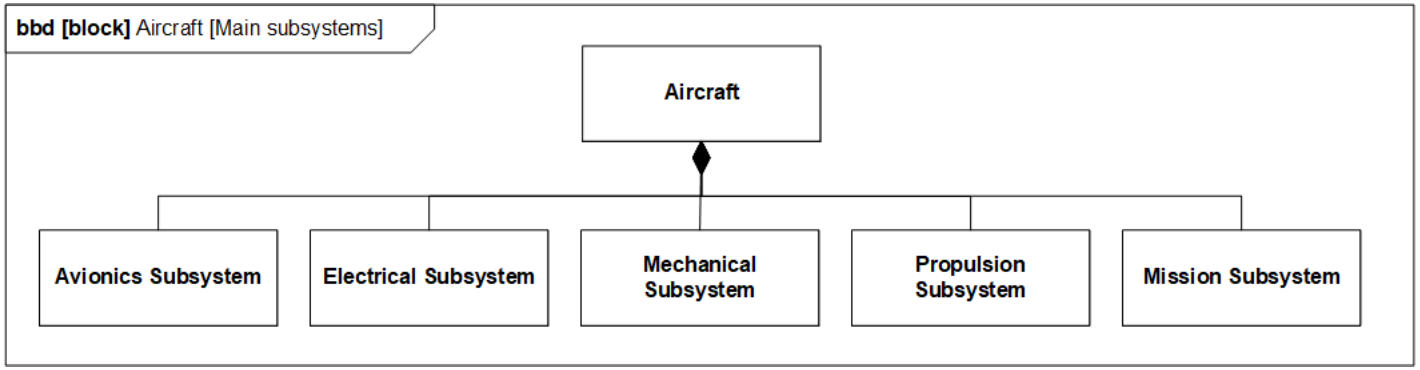
Maintainability(intentionally incomplete)

- NFR_Maint_005. The operator shall mount the aircraft without tools
- NFR_Maint_006. Replaceable parts of the aircraft shall be substituted in less than 5 minutes

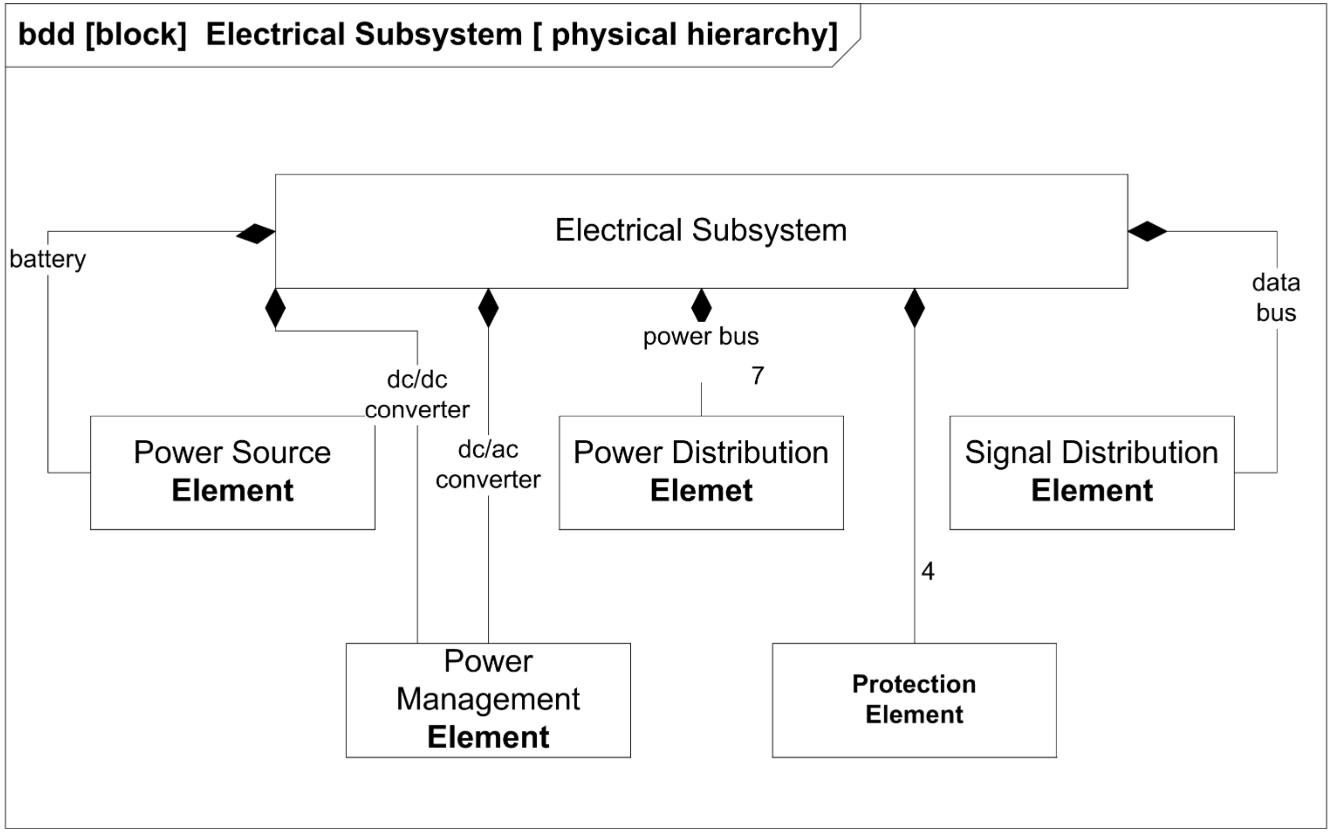
Rationale: replaceable parts include those who are liable to be broken or damaged during the operation, in case of a harsh landing or due to harsh environmental conditions. Those parts typically include motor’s blade and payload, for instance. Hence, once the final selection of components is completed, a tag “replaceable” will be included when appropriate and will help list these components and keep their traceability

- NFR_Maint_007. Embedded software shall be updated without disassemble the aircraft.
- NFR_Maint_008. Aircraft shall provide a port to check the status of the aircraft

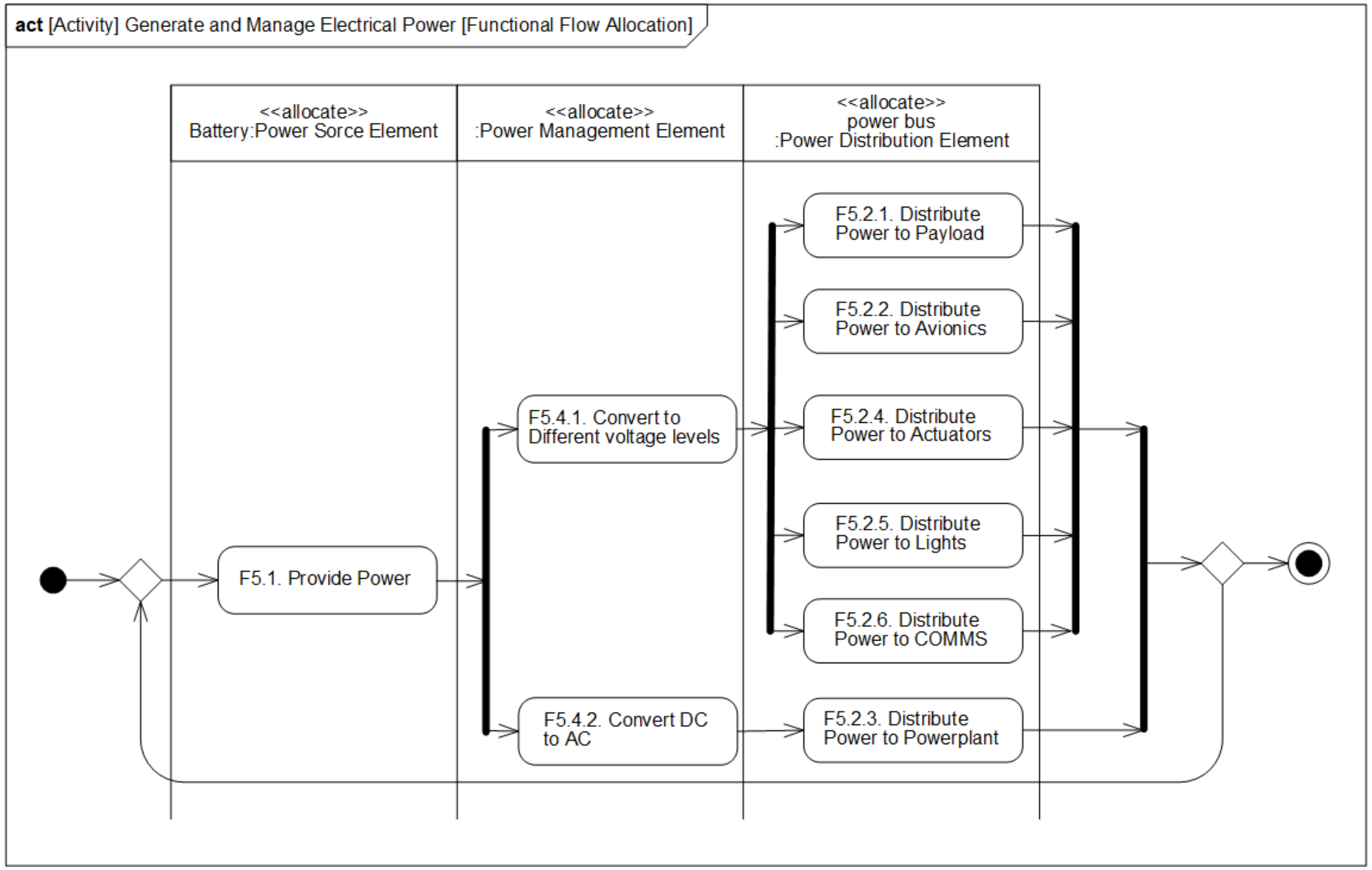
Physical architecture-main subsystems



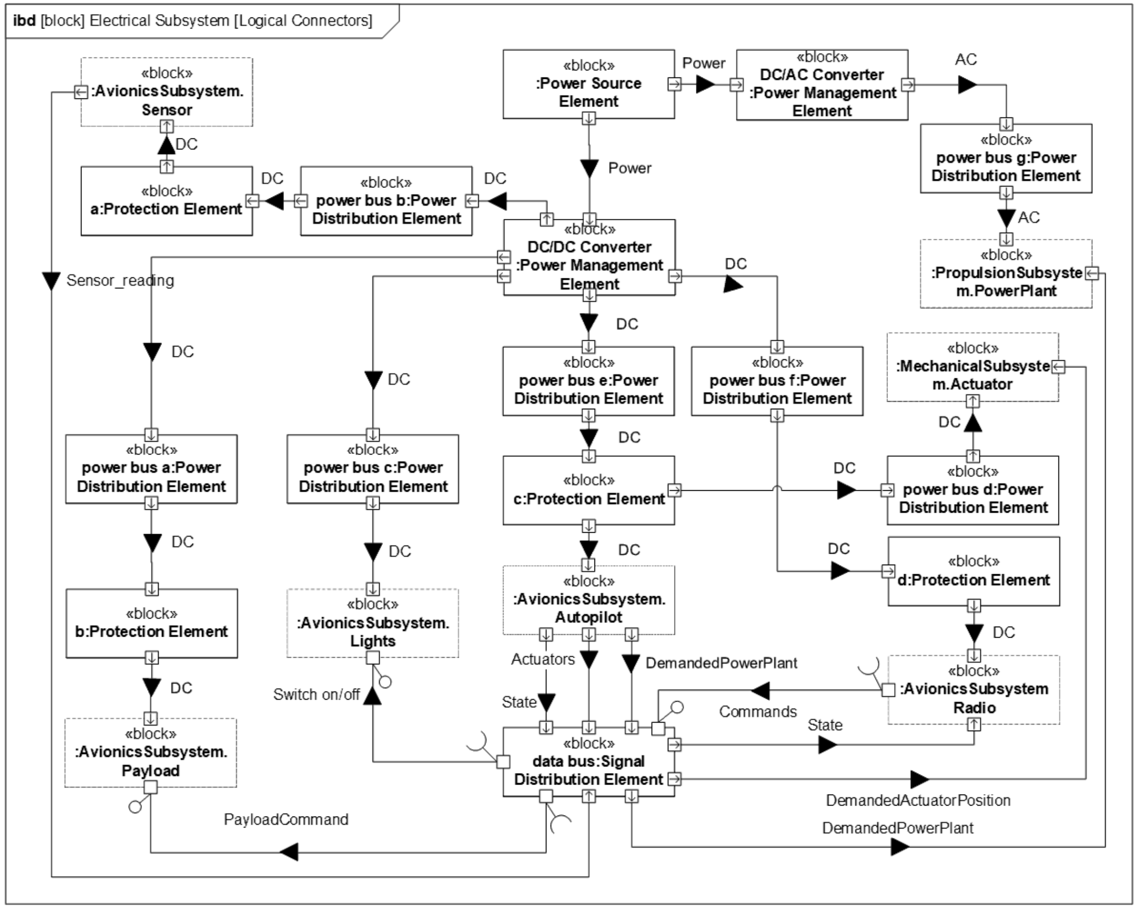
Physical architecture- Electrical subsystem bdd



Functional allocation



Physical architecture-ibd with logical connectors



Heuristics applied

- **SA_Heu_6**. Group and allocate functions that are strongly related to each other, separate functions that are unrelated.
- **SA_Heu_8**. Choose a configuration with minimal communications between the subsystems.
- **Man_Heu_8.** Isolate the expected change. Applied here to non-software building elements as well.
- **Man_Heu_9**. Raise the abstraction level. Applied here to non-software building elements as well.
- **Man_Heu_13**. Maintain existing interfaces. Applied here to non-software building elements as well.
- **SF_Heu_2**. Minimize the number of components and interactions.
- **SF_Heu_4**.Enforce time requirements.

Example of physical part description

DC/DC Converter: Power Management Element
In flow: power
Out flow: direct current
Allocated functions: • F5.4.1 Convert to different voltage levels

To conclude

The **ISE&PPOOA** methodology and its book is useful for several audiences: **graduate and undergraduate students of systems engineering** may use it as an instruction text; **experienced systems engineers** trying to apply **MBSE** can learn **how to think** to develop system models; and **specialty engineers** such as mechanical, electrical, software, safety and logistic engineers can use the book as an approachable guide to MBSE that will help them **understand** the information captured in **the system models**.