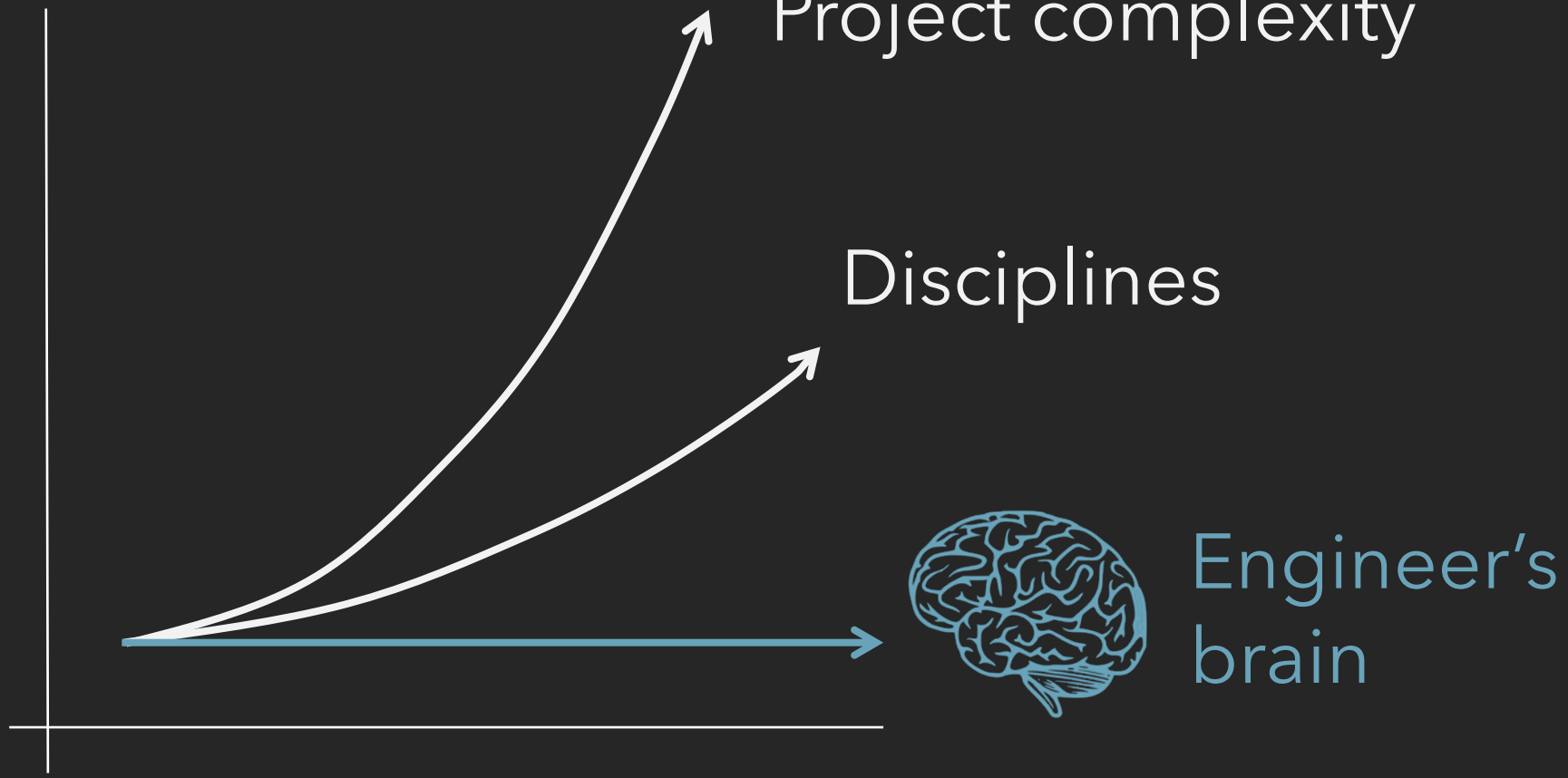
THALES

Fostering MBSE in engineering culture

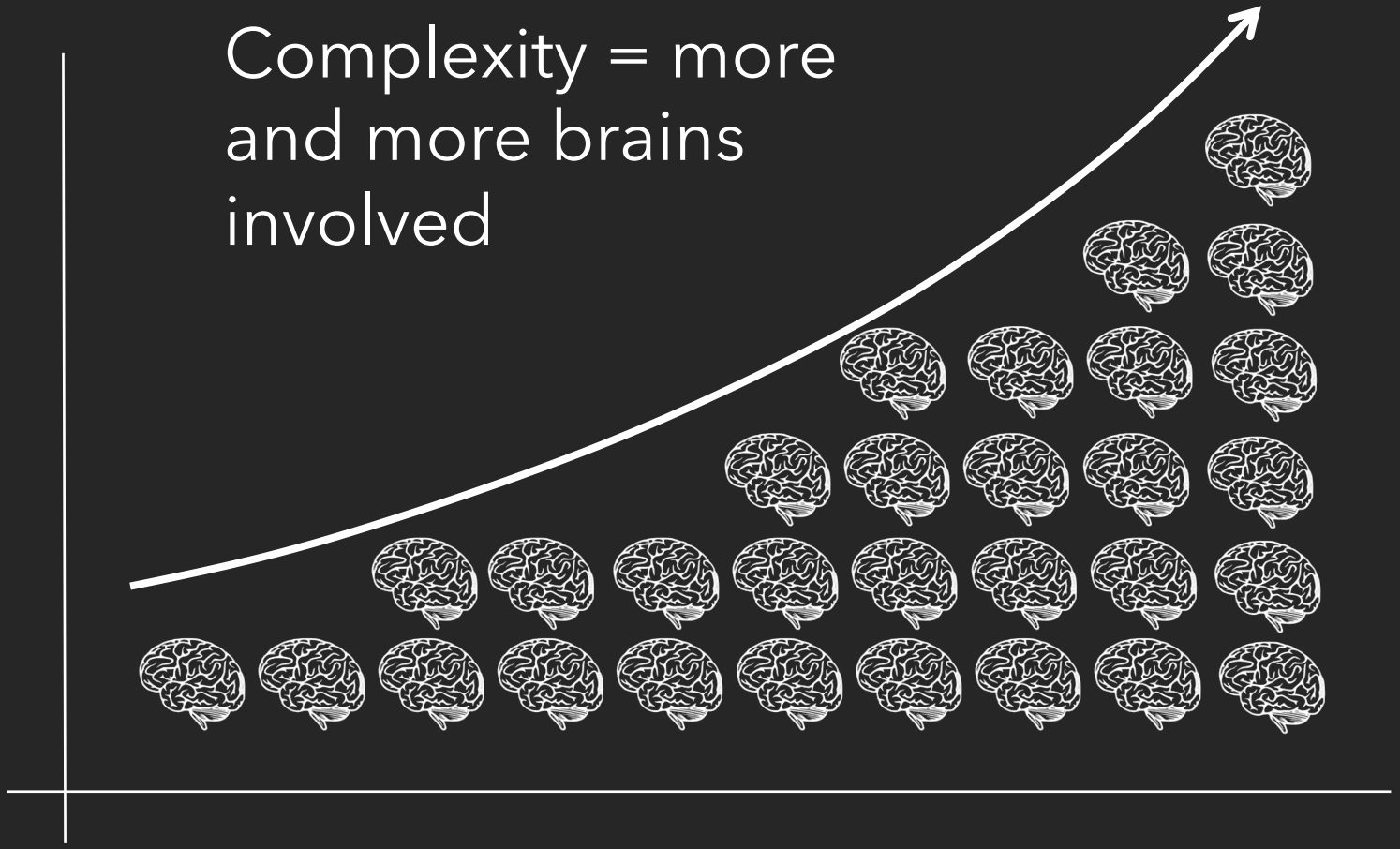
INCOSE INTERNATIONAL WORKSHOP 2020

Stéphane Bonnet
Thales Corporate MBSE Coaching

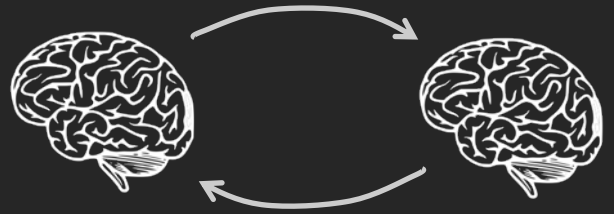
With contributions of Jean-Luc Voirin, Juan Navas, Eric Lepicier, Guillaume Journaux, Karin Pellen, and others


Project complexity
Disciplines
Engineer's brain

Complexity = more and more brains involved

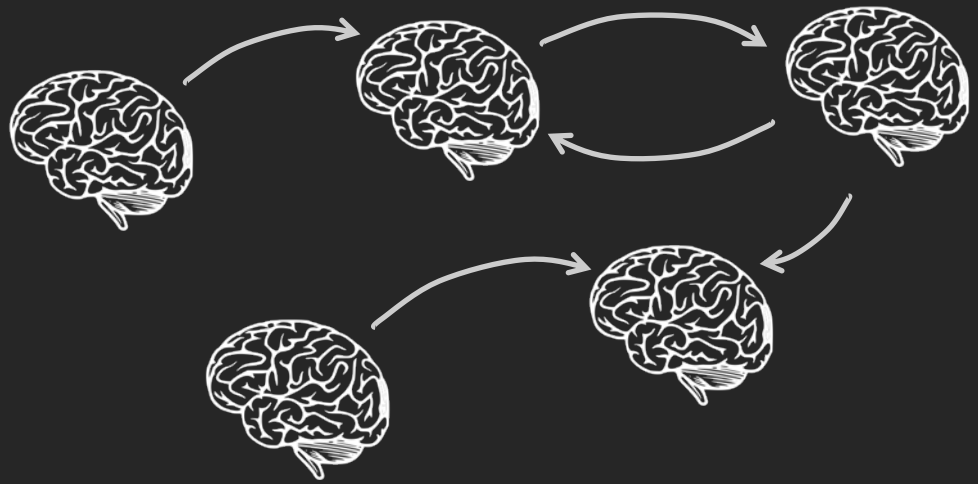


Doing more, with more constraints and less time



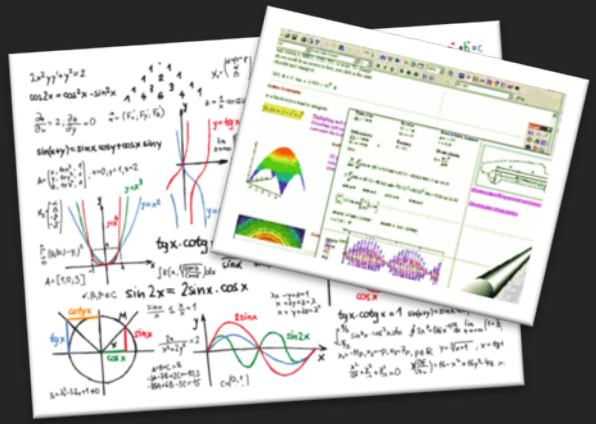
Coping with very demanding customers

Interacting with more peers

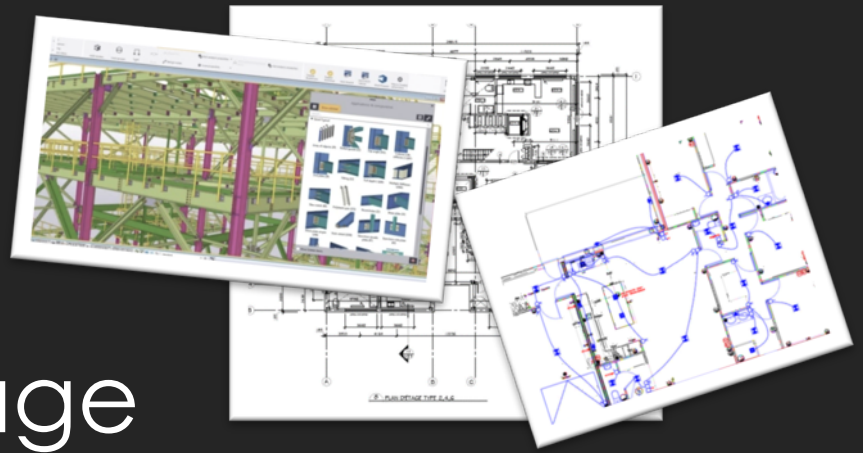


Communication and information management problem

Mathematics



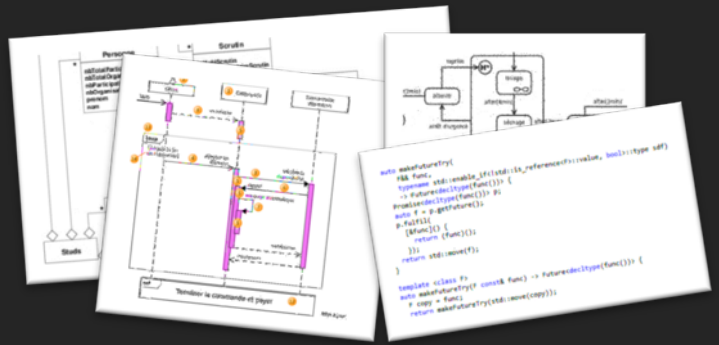
Construction

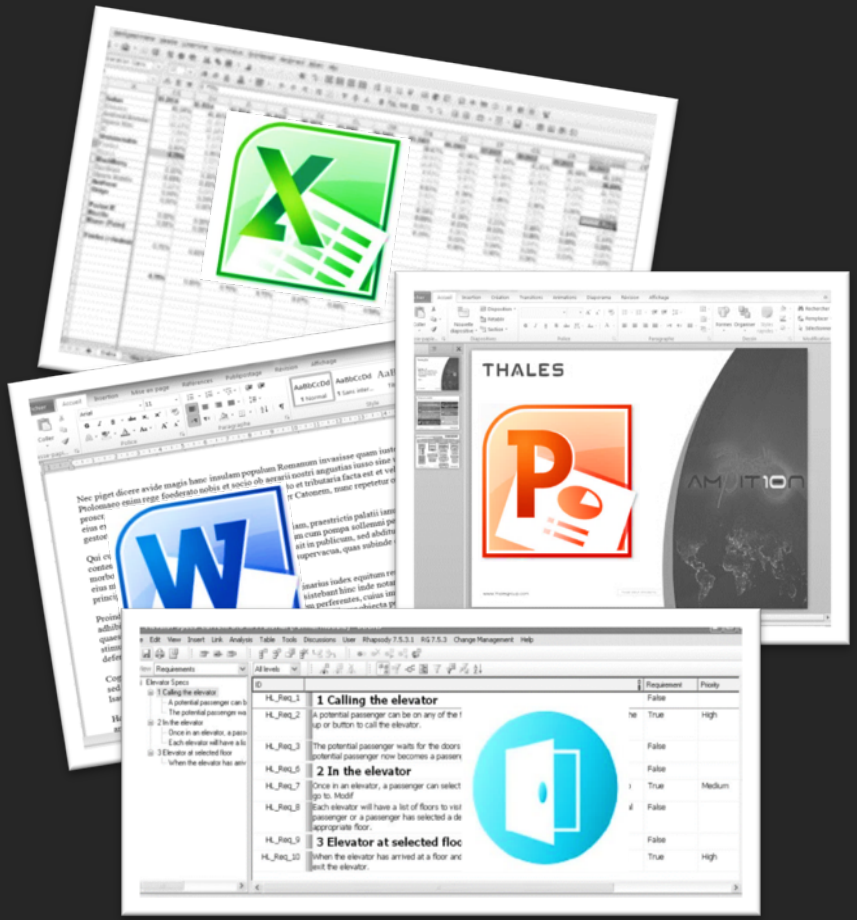


Language

Electronics

Software





Systems Engineering

Model-based systems engineering is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases."

Vision 2020 (INCOSE-TP-2004-004-02, Sep 2007)

1. Fundamentals: method and concepts
2. Engineering practices
3. Framework of (model-based) engineering objectives
4. Organizational aspects of deployment

1. Fundamentals

Methodology and high level concepts and viewpoints

Purpose-built to provide the notation and diagrams fitting the Arcadia approach

Need model

helps formalize and consolidate customer and system requirements

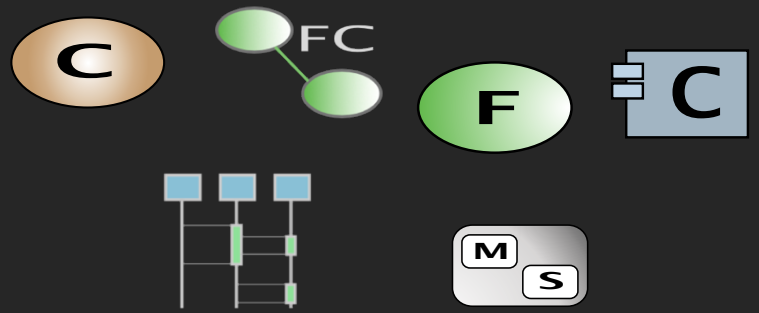
Requirements

are at the heart of the current engineering practices

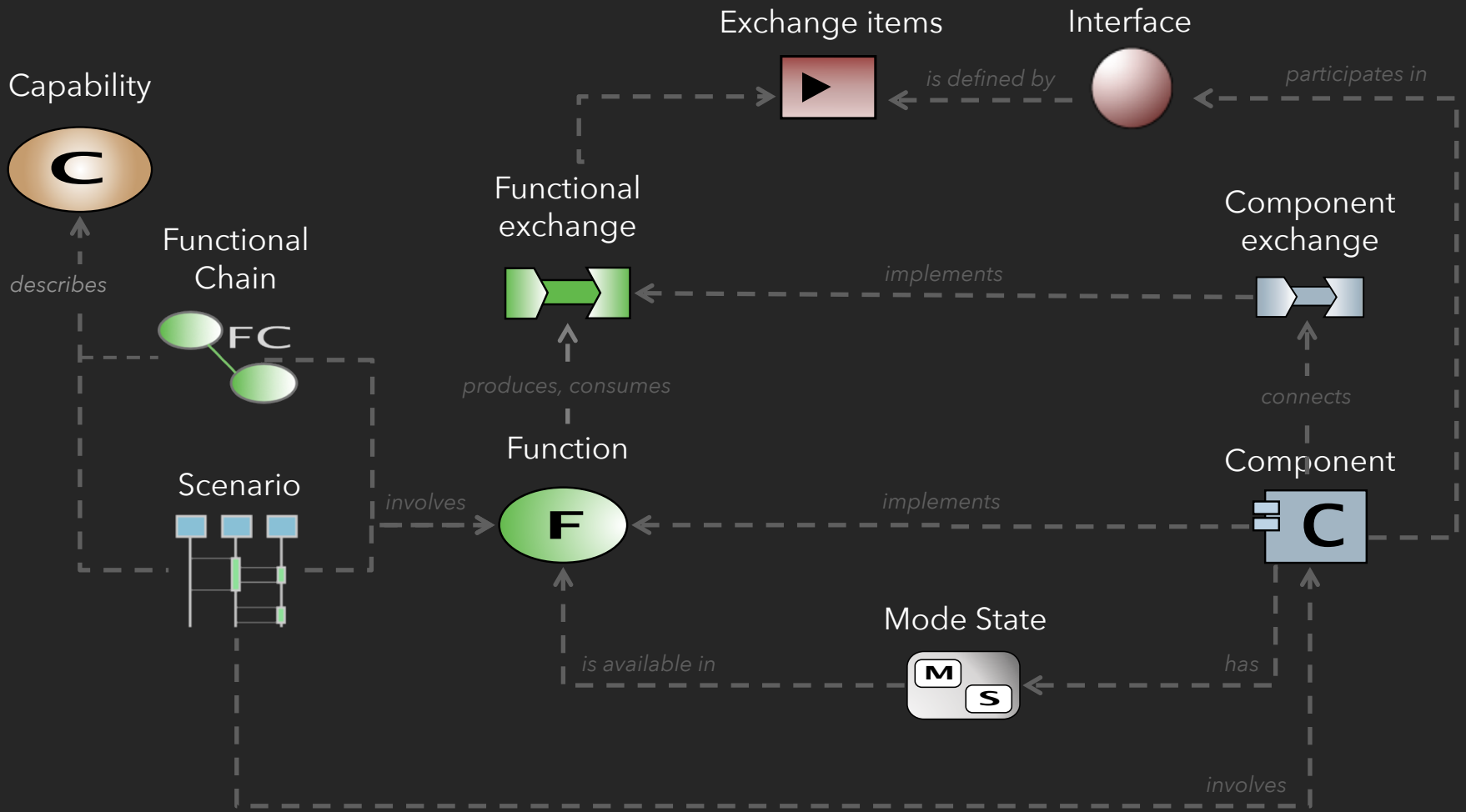
Solution model

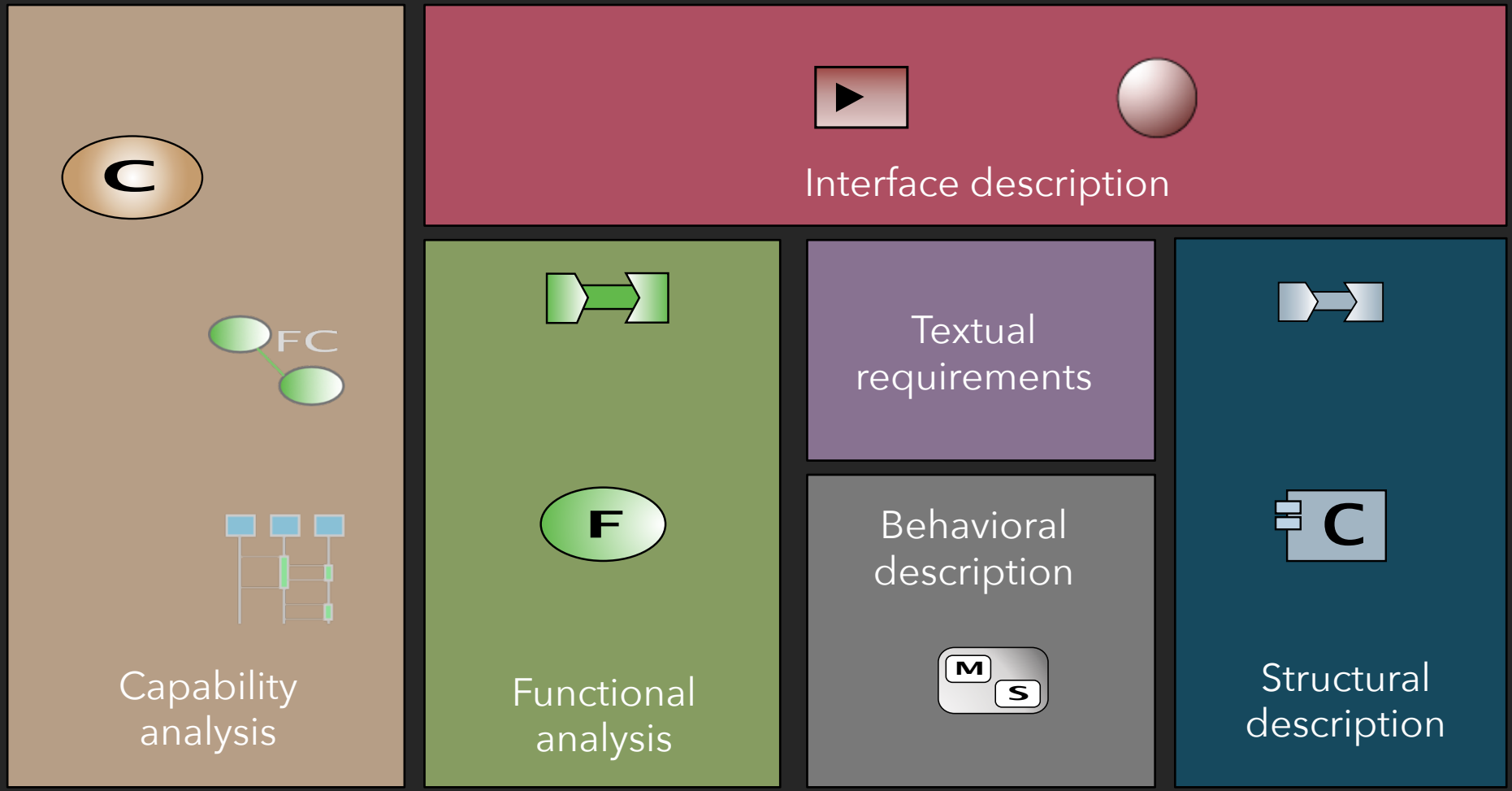
helps validate feasibility, elicit/justify new requirements for system/subsystems

Model concepts



Engineering Perspectives (need + solution)

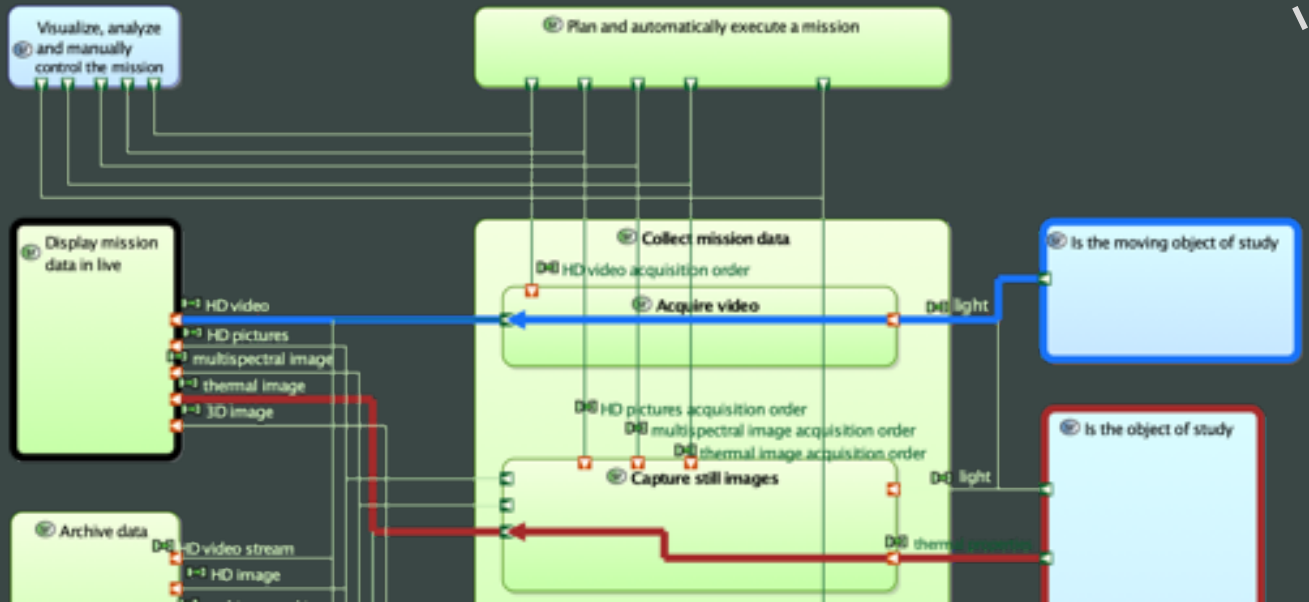

Exchange items
Interface
Capability
is defined by
participates in
C
Functional exchange
Component exchange
Functional Chain
describes
implements
FC
produces, consumes
connects
Function
Component
Scenario
involves
implements
F
C
Mode State
is available in
has
M
S
involves

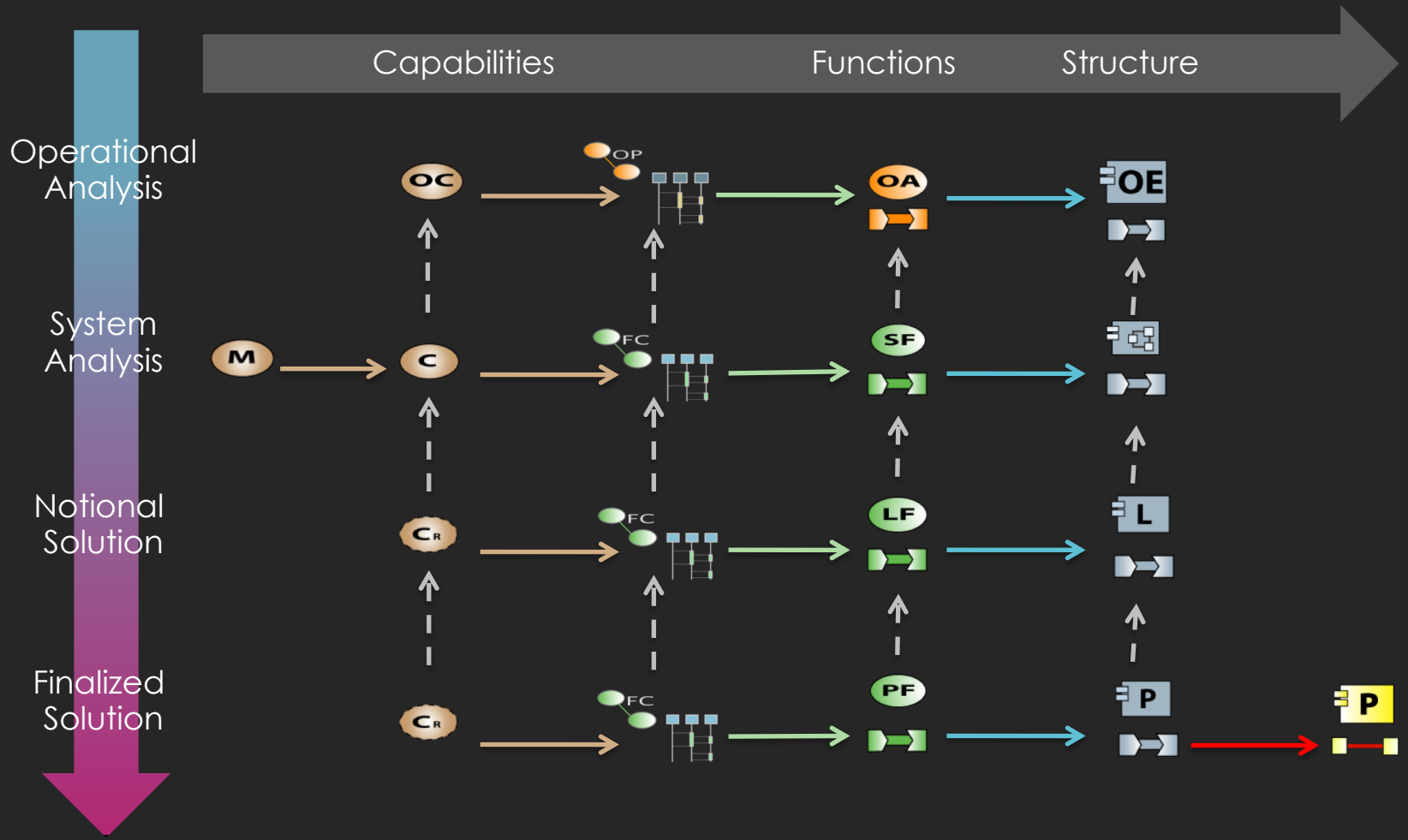

C
FC
Capability analysis
Interface description
F
Functional analysis
Textual requirements
Behavioral description
M
S
C
Structural description

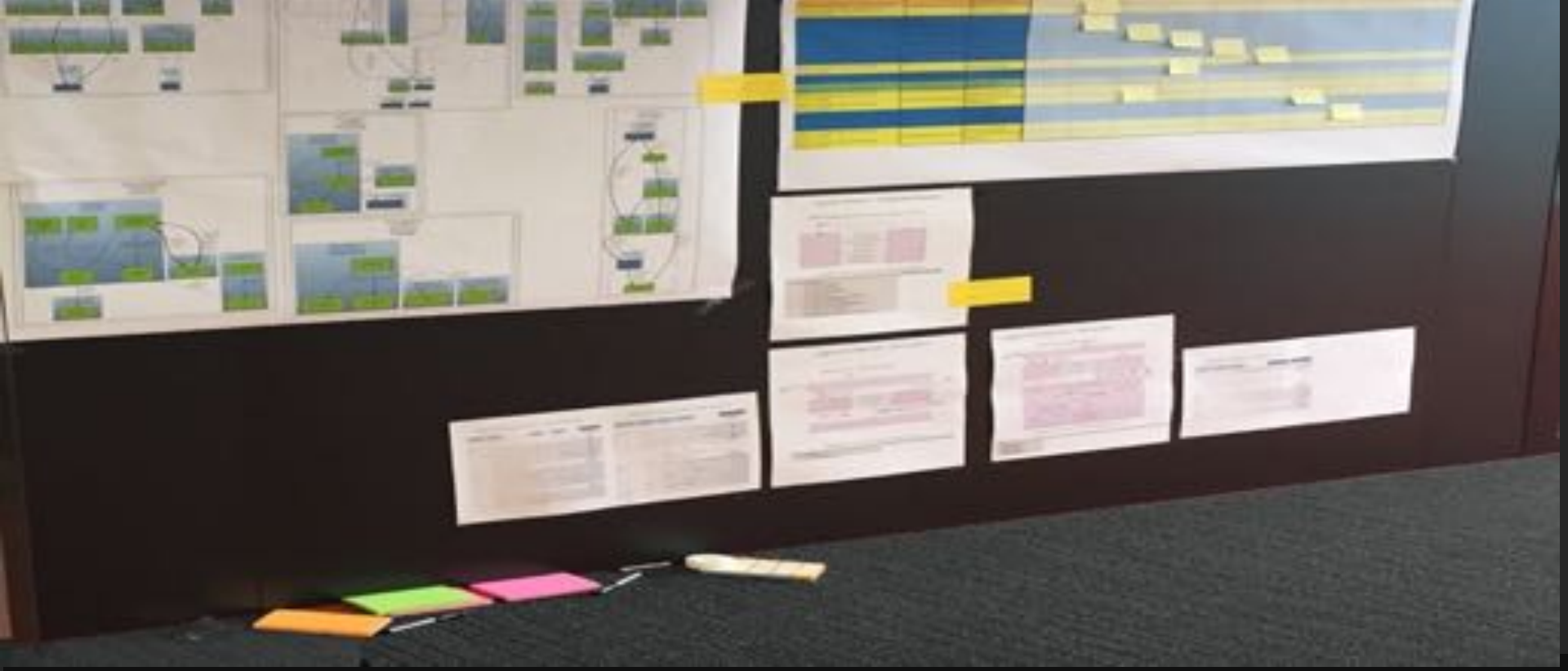
C

Visualize live data during flight

- FC Display live acquired HD video
- FC Display live multi-spectral image
- FC Display live thermal image
- FC Visualize all collected mission data
- FC Visualize substance levels in real time

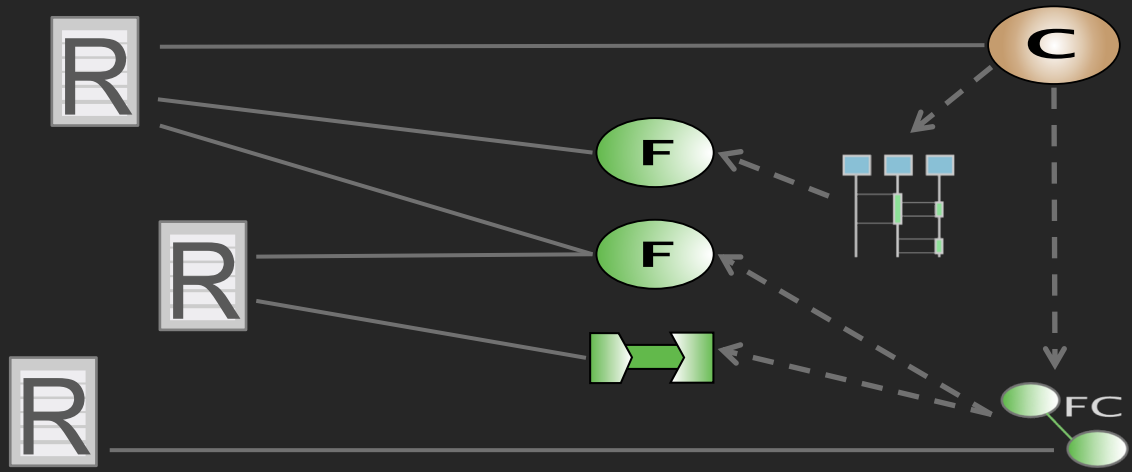



Capabilities
Functions
Structure
Operational Analysis
System Analysis
Notional Solution
Finalized Solution
OC
OP
OA
OE
M
C
FC
SF
CR
LF
L
PF
P
P



2. Engineering practices

Requirements Model elements

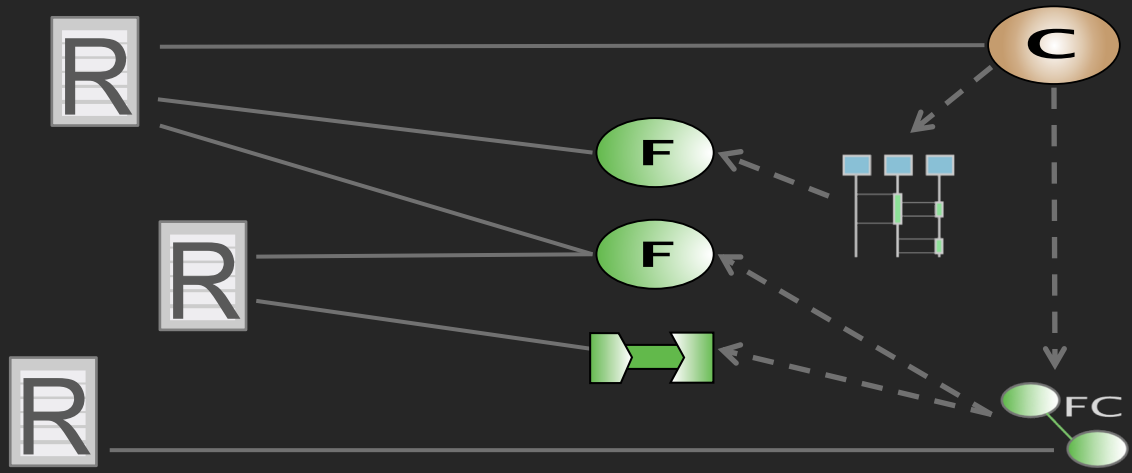


Models add rigor to need expression / solution description

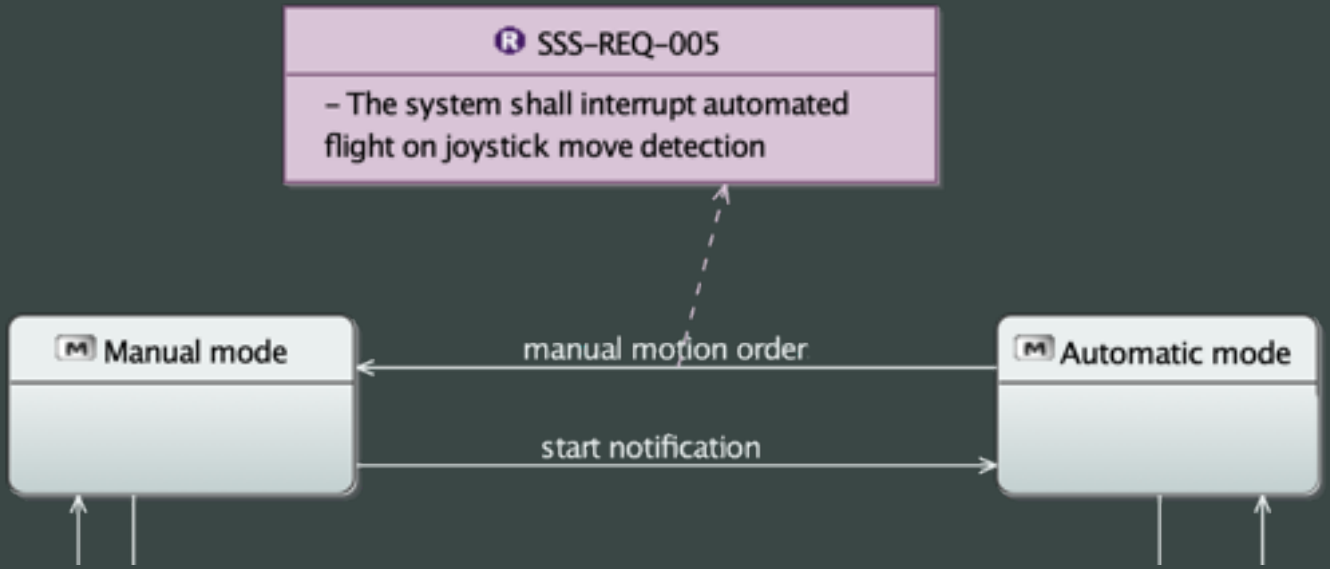
Models enable automated processing

Textual Requirements

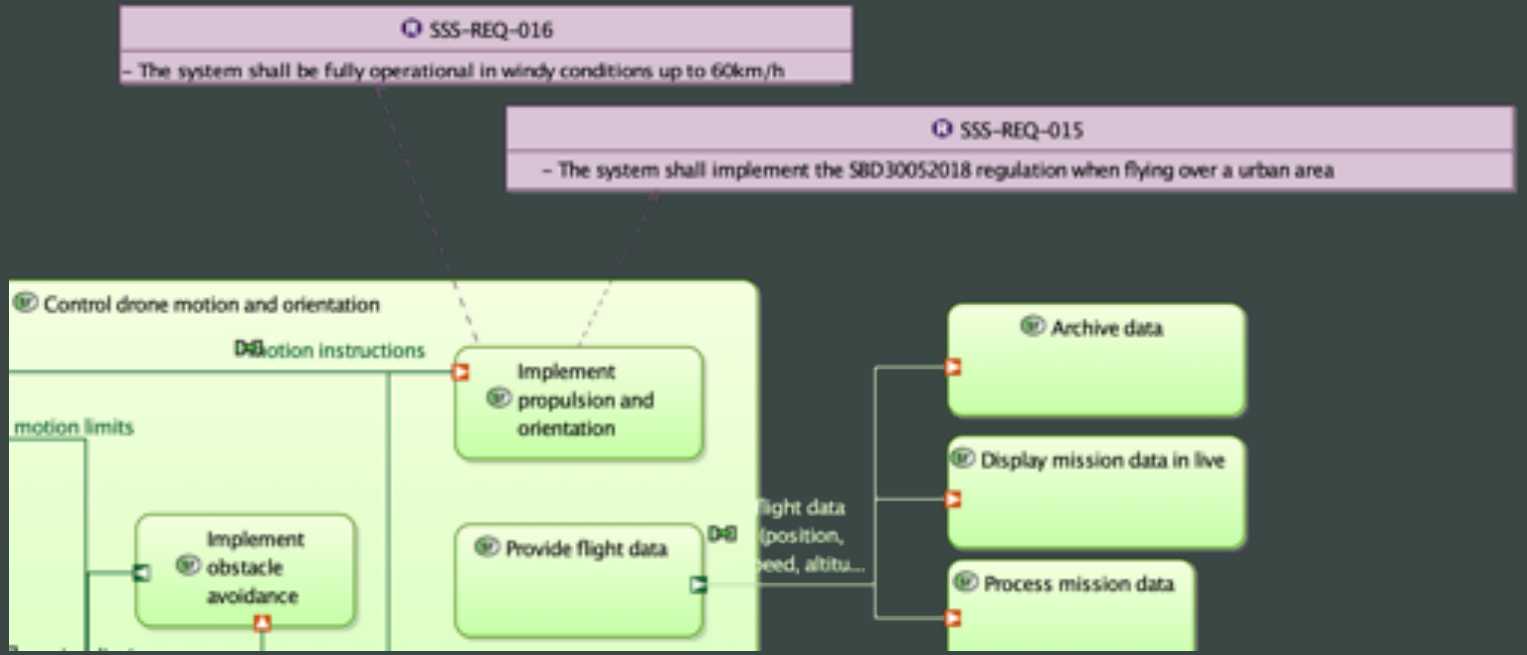
~~Model elements~~ Requirements



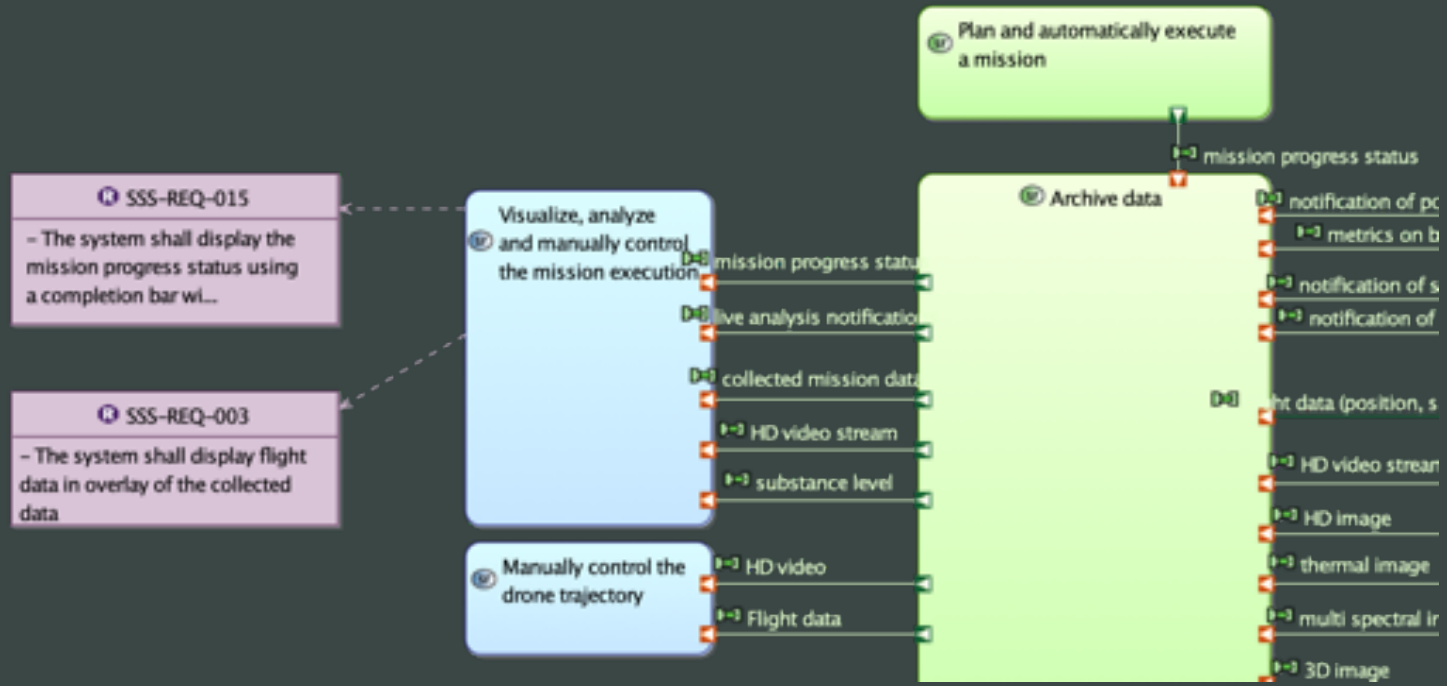
A model requirement can formalize a textual requirement and explicit its effects and ramifications



Some expectations (environmental, regulations, etc.)
are easier to express with textual descriptions.



Some expectations on a model element at a given engineering level do not require a formal modeling (which is left to subsystem design)



Happy consequences

Contracts between engineering levels

Verification and validation

Incremental (agile) development strategy

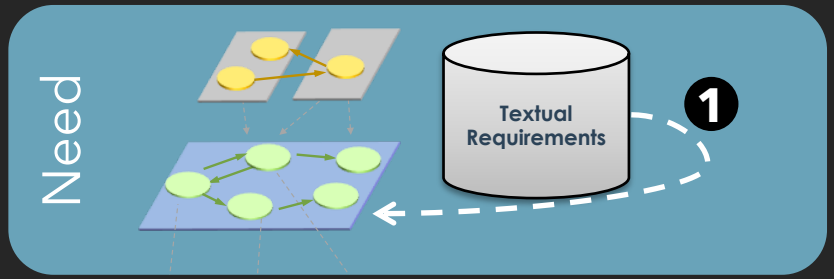
Happy consequences

Contracts between engineering levels

Verification and validation

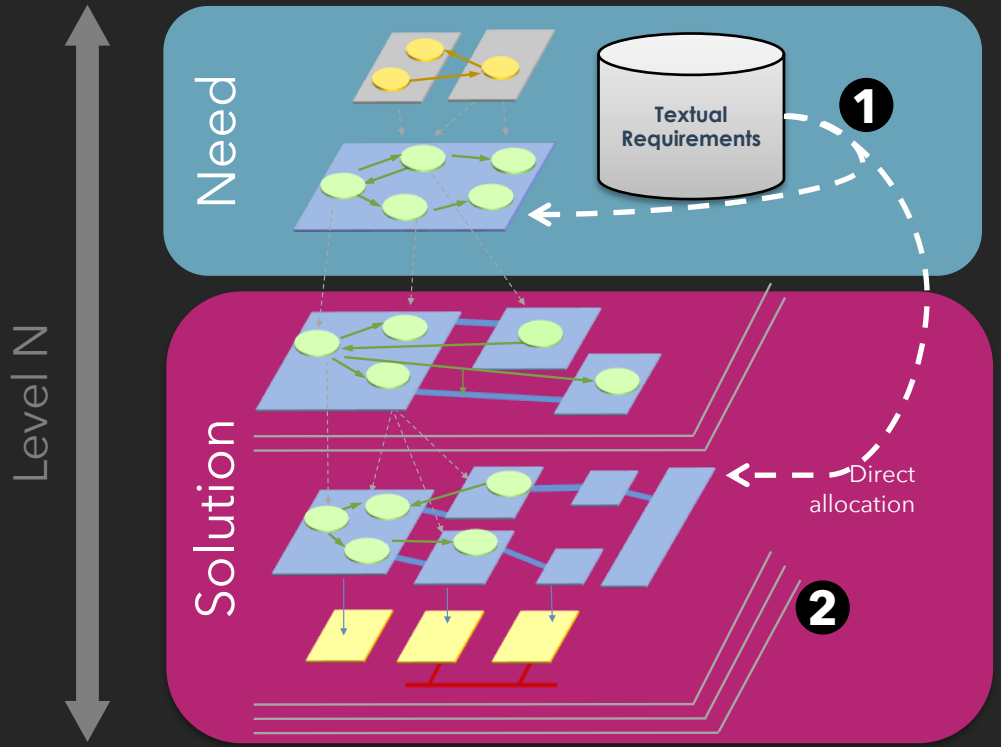
Incremental (agile) development strategy

Level N



1.

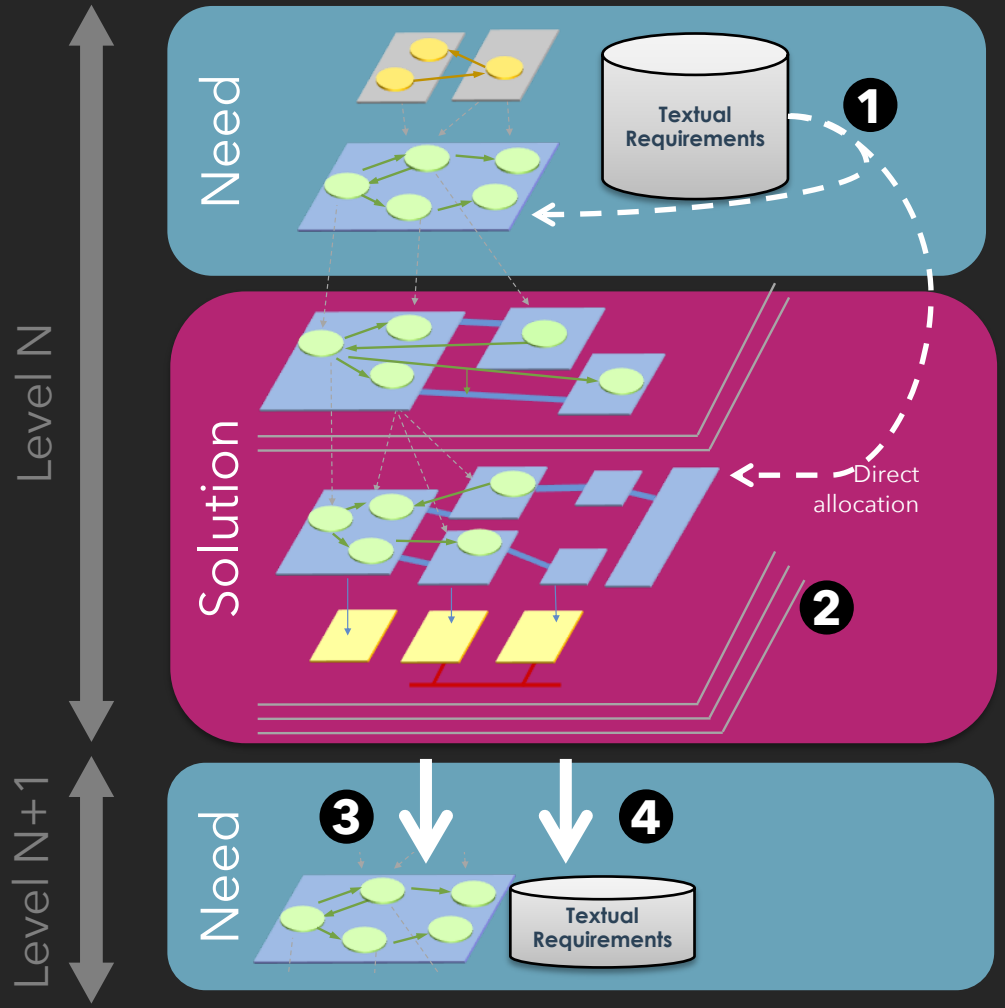
Elicitation of model and textual requirements on the system



2.

Architecture description specifies with the adequate level of detail how the system works and what is expected from each constituent

Objective: Prepare the contracts for all subsystems and guarantee their proper integration.



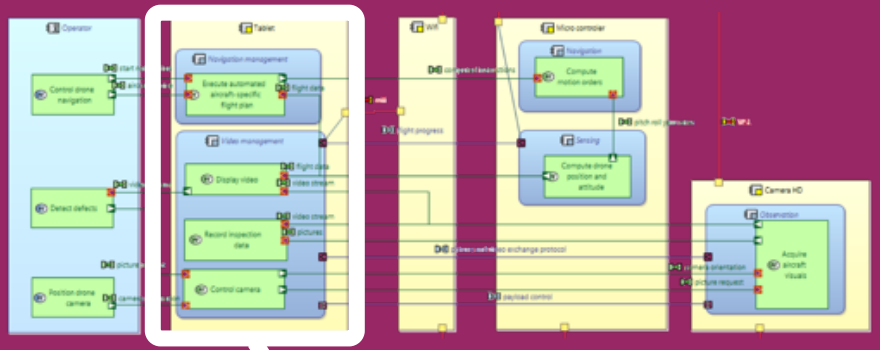
3.

The context of a given system constituent is entirely computed (anything contributing to the definition of this constituent including allocated Functions, interfacing Components, etc.)

4.

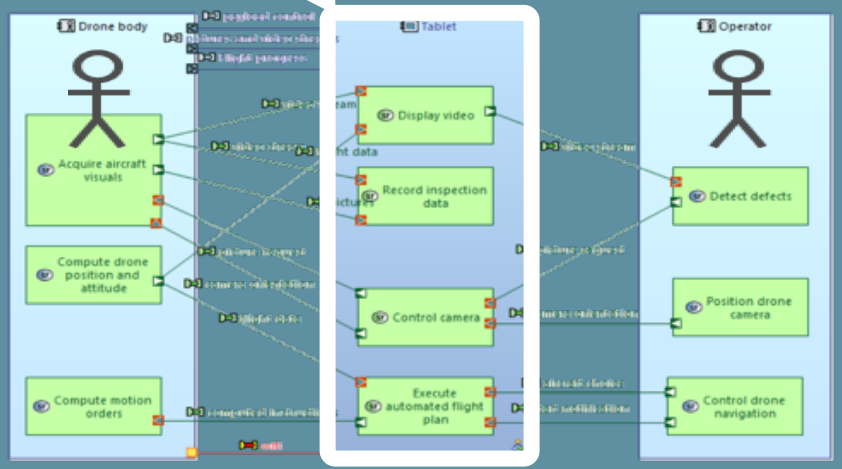
Textual requirements are created when needed, in addition to the model requirements: legal, non-functional, additional specification of internal expected behavior

Tablet is a constituent of a drone-based system



3

Tablet is the (sub)system of interest



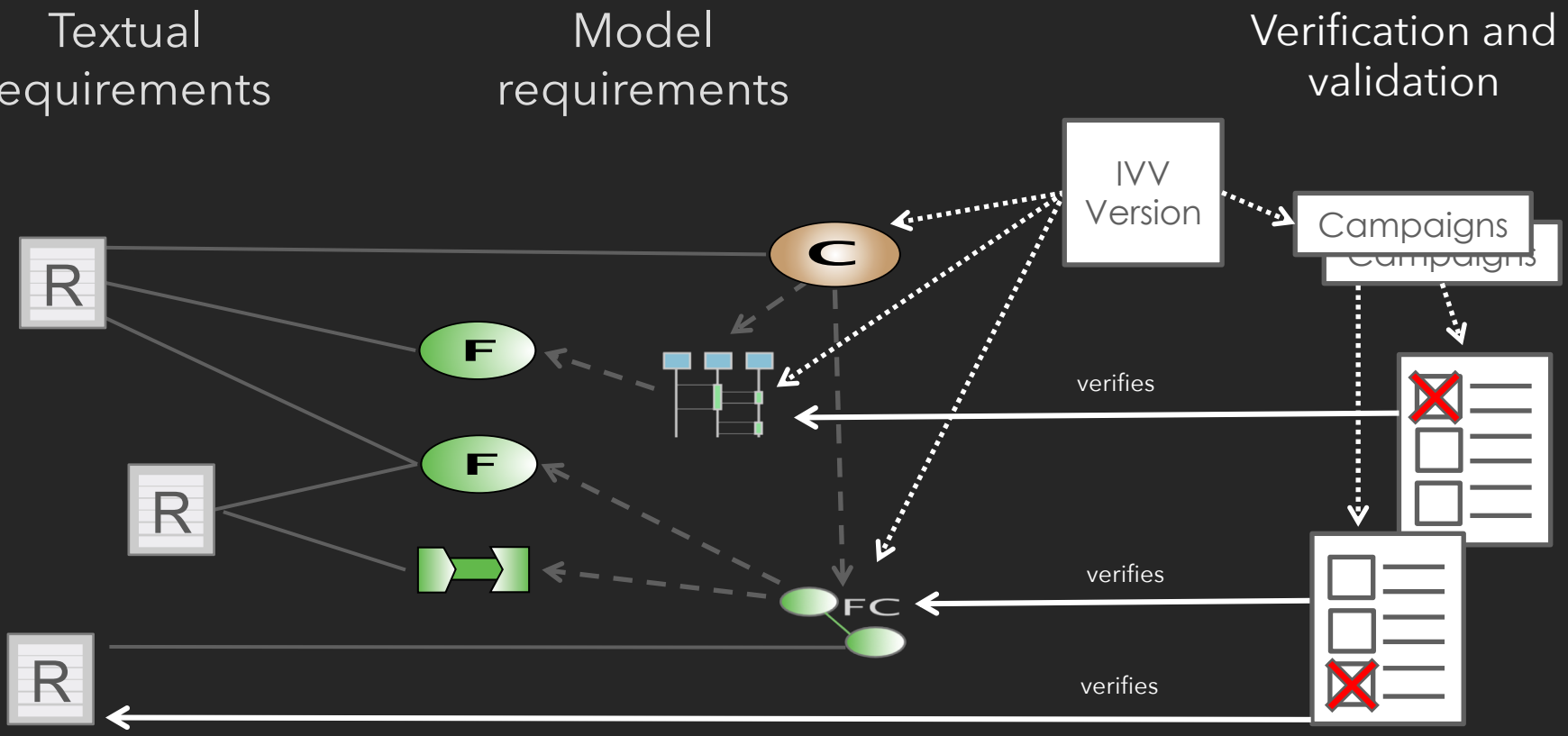
Model-based workflow favors co-engineering over the traditional differentiation between “customer” requirements and “system” requirements

Happy consequences

Contracts between engineering levels

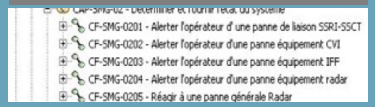
Verification and validation

Incremental (agile) development strategy

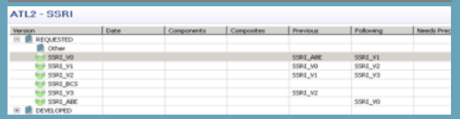

Textual requirements
Model requirements
Verification and validation
IVV Version
Campaigns
C
R
F
F
R
FC
R
verifies
verifies
verifies

2 years
30 persons
8 subsystems
9 engineering data packages

Functional Chains list

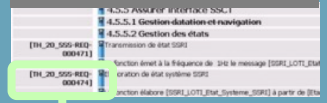


Functional Chains release definition

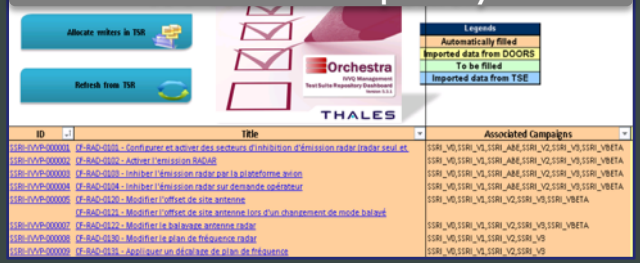


Functional chain

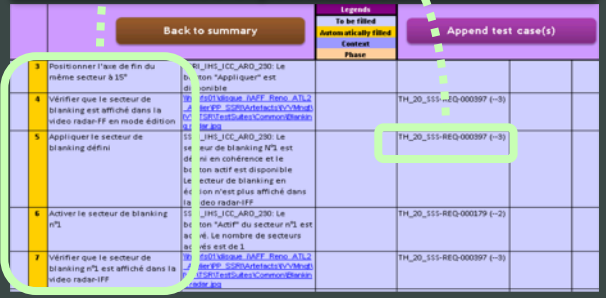
Requirements



IVV Test Suite Repository



IVV procedure

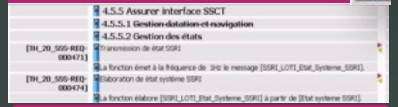


2 years
30 persons
8 subsystems
9 engineering data packages

Check of consistency with specification

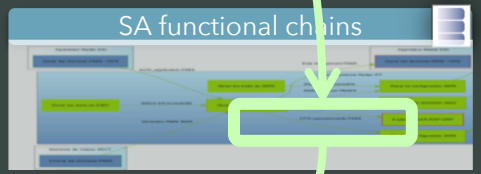
Test suites

Requirements



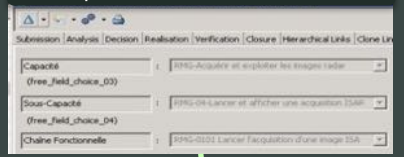
LA functions & exchanges

SA functional chains

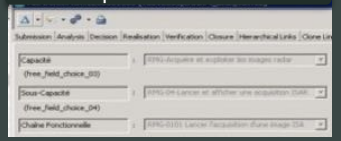


Impact analysis

Specification Defects



Component Defects



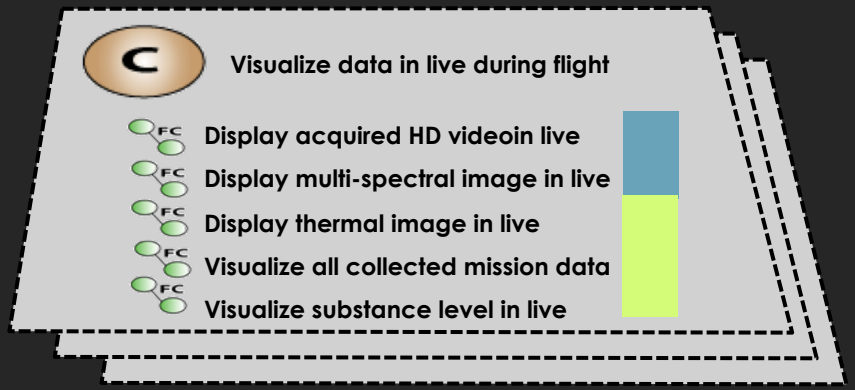
Component Evolution

Happy consequences

Contracts between engineering levels

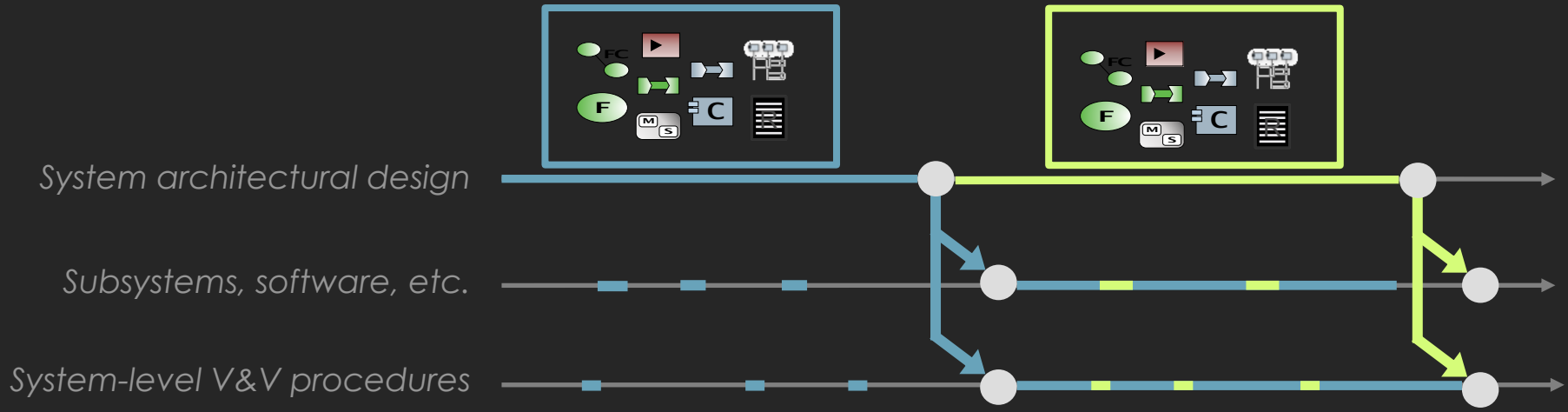
Verification and validation

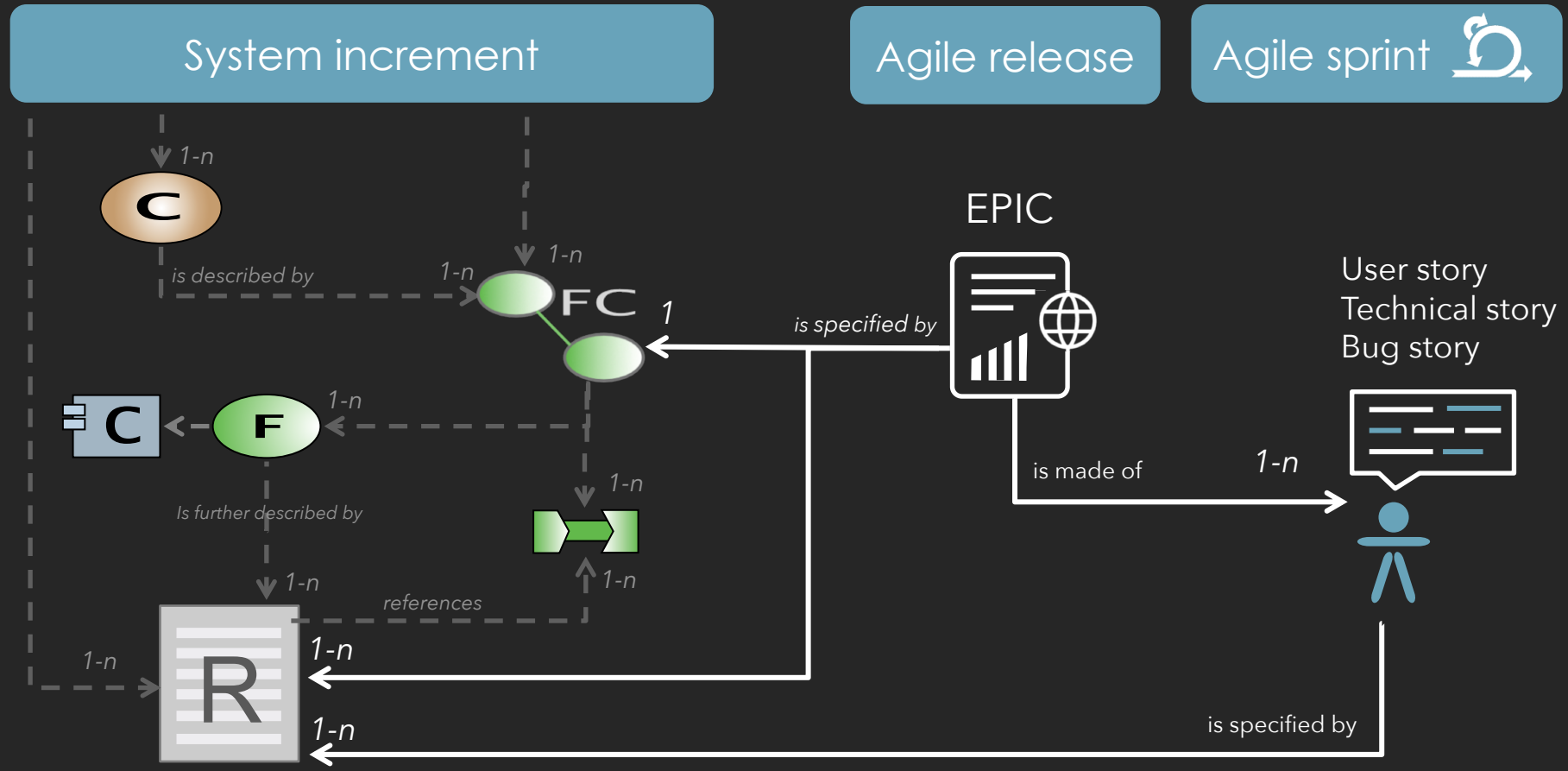
Incremental (agile) development strategy

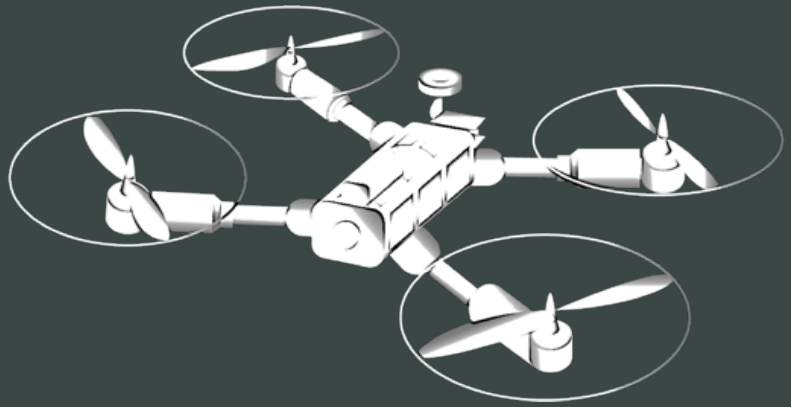


Definition of increments with expected functional chains

Vertical slices of architectural design across need and solution models




System increment
Agile release
Agile sprint
1-n
C
is described by
1-n
1-n
FC
1
is specified by
EPIC
User story
Technical story
Bug story
C
F
1-n
1-n
is made of
1-n
Is further described by
1-n
1-n
references
1-n
R
1-n
1-n
1-n
is specified by



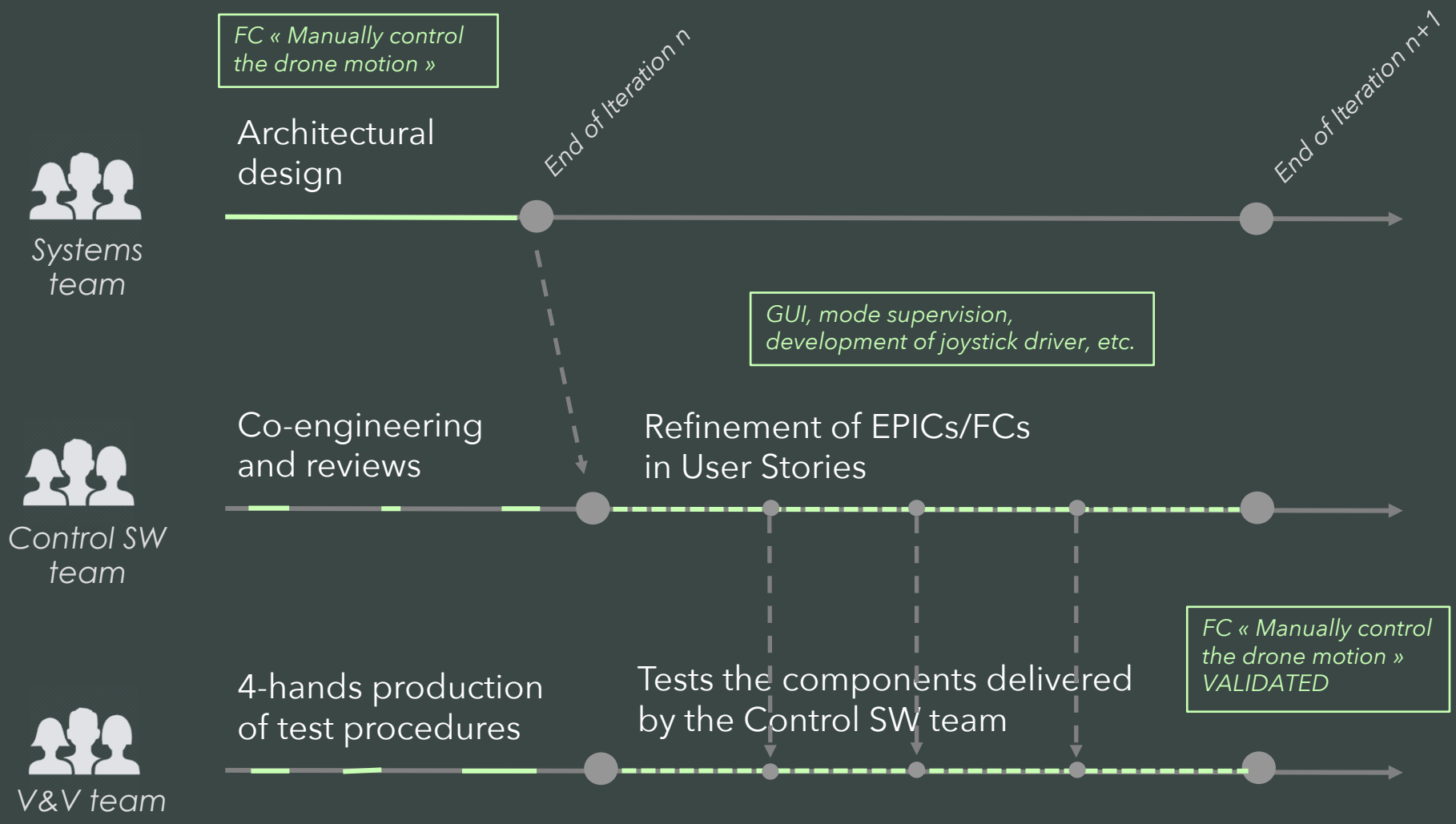
Systems team

V&V team

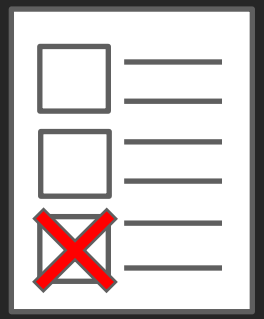
Software team

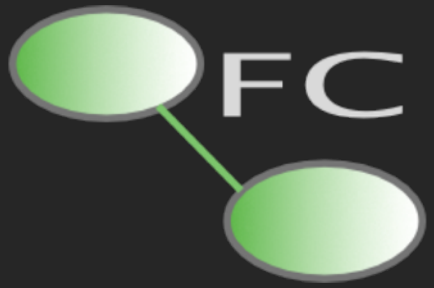
System iteration: 3 months
SW sprint: 3 weeks

Vision of the high-level capabilities of the product is known and shared, functional chains have been dispatched in several system expected increments


FC « Manually control the drone motion »
Architectural design
End of Iteration n
End of Iteration n+1
Systems team
GUI, mode supervision, development of joystick driver, etc.
Co-engineering and reviews
Refinement of EPICs/FCs in User Stories
Control SW team
FC « Manually control the drone motion » VALIDATED
4-hands production of test procedures
Tests the components delivered by the Control SW team
V&V team

Functional chains are the new backbone

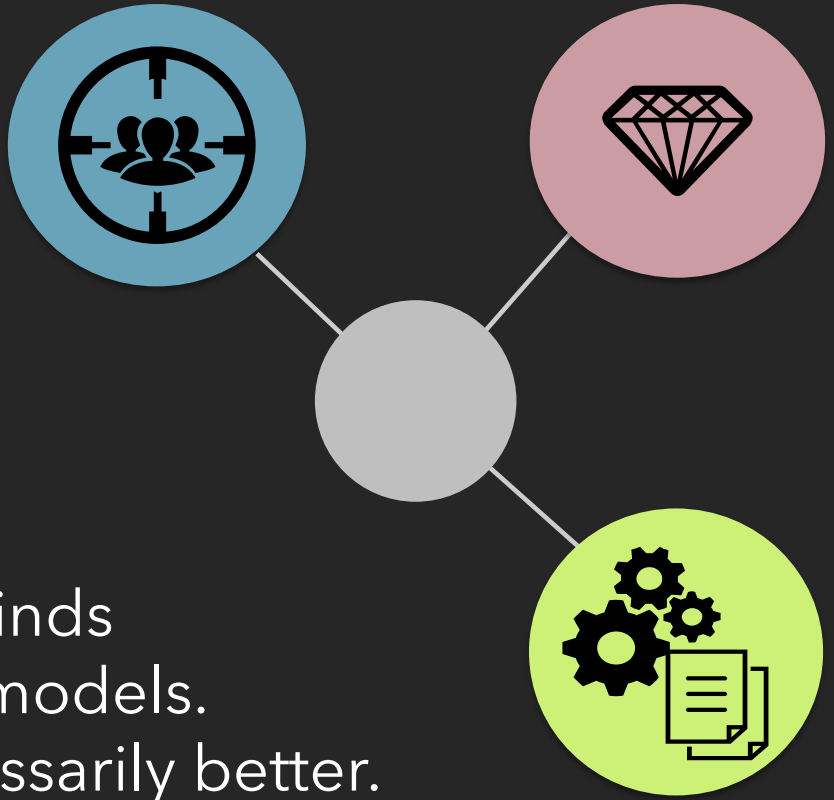




3. (MB) Engineering objectives

SHARE

Improve communication and reduce ambiguities



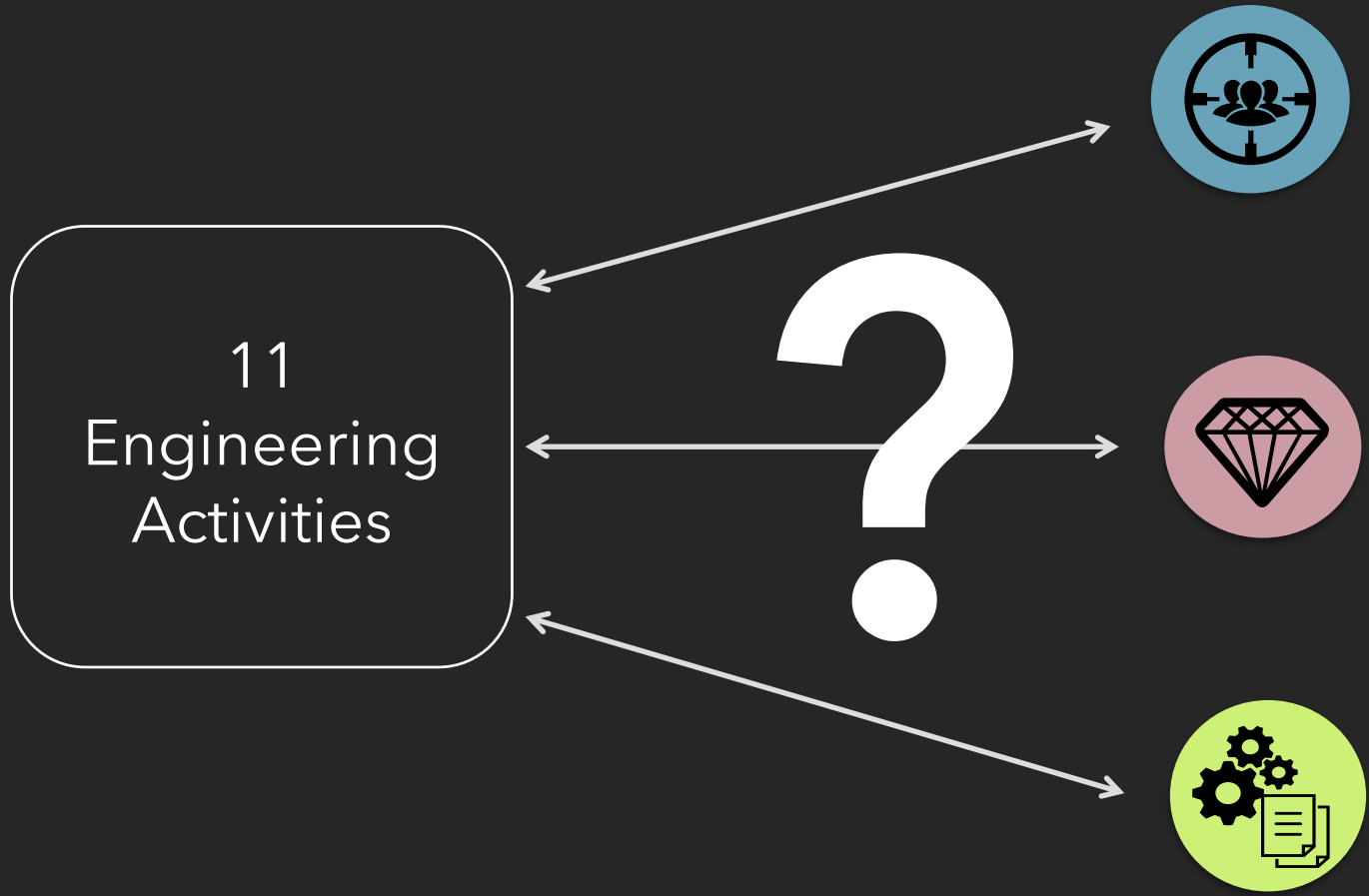
SECURE

Analyze and evaluate to master complexity, drive engineering activities

AUTOMATE

Generate documentation, code, models, etc.

Three different kinds
of purposes for models.
More is not necessarily better.


11
Engineering
Activities

	SHARE	SECURE	AUTOMATE
3. DESIGN THE ARCHITECTURE	Arcadia LA/PA models describe the architectural design: **functional expectations** from each solution constituent, interfaces, functional chains. They provide a **common understanding of the chosen solution**, are known by all stakeholders, and are used in documentation on ad hoc basis.	Arcadia LA/PA models describing the architecture are exploited to **guarantee the consistence and completeness of the design**. They are used to consider **alternatives** and support **early evaluation** (sizing assumptions, performance, cost, etc.). Model and textual requirements are rigorously articulated (derivation, justification, etc.).	**SSDD/ICD are (at least partially) generated** from the Arcadia LA/PA models. Engineering data is extracted from the model to feed engineering specialties models and vice versa. Early design validation is performed with **simulation techniques**.
4. MANAGE VARIABILITY AND REUSE	Models of reusable assets and of product architecture exist and are known. They are mainly **used for documentation** purpose. Feature models describe the product variability and standard configurations	**Models of reusable assets are rigorously governed** and managed in configuration, they are assembled in solution architecture models. **Architecture models are mapped to feature models,** they help **capture, strengthen, and optimize** product variability and configurations.	Project models are **automatically initialized or derived** from feature models and architecture models

Orientation and assessment

10 000 feet

1 000 feet

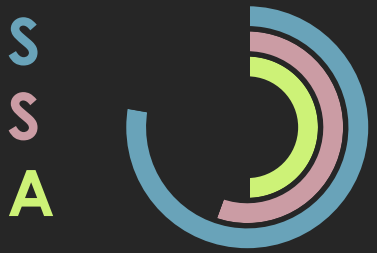
10 feet

Orientation and assessment

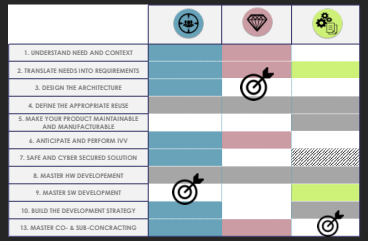
1. Analyze context, identify relevant activities

2. Answer orientation/assessment questions (10 000/1000/10)

3. Get a quick and visual overview/progress status of the MBSE footprint in engineering practices



4. Define concrete improvement actions: how can MBSE help solved specific engineering problems



4. Organizational aspects of deployment

Core enablers

Constant and renewed commitment from the management

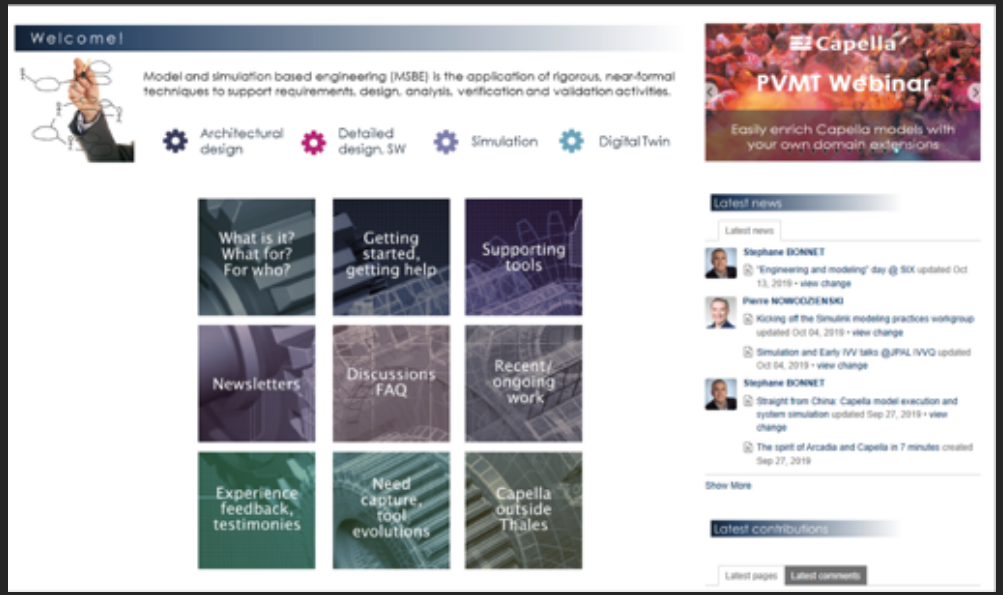
Strong motivation and resilience of a network of highly skilled individuals who work together on a common goal

Sizable mentoring/coaching force

Mantras

1. Delivering ≠ Being competitive
2. Visio diagrams are not enough: You need more rigor
3. Don’t seek the big bang, focus on specifics
4. Manage/monitor your modeling activities
5. Get help

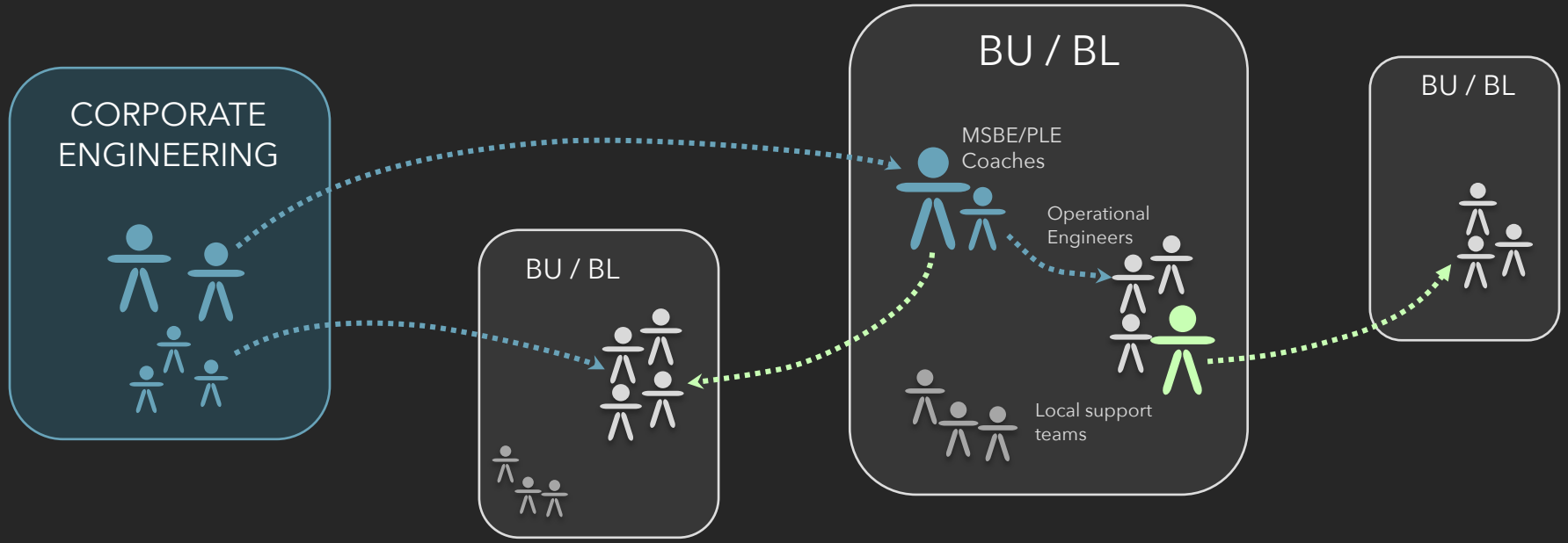
MSBE Community



Capella Users Days

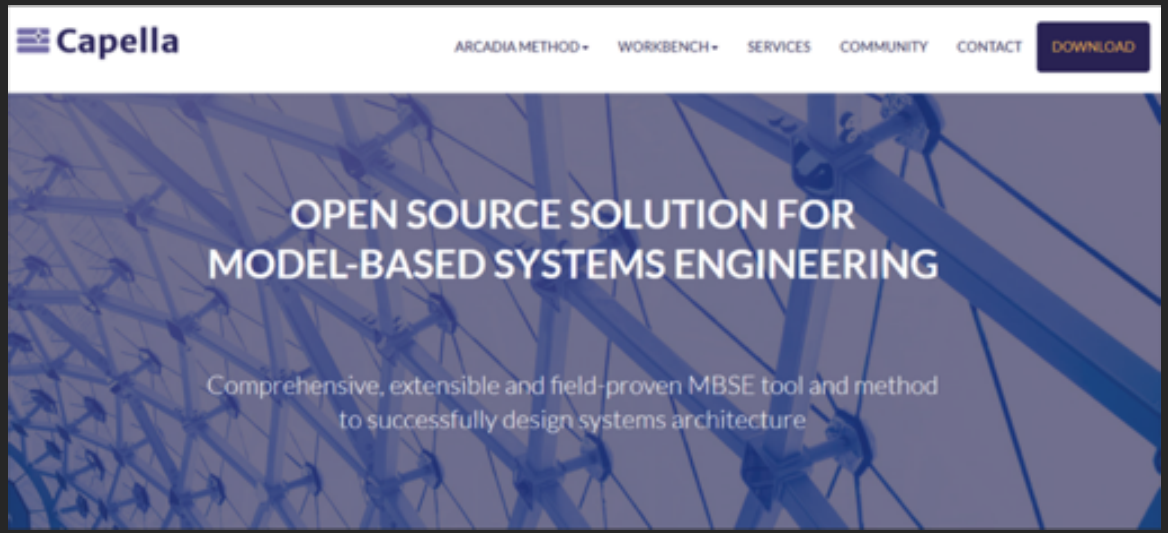
MSBE Services

Network of coaches



Resources

https://eclipse.org/capella



Learning material

Industrial case studies

Public forum

Youtube channel (webinars)

Free download

Thank you! Questions?

Capella website:
http://www.polarsys.org/capella/
LinkedIn
https://www.linkedin.com/in/capella-mbse-tool
https://www.linkedin.com/groups/8605600
Twitter
https://twitter.com/capella_arcadia