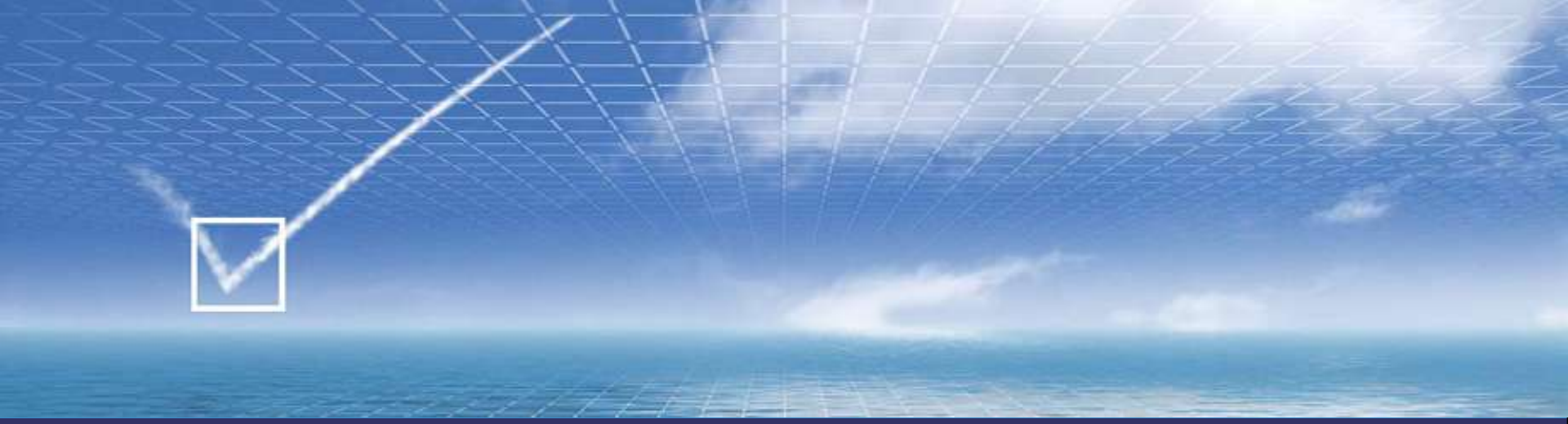


A Model-Based Engineering Method for System, Software and Hardware Architectural Design

ARCADIA

A method best-supported by

Capella

jean-luc.voirin@fr.thalesgroup.com
stephane.bonnet@thalesgroup.com
daniel.exertier@thalesgroup.com

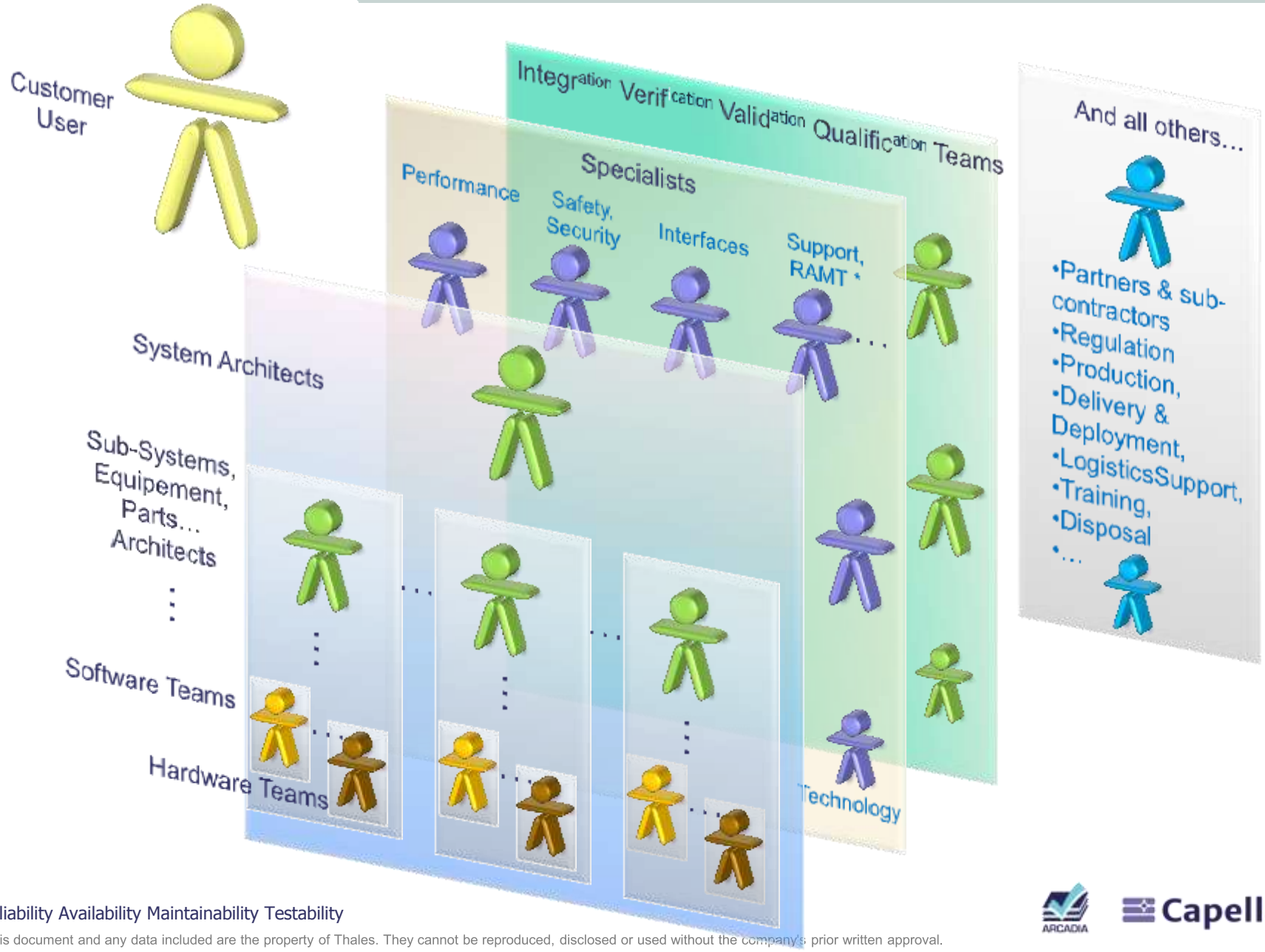
January 15th, 2015

ARCADIA Capella

Current Engineering Practices and Gaps

Engineering practices and their limits

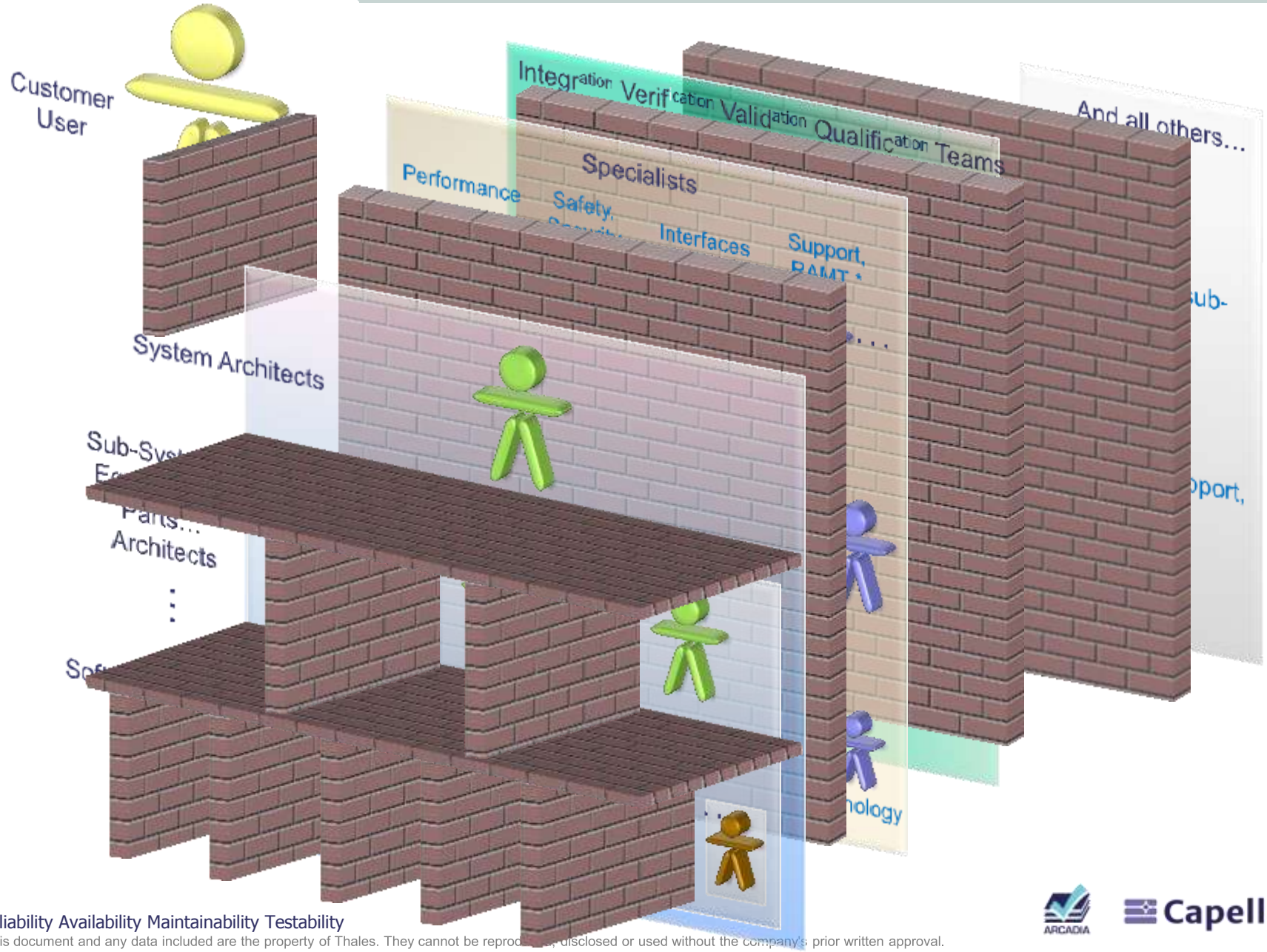
The System Engineering « Ecosystem » Challenge: Collaboration



* RAMT: Reliability Availability Maintainability Testability

ARCADIA Capella

The System Engineering « Ecosystem » Challenge: Collaboration



* RAMT: Reliability Availability Maintainability Testability

ARCADIA Capella

Seven Deadly Sins* in System Engineering

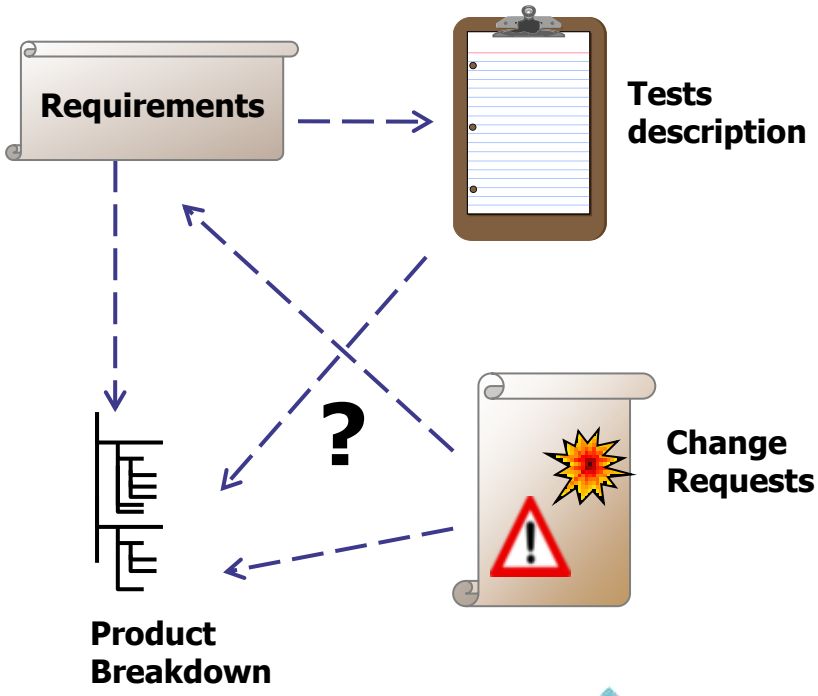
- "Full Requirements driven approach weaknesses"
- Bad understanding of CONOPS/CONUSE**
- Incoherent reference & decisions between engineering specialties
- Poor continuity between engineering levels
- Late discovery of problems in definition & architecture
- Underestimated Architectural Design impact / benefit
- No anticipation of IV&V, no functional mastering
- No justification, no capitalisation for reuse/product line

* Deadly Sins: Péchés capitaux
** CONcepts of OPerationS, CONcepts of USE
*** IV&V : integration, verification, validation

Limits of Textual Requirements-Driven Engineering

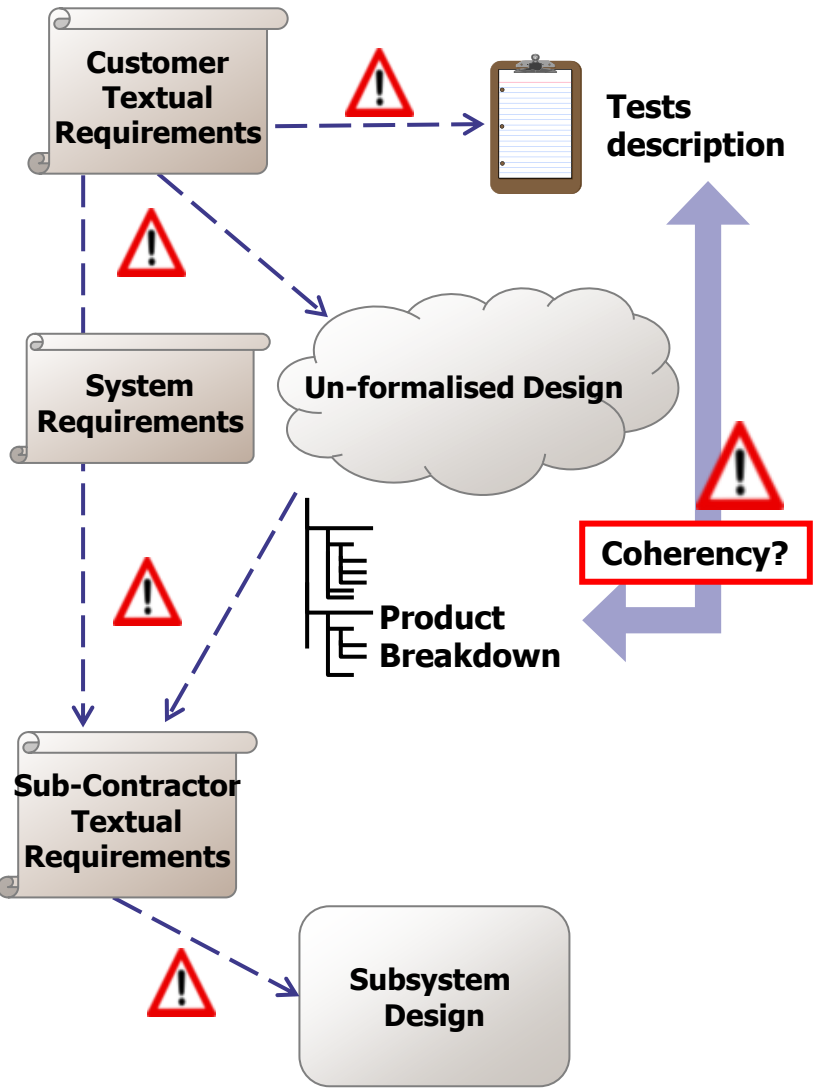
- Textual requirements are today the main vector of technical management contract with the customer

- However they have significant limitations:
  - Informally described and not adapted to validation by formal methods
  - Inadequate to support Design
  - Unable to describe a solution
  - Traceability links Creation process unclear and hard to formalize
  - Traceability links unverifiable
  - Difficulties to securely reuse requirements alone
  - …

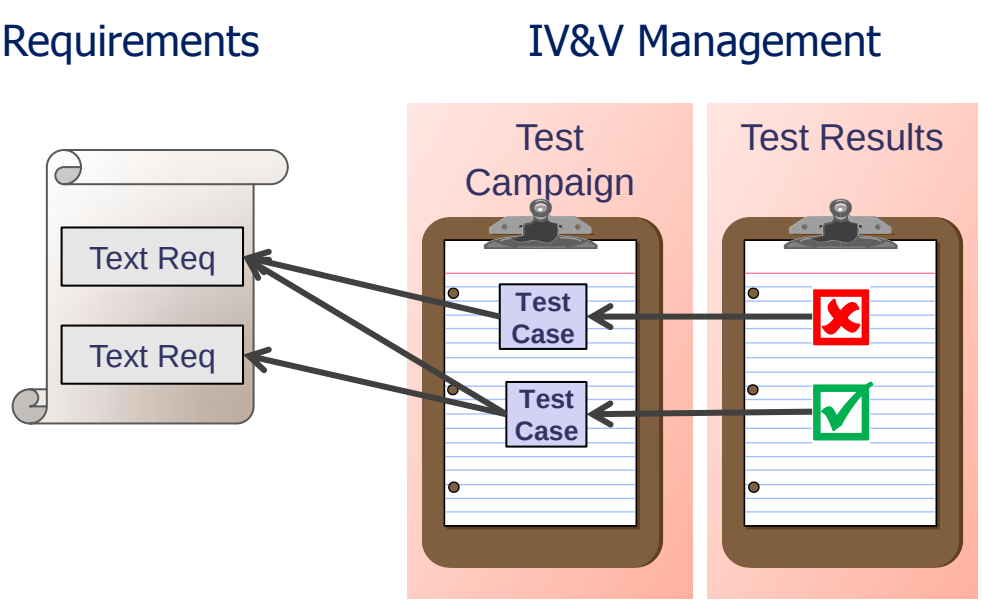


Requirement-Based Approach

Definition and subsystem specification



Requirement validation process



IV&V : integration, verification, validation

Some Consequences on Engineering

- No solution is described, only requirements allocation
- Definition of suppliers delivery is weak and not sufficient
- Checking quality of the definition is not possible before IV&V
- Thus, justification of definition is poor and unreliable
  - Components functional contents, behaviour, interface definition…
- Examples of consequences in IV&V :
  - Poor control of versioning and complexity
  - Missing components when verifying a requirement
  - No mastering of the consequences of non-maturities (PCR ...)
  - Poor mastering of behaviour(startup, non nominal states ...)
  - Difficulty in organizing / optimizing regression tests
  - Difficulty of locating faults and impact analysis ...
- These problems increase along with system or project complexity

IVV : integration, verification, validation

ARCADIA Goals and Action Means

Solving the walls issue

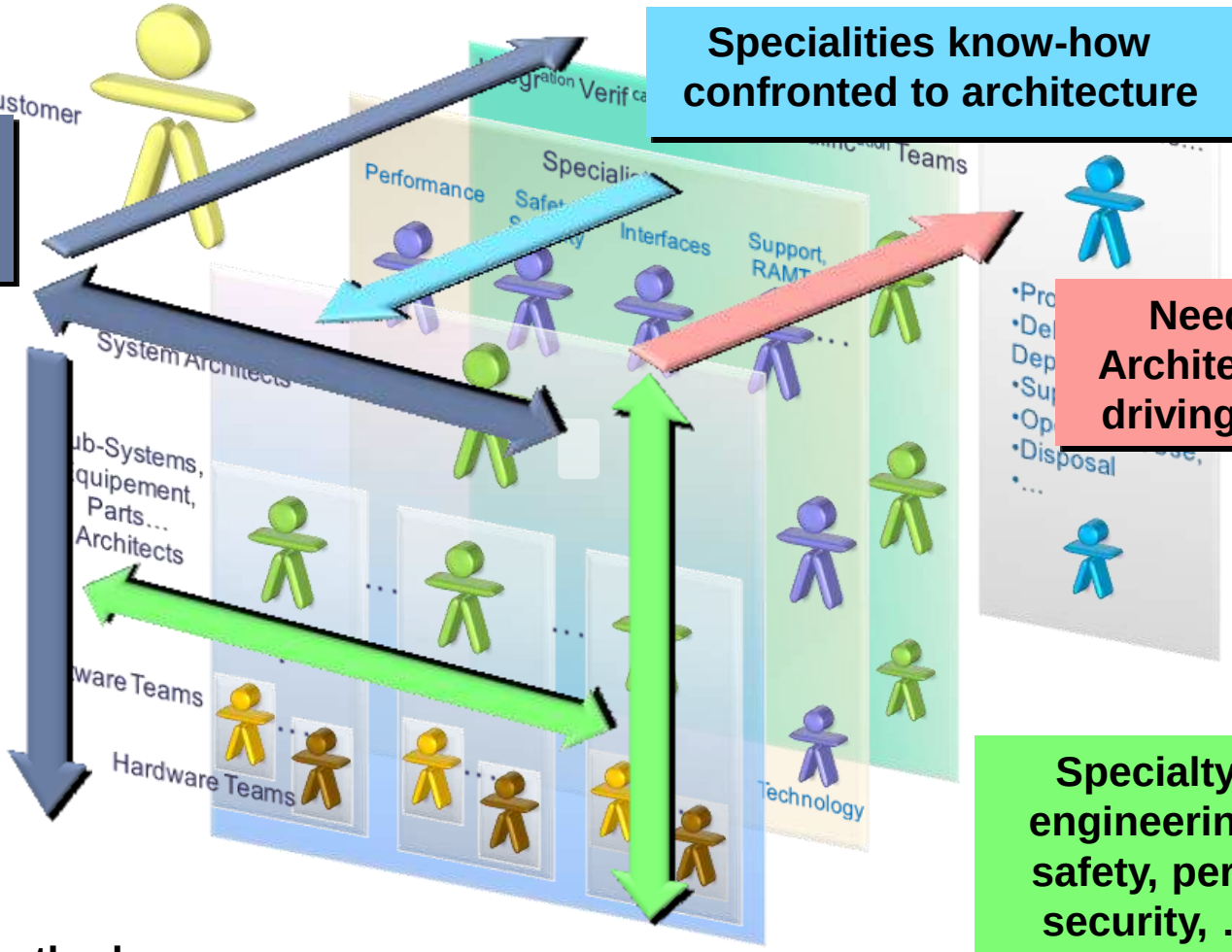
What ARCADIA Is

A tooled method devoted to systems, software and hardware Architecture Engineering

- To understand the real **customer need**,
- To define and **share** the system **architecture** among stakeholders,
- To **early validate** system design and **justify** it,
- To ease and **master IVVQ** (Integration, Validation, Verification, Qualification).

➔ Improve efficiency and quality of System Engineering
➔ Master complexity of products
➔ Foster and secure collaborative work of engineering stakeholders
➔ Reduce development Costs & Schedule

ARCADIA Action Means – Engineering Collaboration



One Need Definition for all

Specialities know-how confronted to architecture

Need & Architecture driving IVVQ

Specialty engineering: safety, perf, security, …

One global method, adaptable/adapted to each domain

ARCADIA Capella

Seven Cardinal Virtues in System Engineering

☹ *Bad understanding of operational use*

☹ *Incoherent reference & decisions between engineering specialties*

☹ *Poor continuity between engineering levels*

☹ *Late discovery of definition & architecture problems*

☹ *Underestimated Architecture impact / benefit*

☹ *No anticipation of IVVQ, non functional mastering*

☹ *No justification, no capitalisation for reuse/product line*

Analyse and formalise **stakeholders need**:
operational scenarios & processes, functional & non-functional need

Drive specialties through **a unique, shared reference**;
coordinate and evaluate global impact of decisions on it

Share **reference** between **engineering levels**;
secure its application for impact analysis

Early check **the technical solution**
against Oper/Funct/NF need as well as against engineering constraints

Use **architecture to strengthen engineering**
according to engineering stakes

Manage **IV&V** using the **common functional reference** to schedule, define and master it

Capitalise by formalising **definition and design**, including decisions & justifications

JLV

ARCADIA Capella

What Arcadia Is and Is Not

ARCADIA is NOT…

☹ *An academic work or a tool-vendor offer; Tested on toy problems*

☹ *Designed for traditional top-down, waterfall or « V » lifecycle*

☹ *Only for large / complex projects*

☹ *Only for projects starting from scratch*

☹ *Only for software dominant, or large system engineering, or…*

☹ *Costly and complex to set and use*

☹ *Restricted to system definition phase*

☹ *Too un-mature to be used yet*

ARCADIA is…

A set of practices **specified and tested by real projects engineers** wanting to address their top priority needs

Designed for adaptation to most processes and lifecycle constraints : bottom-up, legacy reuse, incremental, iterative, partial…

Used for bids and partial problems, down-scalable

Dealing with reuse, reverse engineering, evolution mastering, hot spots addressing

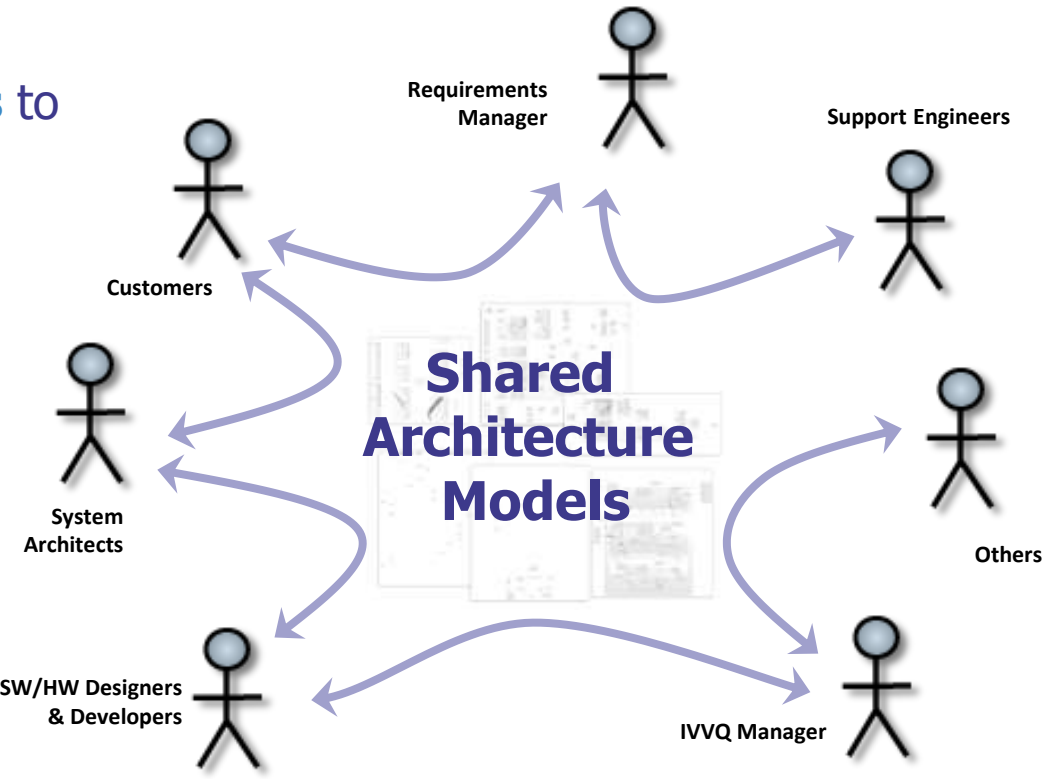
Used for thermal and power as well as information systems or software… architectures

Adjustable: **Focus on your major problems first and you will get ROI**

Also addressing & easing IV&V (integration verification validation)
Tested in tens of operational contexts

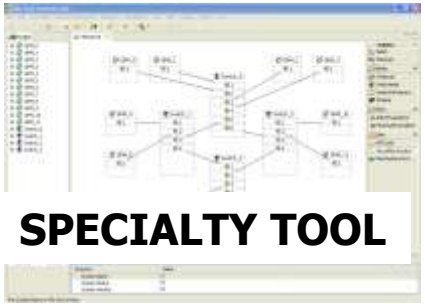
Needs to Support Collaborative Architecture Building

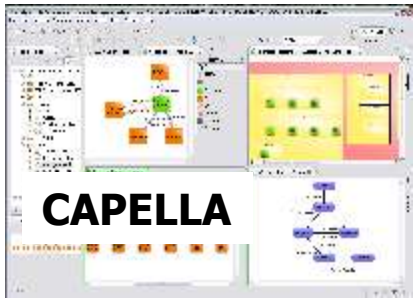
1. Common unified concepts to structure engineering :
Product-centric description & capitalisation



2. Collaborative workflow
Based on **shared description and unique reference**

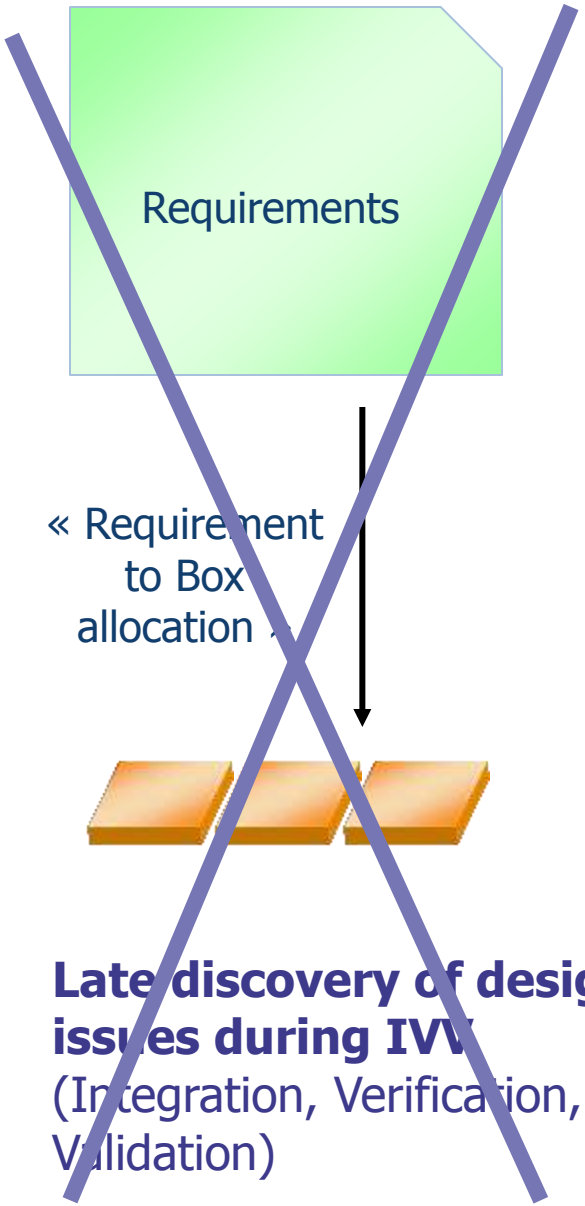
3. Architecture Analysis Capability & Tools



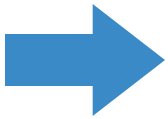


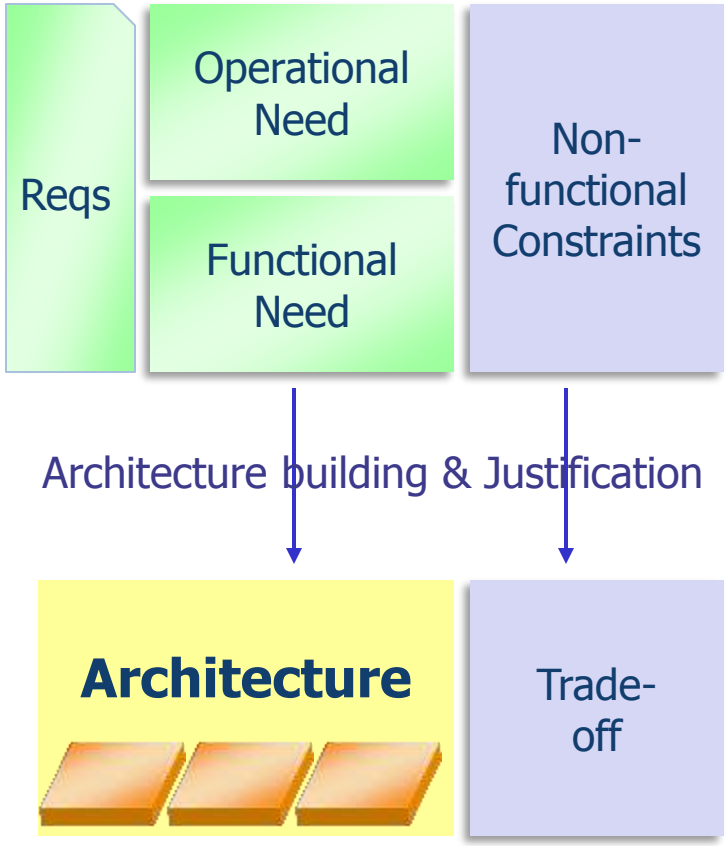
ARCADIA Capella

From Requirements Allocation to Architecture Mastering



Late discovery of design issues during IVV
(Integration, Verification, Validation)



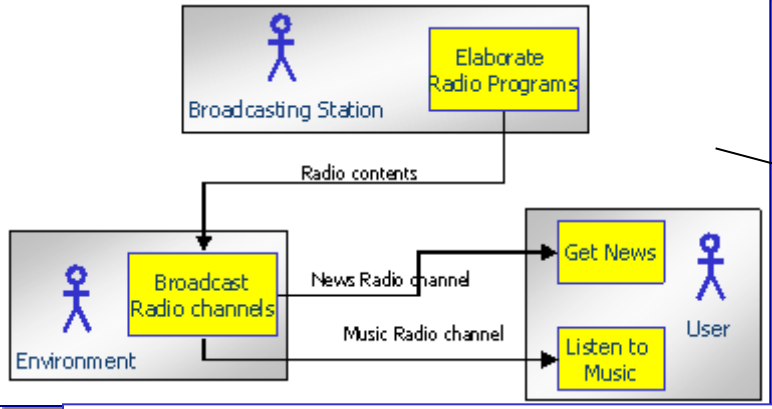


Early validation of the architecture
Mastering and optimisation of product and IV&V

ARCADIA Concepts

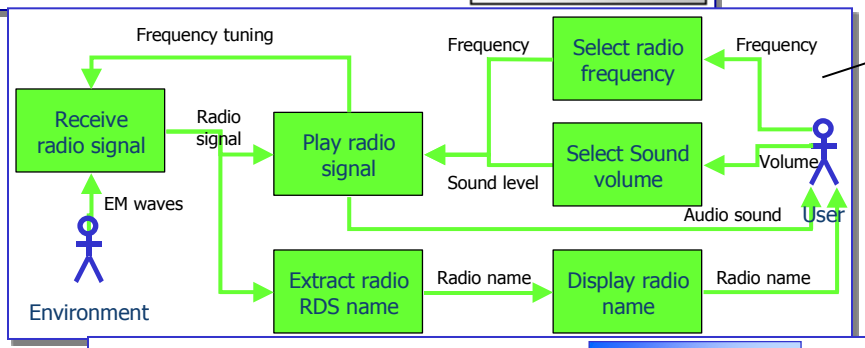
Examples

ARCADIA Concepts: Radio Set Example

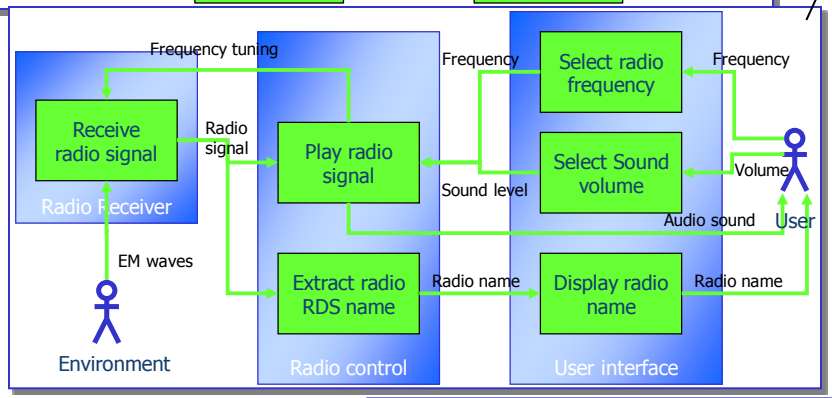


1. What **the users** of the system need to accomplish

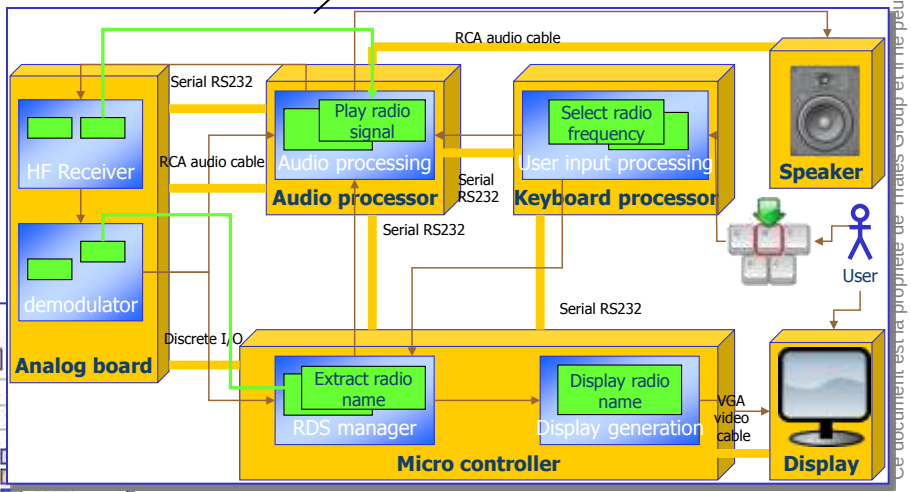
2. What **the system** has to accomplish for the users



3. How the system **will work** so as to fulfil expectations



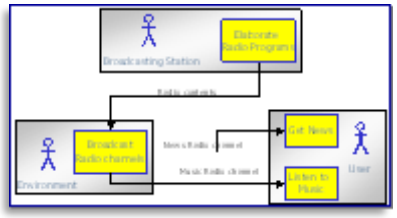
4. How the system will be **developed & built**

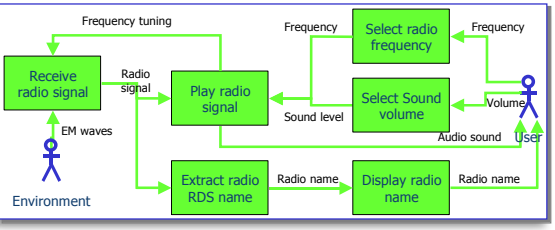


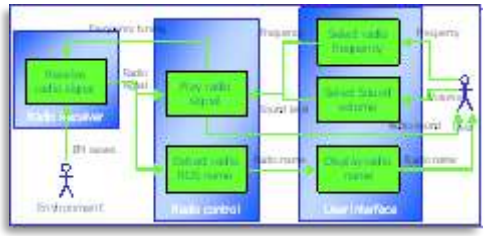
5. **What is expected** from each designer / sub-contractor

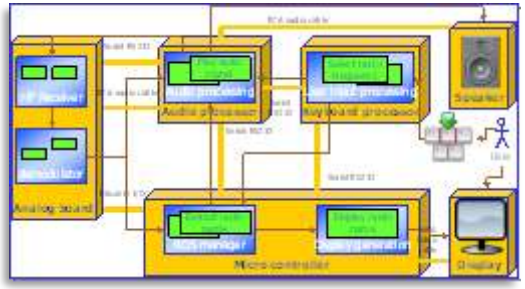
ARCADIA Capella

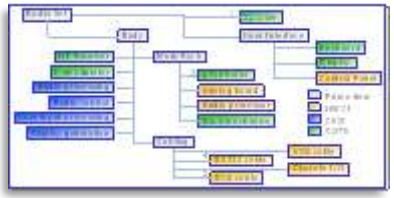
Method Steps

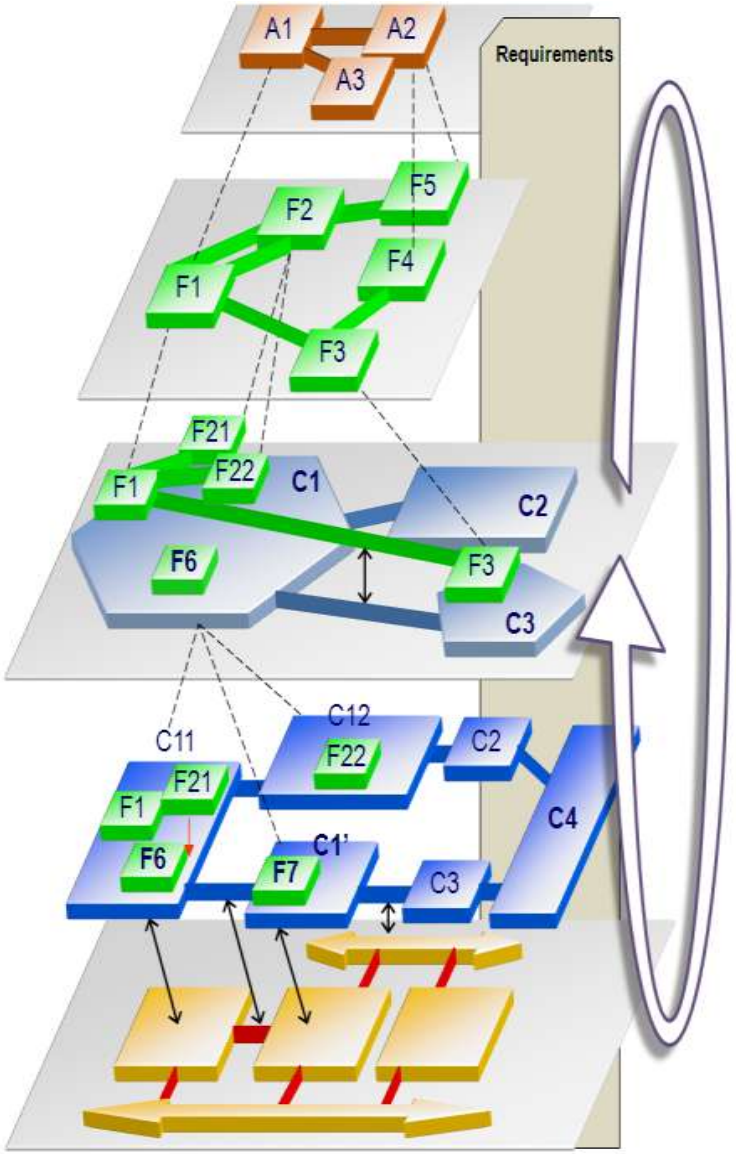












EPBS – End Product Breakdown Structure

User Need

System Need

Notional Solution

Final Solution

Sub-contractors input

ARCADIA Capella

An Example of Modelling & Early Validation: In-Flight Entertainment System

Playing videos on demand
Listening to music
Surfing the web
Gaming…

ARCADIA Capella

ARCADIA Concepts

Operational Analysis

Operational Analysis - Overview

- Focuses on analysing the **needs and goals, expected missions & activities**
- Is expected to ensure **good adequacy of System definition with regards to its real operational use** – and define IVVQ conditions
- Outputs : Operational Needs Analysis
  - Needs, in terms of actors/users,
  - Operational capabilities and activities,
  - Operational use scenarios (dimensioning parameters, operational constraints including safety, security, system life cycle….)

Operational Analysis: Main Concepts (1/2)

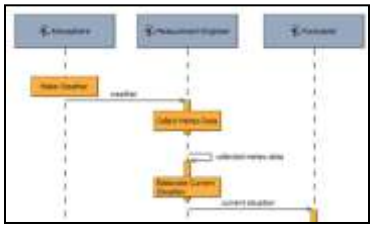
- Operational Capability
  - A capability is the ability of an organisation to provide a service that supports the achievement of high-level operational goals.

- Operational Activity
  - Process step or function performed toward achieving some objective, by entities that could necessitate to use the future system for this
  - e.g. Control traffic, go along a place, detect a threat

- Operational Entity
  - An operational Entity is a real world entity (other system, device, group or organisation…), interacting with the system (or software, equipment, hardware…) under study, or with its users

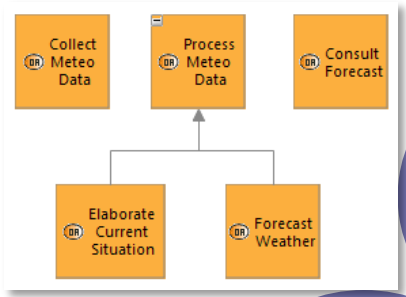
- Operational Actor
  - An actor is a [usually human] non decomposable operational Entity

Operational Analysis: Main Concepts (2/2)



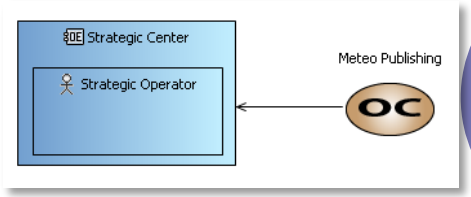
- Operational Interaction
  - Set of Operational services invocations or flows exchanged between Operational Activities, (e.g. Operational Interactions can be composed of Operational data, events...).
- Operational Process
  - A logical organization of Interactions and Activities to fulfil an Operational Capability
- Operational Scenario
  - A scenario describes the behaviour of a given Operational Capability

Operational Analysis Workflow and Main Diagrams



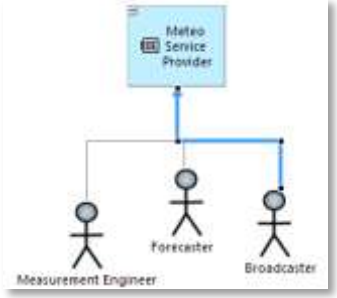
Define the Operational Activities

Define the Interactions between Operational Activities



Define Operational Capabilities

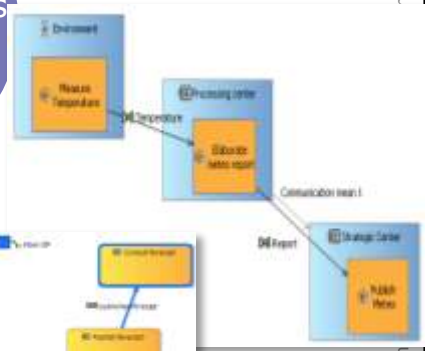
Define the Operational Entities

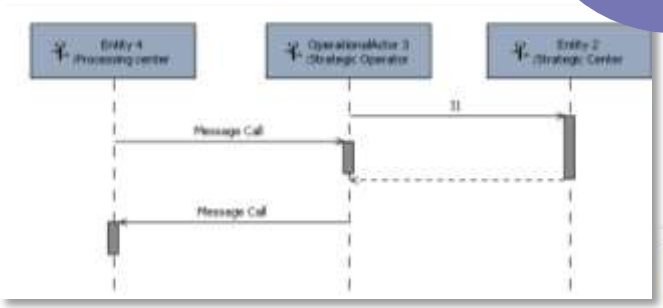


Operational Analysis

Define Operational Entity Scenarios

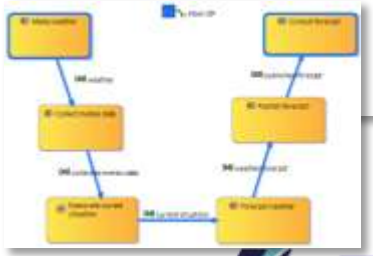
Allocate Operational Activities to Operational Entities

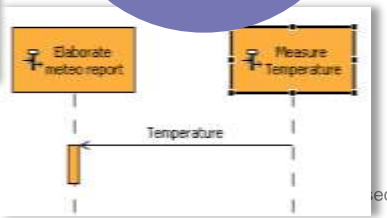




Define Operational Activity Scenarios

Define Operational Processes





ARCADIA Capella

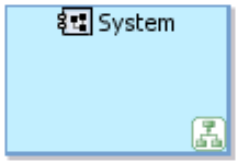
ARCADIA Concepts

System Need Analysis

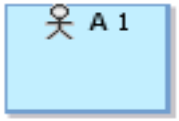
System Analysis - Overview

- Define **how the system can satisfy the former operational needs**:
  - System functions to be supported & related exchanges
  - Non functional constraints (safety, security…)
  - Performances allocated to system functional chains
  - Role sharing and interactions between system and operators
- **Checks for feasibility** (including cost, schedule and technology readiness) of customer requirements
- Outputs: System Functional Analysis description
  - Interoperability and interaction with the users and external systems (functions, exchanges plus non-functional constraints), and system requirements

System Analysis: Main Concepts (1/2)



- System
  - An organized set of elements functioning as a unit.
  - An aggregation of end products and enabling products to achieve a given purpose.



- System Actor
  - External actor interacting with the system via its interfaces

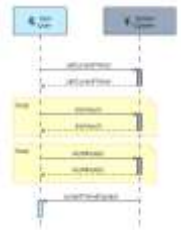
- System Mission
  - A mission describes a major functionality of the system from a very high level point of view. It is a reason why the system is developed.
  - high-level operational goal

- System Capability
  - A capability is the ability of a system to provide a service that supports the achievement of high-level operational goals

- System Function
  - Function at System level
  - A function is an action, an operation or a service fulfilled by the system or by an actor when interacting with the system
  - e.g. 'measure the altitude', 'provide the position'

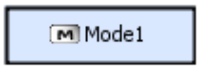
System Analysis: Main Concepts (2/2)

- Exchange and Port
  - An Exchange is an interaction between some entities such as actors, the system, functions or components, which is likely to influence their behaviour.
  - e.g. tuning frequency, radio selection command…
  - The connection point of an exchange on an entity is called a port.
- Functional Exchange
  - Piece of interaction between functions that is composed of data, events, signals, etc. A Flow Port is an interaction point between a Function and its environment that supports Exchanges with other ports



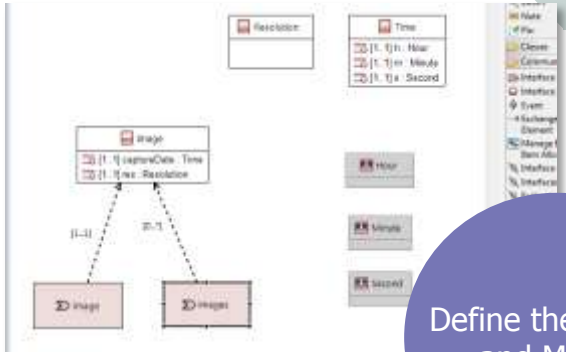
- Scenario
  - A scenario describes the behaviour of the system in a given Capability.
  - Scenarios permit to specify the dynamical behaviour of the system by showing interaction sequences performed by the actors and by the system

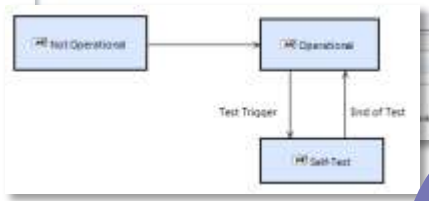
- State
  - a physical and operational environment condition

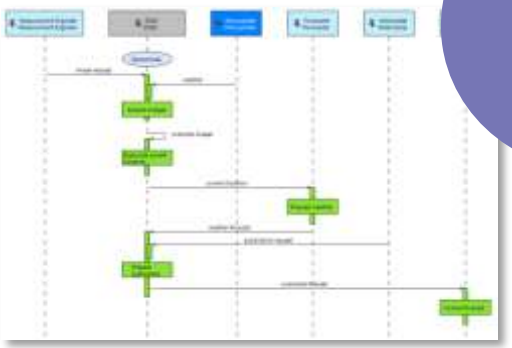


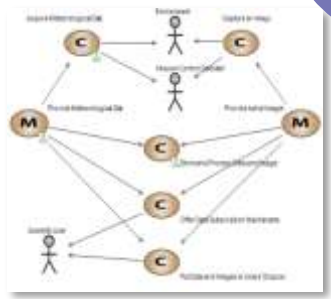
- Mode
  - a type of operation in a given state of the system, or the performance level within a state

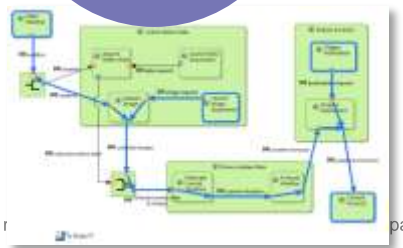
System Analysis Workflow and Main Diagrams

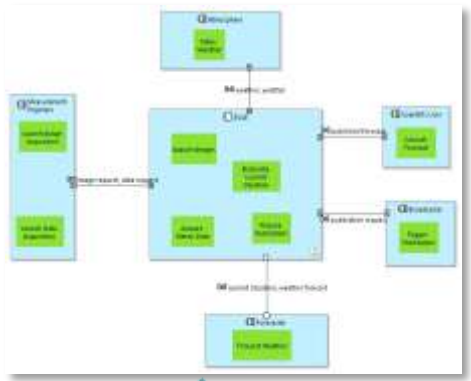


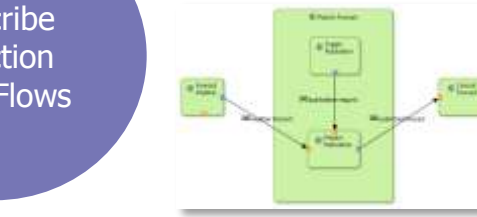


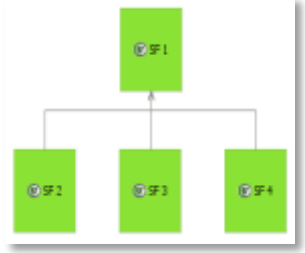












Define the Data Model

Transition from Operational Analysis

Define the States and Modes

Refine System Functions

System Analysis

Define Scenarios

Describe Function Data Flows

Define System Capabilities

Define Functional Chains

Assign Functions To System and Actors

ARCADIA Capella

ARCADIA Concepts

Logical Architecture

Logical Architecture - Overview

- Intends to identify the **system components, their contents, relationships and properties, excluding implementation or technical/technological issues**. This constitutes the system logical architecture.

- All major [non-functional] constraints (safety, security, performance, IVV…) are taken into account so as to find the **best compromise** between them.

- Outputs : selected logical architecture:
  - Components & interfaces definition, including formalisation of all viewpoints and the way they are taken into account in the components design.
  - Links with requirements and operational scenarios are also produced

Logical Architecture Design: Managed Entities

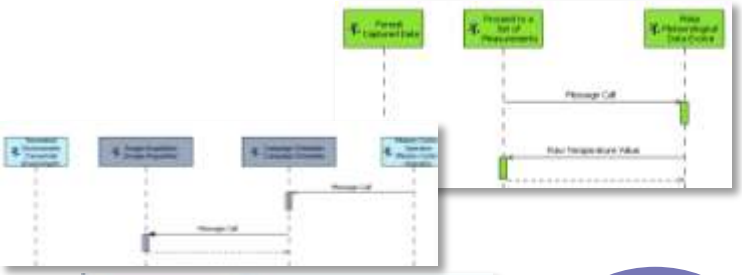
- Logical Function
  - Function applied at Logical level

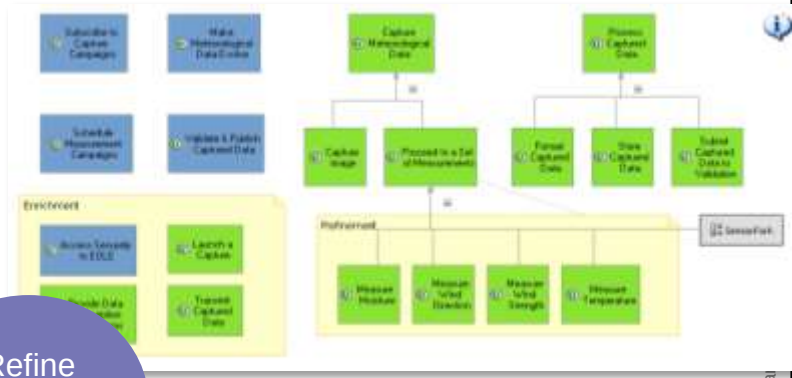
- Logical Component
  - Logical Components are the artefacts enabling a notional decomposition of the system as a "white box", independently from any technological solutions, but dealing with major system decomposition constraints
  - Logical components are identified according to logical abstractions (i.e. functional grouping, logical interfaces)

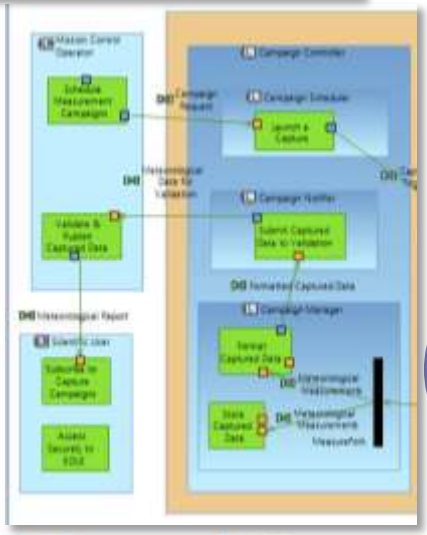
- Functional Exchange
  - Piece of interaction between functions that is composed of data, events, signals, etc. A Flow Port is an interaction point between a Function and its environment that supports Exchanges with other ports

- Component Exchange
  - Represent the interactions between Logical Components

Logical Architecture Workflow and Main Diagrams







Transition from System Analysis

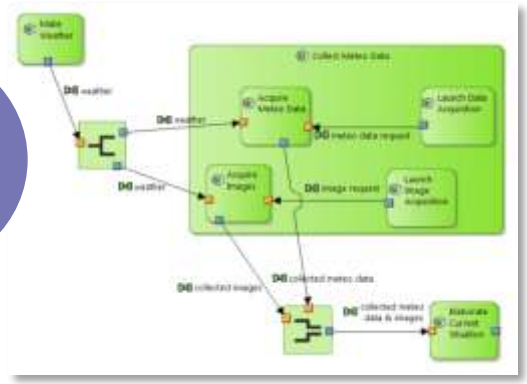
Describe Data Flow Scenarios

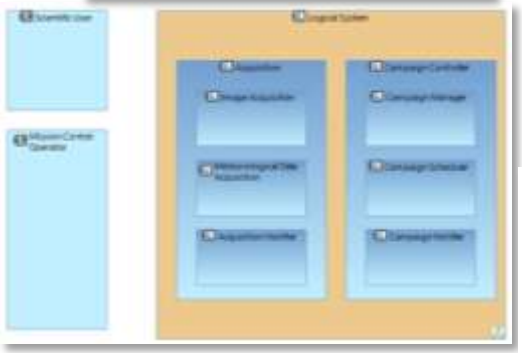
Refine Logical Functions

Logical Architecture

Assign Functions To Components

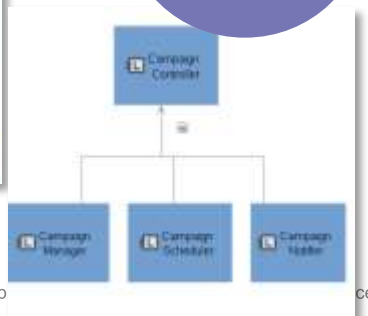
Describe Logical Data Flows

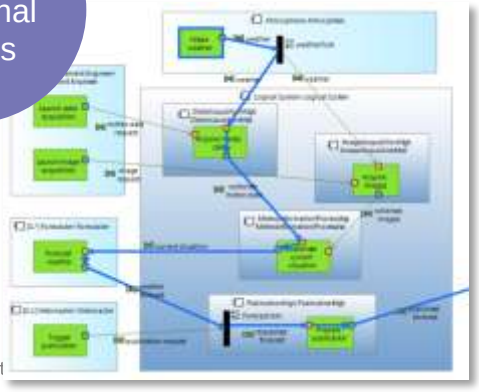




Define Logical Components

Refine Functional Chains





ARCADIA

Capella

ARCADIA Concepts

Physical Architecture

Physical Architecture - Overview

- Intends to identify the system components, their contents, relationships and properties, including implementation or technical/technological issues
- **Introduces rationalisation, architectural patterns, new technical services and components**
- Makes the logical architecture evolve according to implementation, technical & technological constraints & choices (at this level of engineering)
- The same 'Viewpoints-driven' method as for logical architecture design is used for physical architecture definition
- Outputs : selected Physical Architecture:
  - Physical Components, including formalisation of all viewpoints and the way they are taken into account in the components design
  - Links with requirements and operational scenarios

Physical Architecture Design: Main Concepts (1/2)

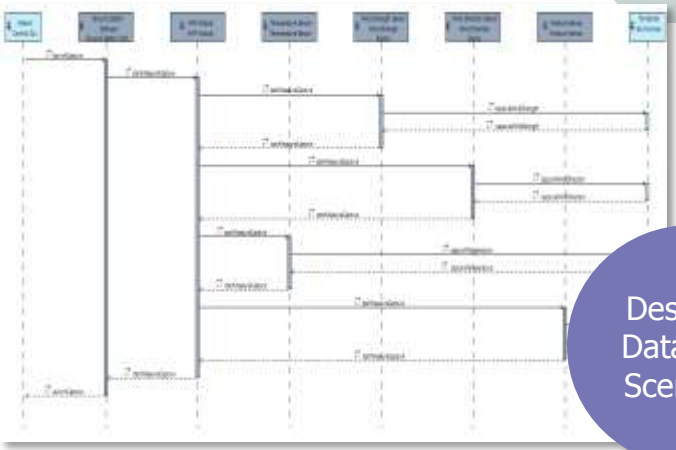
- Physical Function
  - Function applied at physical level
- Physical Component
  - Physical Components are the artefacts enabling to describe physical decomposition of the system to satisfy the logical architecture identified at the upper abstraction level. Physical components are identified according to physical rationale (i.e. components reuse, available COTS, non functional constraints...).
  - Two natures of components :
    - Behaviour
      - physical component in charge of implementing / realising part of the functions allocated to the system
      - e.g. operational software, radar antenna, …
    - Node or implementation
      - material physical component, resource embedding some behavioural components, and necessary to their expected behaviour
      - e.g. motherboard, units of memory, middleware's and operating systems …

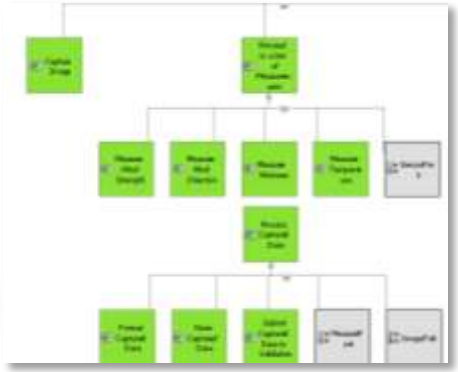
Physical Architecture Design: Main Concepts (2/2)

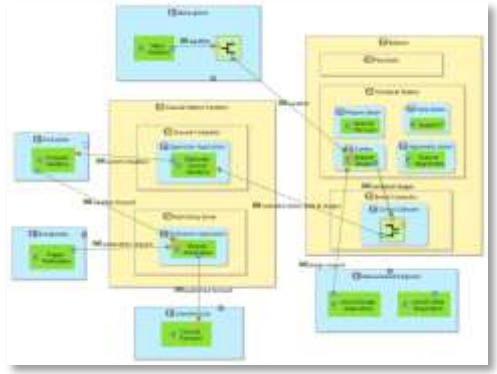
- Component Exchanges are meant to be used between Behaviour Components.
  - They are identical to the Component Exchanges of the System Analysis and the Logical Architecture

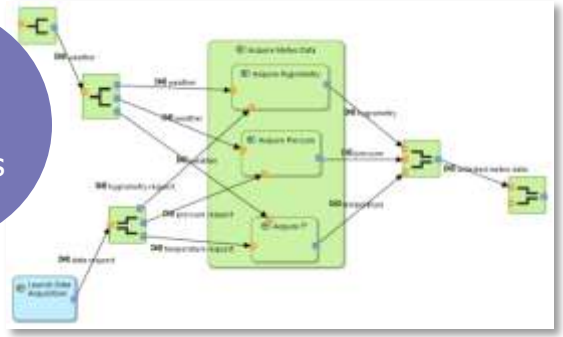
- Physical Links are non-oriented material connections between Node Components, through Physical Ports.
  - They realize Component Exchanges, and appear in red on the diagram

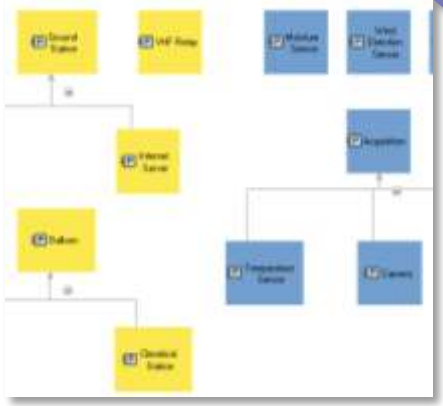
Physical Architecture Workflow and Main Diagrams

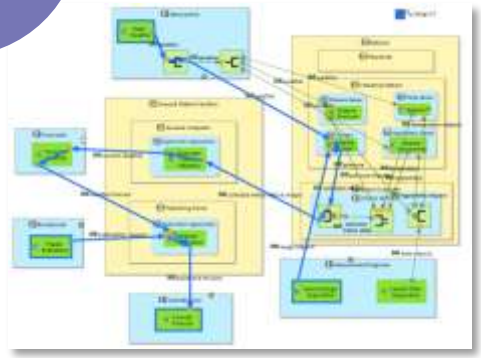












Transition from Logical Architecture

Refine Physical Functions

Describe Data Flow Scenarios

Physical Architecture

Assign Functions To Components

Describe Physical Data Flows

Define Physical Components

Refine Functional Chains

ARCADIA Concepts

End-Product Breakdown Structure

Define the End-Product Breakdown Structure

- At this step, **Configuration Items (CI) contents are to be defined in order to build a Product Breakdown Structure (PBS)**
  - By grouping various former components in a bigger CI easier to manage,
  - Or by federating various similar components in a single implementation CI that will be instantiated multiple times at deployment
- Defines the "final" architecture of the system at this level of engineering, ready to develop (by lower engineering levels)
- Allocation of requirements on configuration items
- Consideration of industrial & subcontracting constraints
- Outputs
  - EPBS

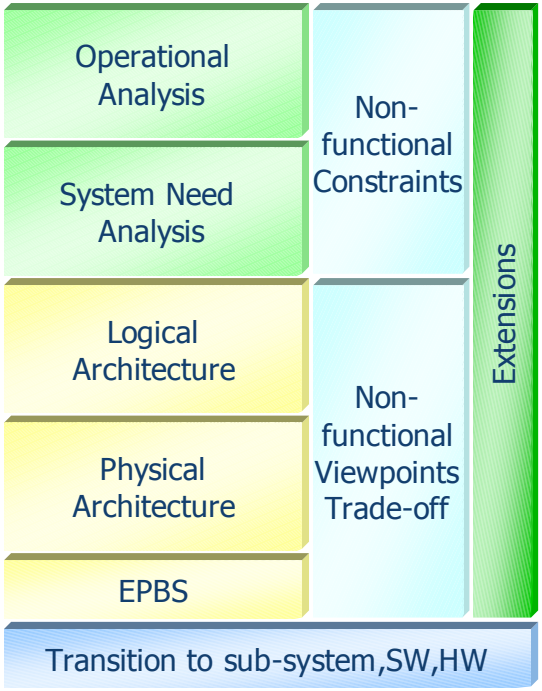
ARCADIA wrt Standards: xAF, SysML, AADL…

Yet another Formalism?

ARCADIA Scope & Coverage (1/2)

ARCADIA goes beyond Formalisms & Languages:

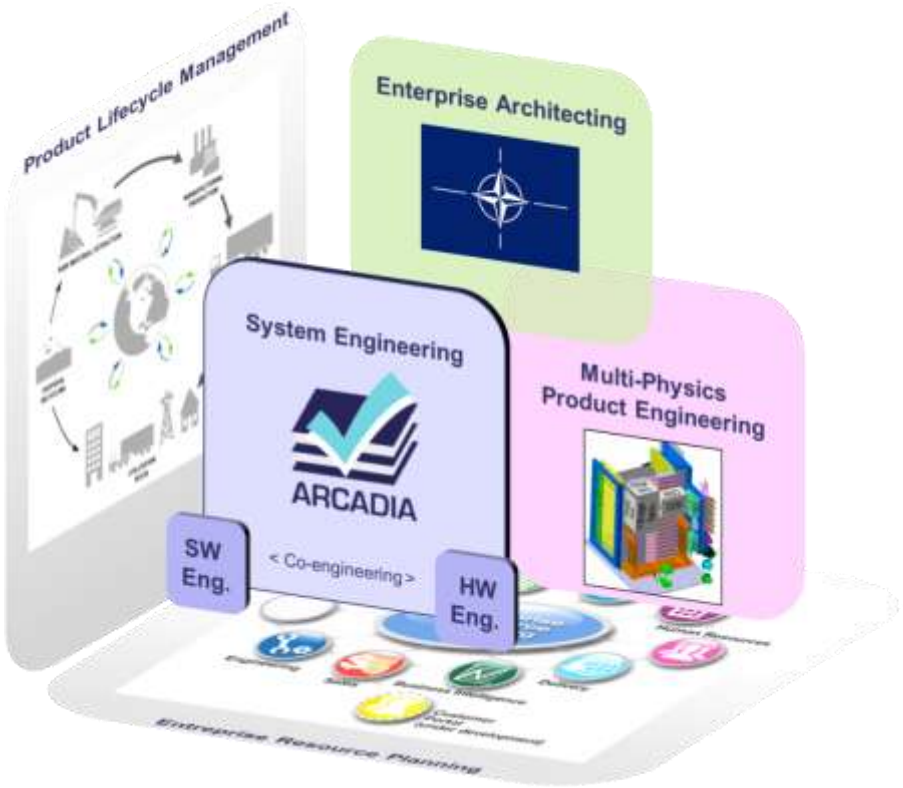
- Method defining full model design & conformance rules
  - How to define elements
  - How to link and relate them to each other
  - How to justify and check definition
- Operational Analysis & Capability integration
- Modelling Viewpoints
  for non-functional constraints support
  - Safety, Performance, HF, RAMST, Cost…



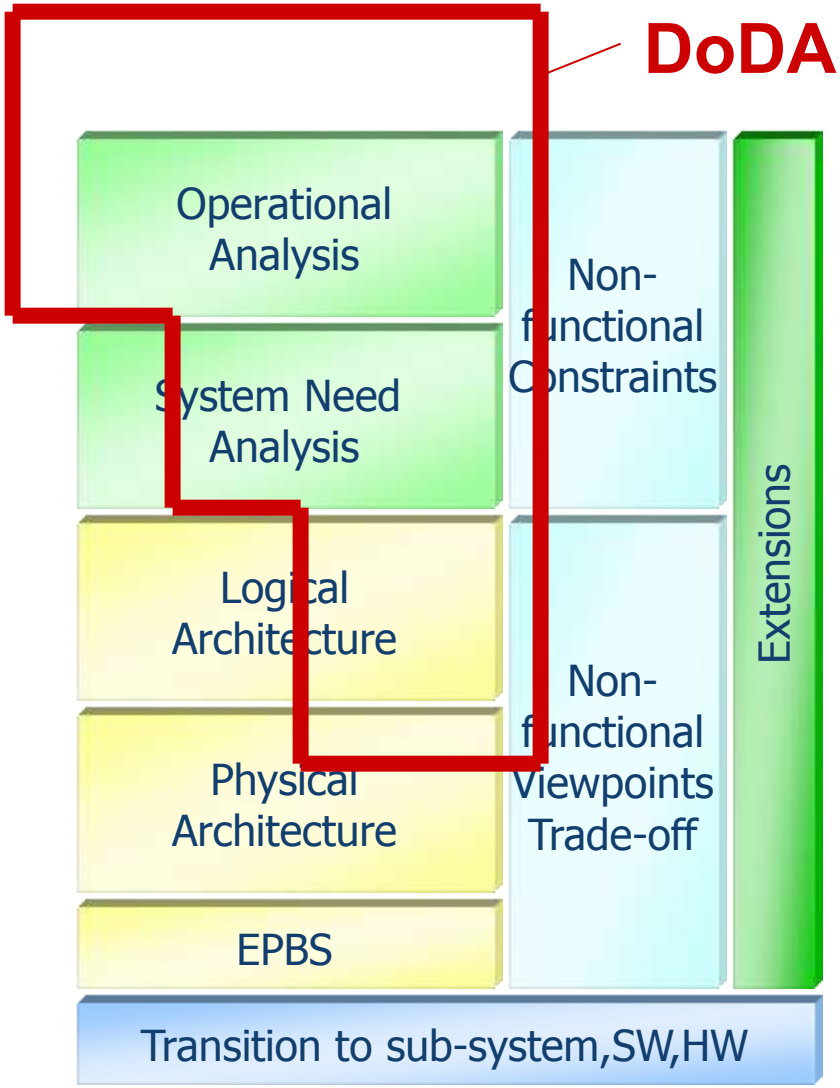
ARCADIA Scope & Coverage (2/2)

ARCADIA goes beyond Formalisms & Languages:

- Semantic architecture Validation through Engineering Rules formalisation
- Multi-viewpoints analysis coupled with fine-grained tuning
- Extensible: viewpoints, model, diagrams & rules



ARCADIA Formalism Vs DoDAF, NAF…



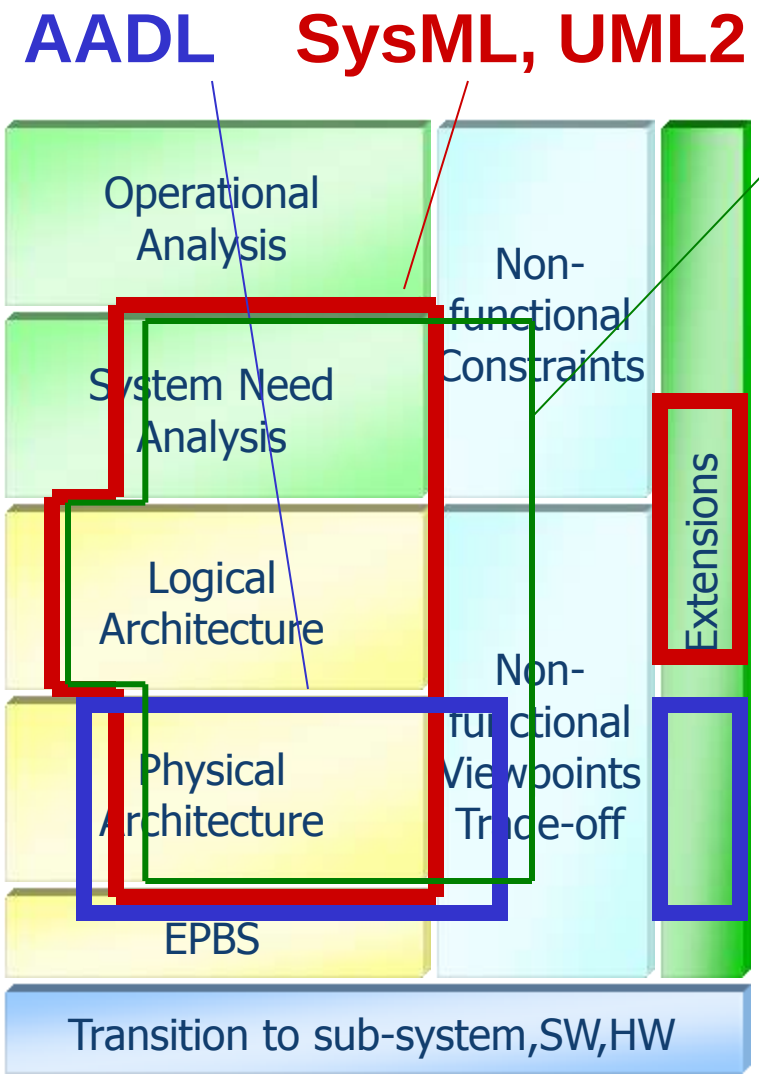
Common or Similar Concepts:

- Operational Entities, Actors, Roles
- Operational Activities, Processes
- Services (extension)
- States & modes
- Functions, dataflows
- System Nodes, equipment (generalised)
- Operational & system data…
- Traceability between operational & system

Similar Capella Diagrams:

- OV2, OV4, OV5, OV6, OV7;
- SOV;
- SV1, SV2, SV4, SV5, SV10…

ARCADIA Formalism Vs UML2, SysML, AADL…



Common or Similar Concepts:

- Blocks, parts, (sub)components
- Data, constraint properties
- Ports, flows
- States & modes
- Profiles & libraries…
- Bus, connection
- Processors, Memory… (extension)
- Threads, processes… (extension)

} AADL

Similar Capella Diagrams:

- Activity diagram,
- Block Definition diagram,
- Internal Block diagram ,
- Package diagram,
- Requirement diagram,
- Sequence diagram ,
- State Machine diagram ,
- Use Case diagram ,
- Allocation tables…

ARCADIA Vs Architecture Languages

ARCADIA concepts are directly compatible with architecture languages such as

- DODAF/NAF Architecture Frameworks,
- UML2 & SysML,
- AADL, …

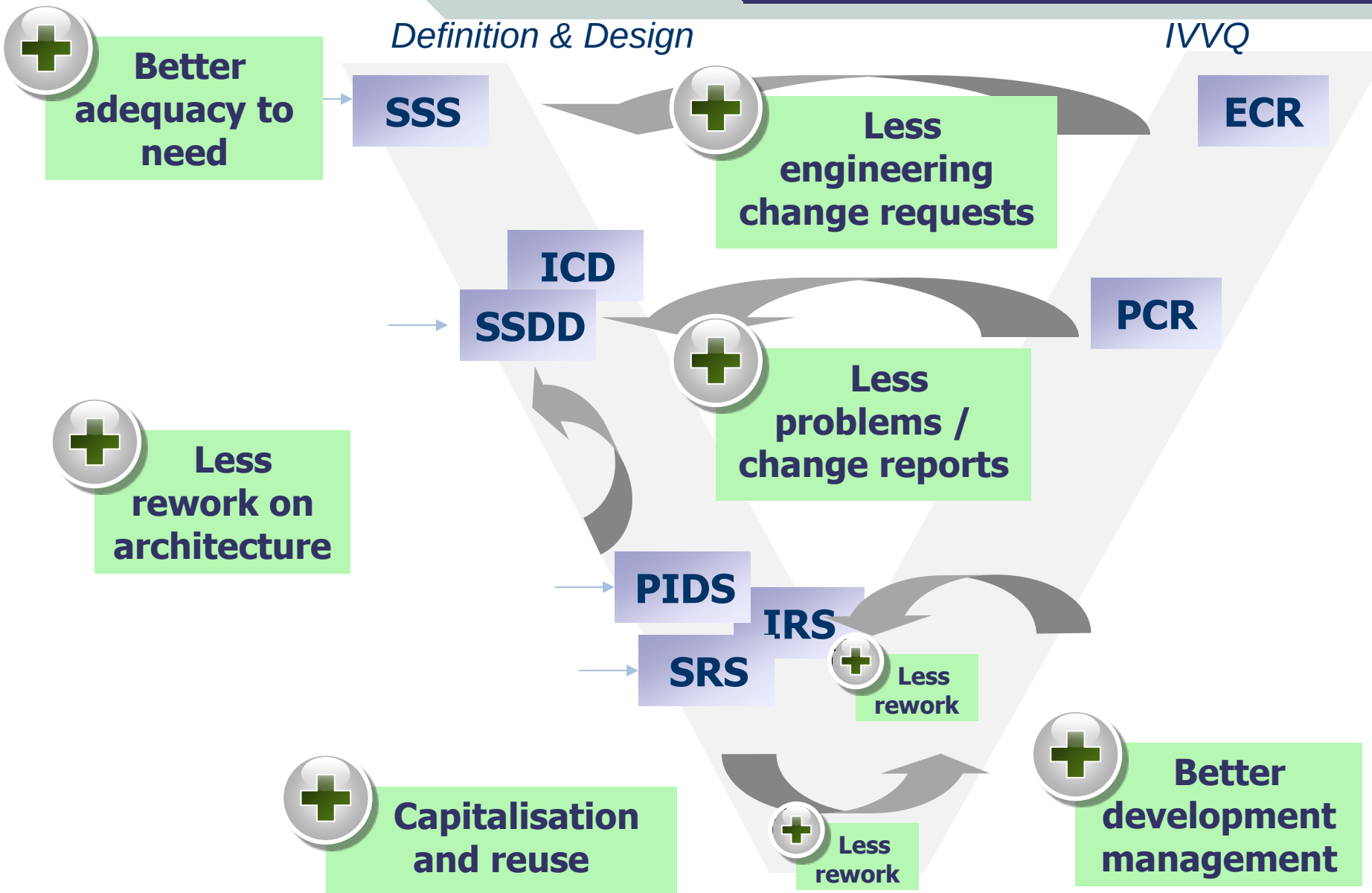
These formalisms can interoperate with ARCADIA, it is just a matter of import/export tooling:

- Export tooling can (will) be developed,
- Selective Import can (will) be developed (e.g. functional analysis, components…)
- Global Import could be possible under method conformance conditions

Benefits of ARCADIA

A quick summary of features and capabilities

Summary of Expected Benefits



Summary of Contribution to Expected Benefits

Better adequacy to need

Thanks to: Customer Need Analysis *justifying* Architecture

Less engineering change requests

Thanks to: Early Check against Operational Need

Less rework on architecture

Thanks to: Early Check of Architecture, non-functional Viewpoints modelling

Less problems / change reports

Thanks to: Integration Contract + Architecture designed for IVVQ

Capitalisation and reuse

Thanks to: Formalised Architecture, Viewpoints & Rules

Better development management

Thanks to: Integration Contract Formalisation