

Why are ARCADIA and Capella relevant for MBSE?

Introduction: Pascal Roques

- Senior consultant, 25+ years of modeling experience
 - SADT, OMT, UML, SysML, ARCADIA/Capella
- UML2 and SysML Certified by the OMG
- Co-founder of the  association
- Trainer for Thales on ARCADIA / Capella
 - 130+ sessions, 1500+ trainees
 - Part of  Clarity Ecosystem for the Model Based Systems Engineering Solution Capella
- Author of UML/SysML best-sellers in France
- ... and of the first Capella book soon!



Agenda



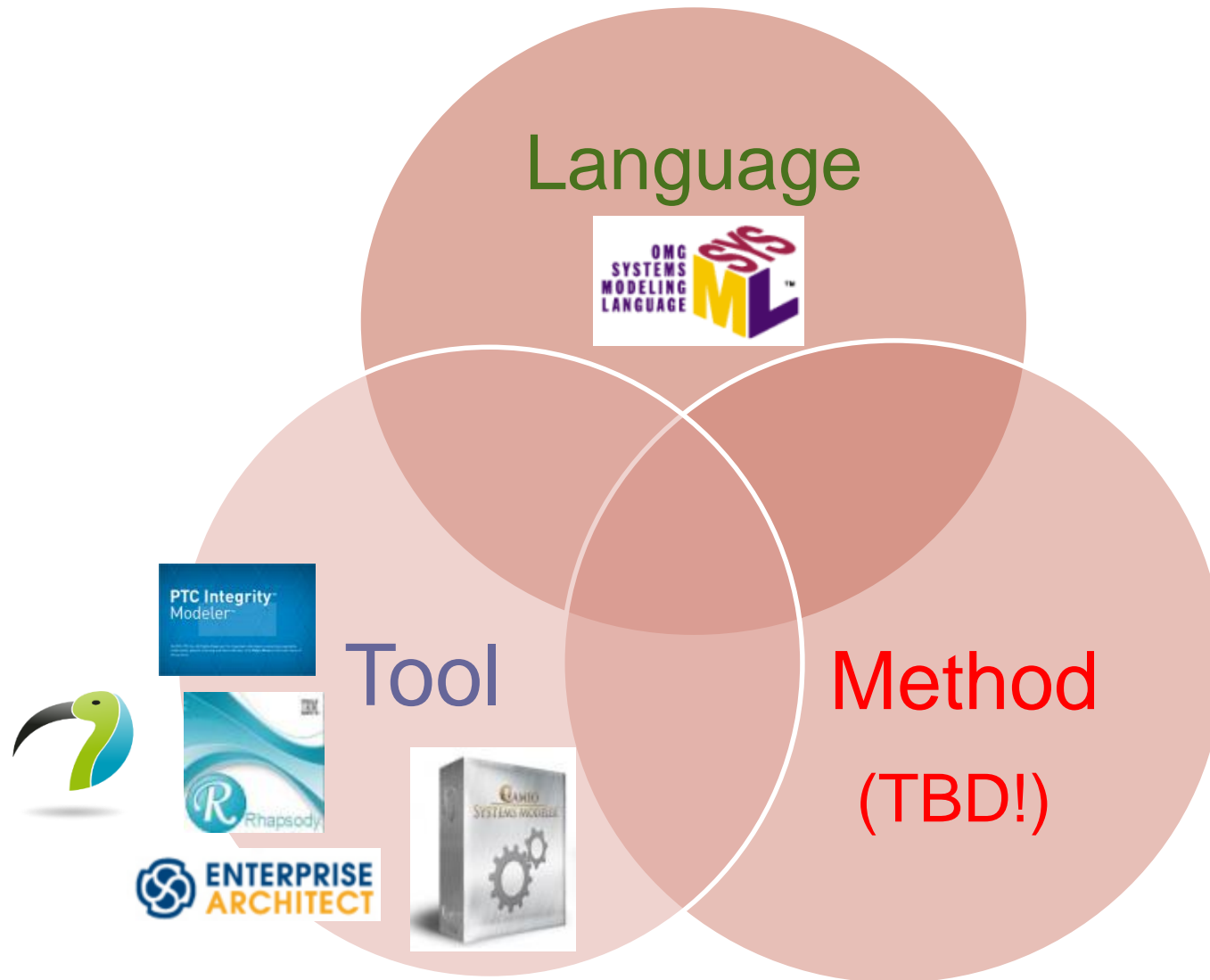
**1. Tooled-Up
Modeling Method**

**2. Systems
Engineering
Continuity**

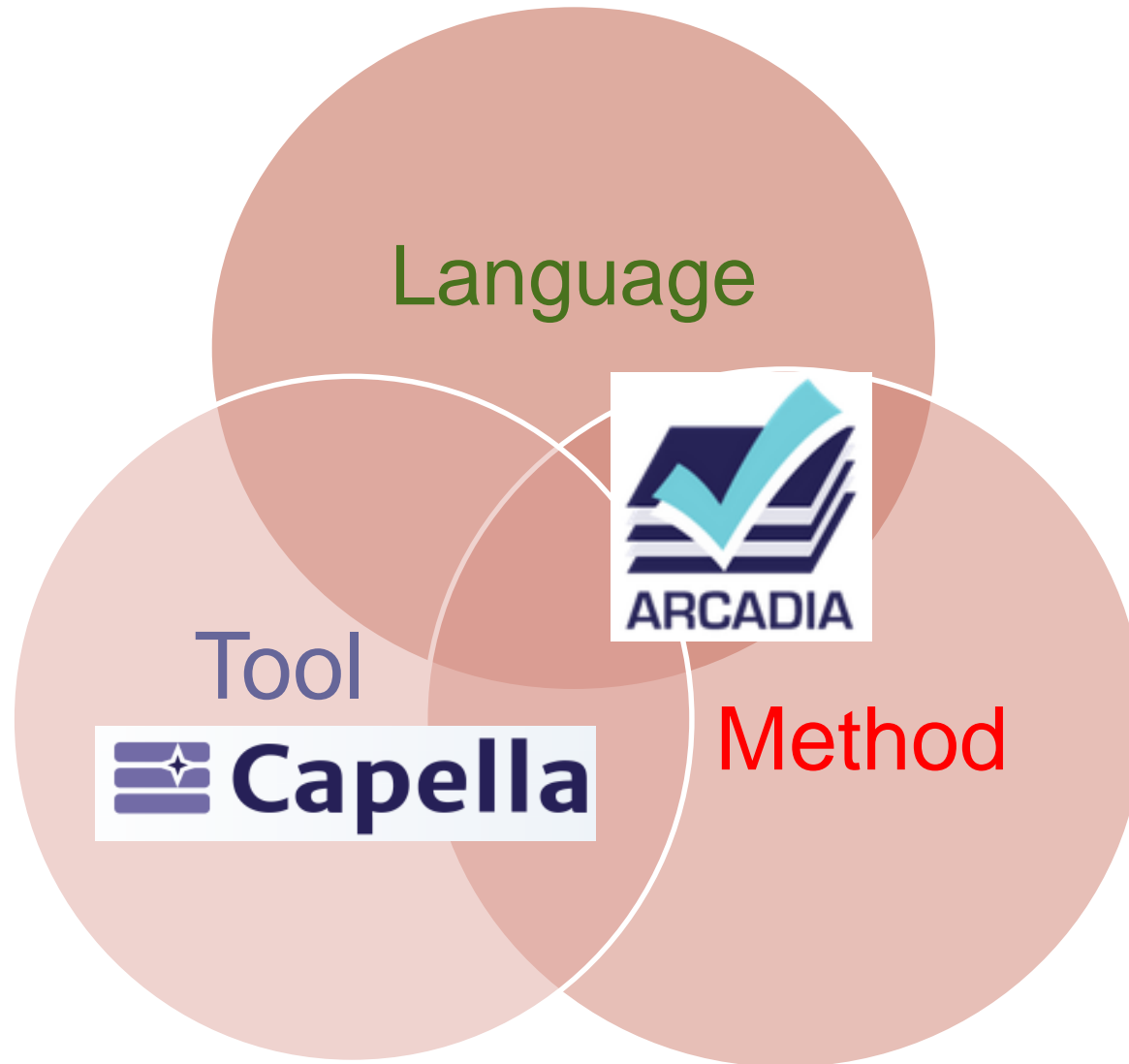
**3. User-Driven
Modeling Tool**

4. Open-Source!

MBSE pillars (SysML)



MBSE pillars (ARCADIA / Capella)



ARCADIA Summary



	METHOD STEPS	TASKS	SAMPLE MODEL	CONCEPTS	DESCRIPTION MEANS
NEED	Customer Operational Need Analysis What the users of the system need to accomplish	<ul style="list-style-type: none"> Define operational capabilities Perform an operational need analysis 		<ul style="list-style-type: none"> Operational capabilities Actors, operational entities Actor activities Interactions between activities & actors Information used in activities & interactions Operational processes chaining activities Scenarios for dynamic behaviour 	<p>Dataflow: functions, op. activities interactions & exchanges</p> <p>Scenarios: actors, system, components interactions & exchanges</p>
	System/SW/HW Need Analysis What the system has to accomplish for the Users	<ul style="list-style-type: none"> Perform a capability trade-off analysis Perform a functional and non-functional analysis Formalise and consolidate requirements 		<ul style="list-style-type: none"> Actors and system, capabilities Functions of system & actors Dataflow exchanges between functions Functional chains traversing dataflow Information used in functions & exchanges, data model Scenarios for dynamic behaviour Modes & states 	<p>Functional chains, operational processes through functions & op. activities</p>
SOLUTIONS	Logical Architecture Design How the system will work so as to fulfil expectations	<ul style="list-style-type: none"> Define architecture drivers and viewpoints Build candidate architectural breakdowns in components Select best compromise architecture 		<p>SAME CONCEPTS, PLUS :</p> <ul style="list-style-type: none"> Components Component ports and interfaces Exchanges between components Function allocation to components Component interface justification by functional exchanges allocation 	<p>Breakdown of functions & components</p>
	Physical Architecture Design How the system will be developed & built	<ul style="list-style-type: none"> Define architectural patterns Consider reuse of existing assets design a physical Design a physical reference architecture Validate and check it 		<p>SAME CONCEPTS, PLUS :</p> <ul style="list-style-type: none"> Behavioural components refining logical ones, and implementing functional behaviour Implementation components supplying resources for behavioural components Physical links between implementation components 	<p>Data model: dataflow & scenario contents, definition & justification of interfaces</p>
	Development Contracts What is expected from each designer/sub-contractor	<ul style="list-style-type: none"> Define a components IVVQ strategy Define & enforce a PBS and component integration contract 		<ul style="list-style-type: none"> Configuration items tree Parts numbers, quantities Development contract (expected behaviour, interfaces, scenarios, resource consumption, non-functional properties...) 	<p>Component wiring: all kinds of components</p>
				<p>Allocation of op. activities to actors, of functions to components, of behav.components to impl.components, of dataflows to interfaces, of elements to configuration items</p>	

Methodological Guidance

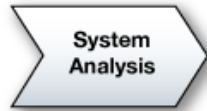
Activity Browser



Operational Analysis

Define Stakeholder Needs and Environment

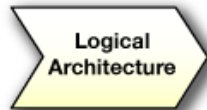
Capture and consolidate operational needs from stakeholders
Define what the users of the system have to accomplish
Identify entities, actors, roles, activities, concepts



System Analysis

Formalize System Requirements

Identify the boundary of the system, consolidate requirements
Define what the system has to accomplish for the users
Model functional dataflows and dynamic behaviour



Logical Architecture

Develop System Logical Architecture

See the system as a white box: define how the system will work so as
Perform a first trade-off analysis



Physical Architecture

Develop System Physical Architecture

How the system will be developed and built
Software vs. hardware allocation, specification of interfaces,
deployment configurations, trade-off analysis



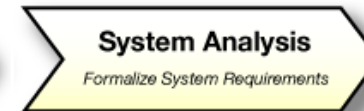
EPBS

Formalize Component Requirements

Manage industrial criteria and integration strategy: what is expected from each designer/sub-contractor
Specify requirements and interfaces of all configuration items



Operational Analysis



System Analysis

Formalize System Requirements



Logical Architecture

Transition From Operational Activities

Define Actors, Missions and Capabilities

Refine System Functions, describe Functional Exchanges



[SFBD] Create a new Functional Breakdown diagram



[SDFB] Create a new Functional Dataflow Blank diagram



[FS] Create a new Functional Scenario

Allocate System Functions to System and Actors



[SAB] Create a new System Architecture diagram



[ES] Create a new Exchange Scenario

Methodological Guidance

■ Semantic Browser

The screenshot displays the Semantic Browser window for the logical function '[Logical Function] Acquire Met Data'. The interface is divided into three main sections: Referencing Elements, Current Element, and Referenced Elements.

Referencing Elements:

- Allocating Logical Component
 - Airborne Subsystem
- Functional Chains
 - Acquisition FC
- Incoming Functional Exchanges
 - env. conditions
 - Provide Env. Conditions
 - met data request
 - Launch Data Acquisition
- In Flow Ports
- Realizing Physical Functions
 - Acquire Met Data
- Scenarios
 - [LES] Acquisition Scenario

Current Element:

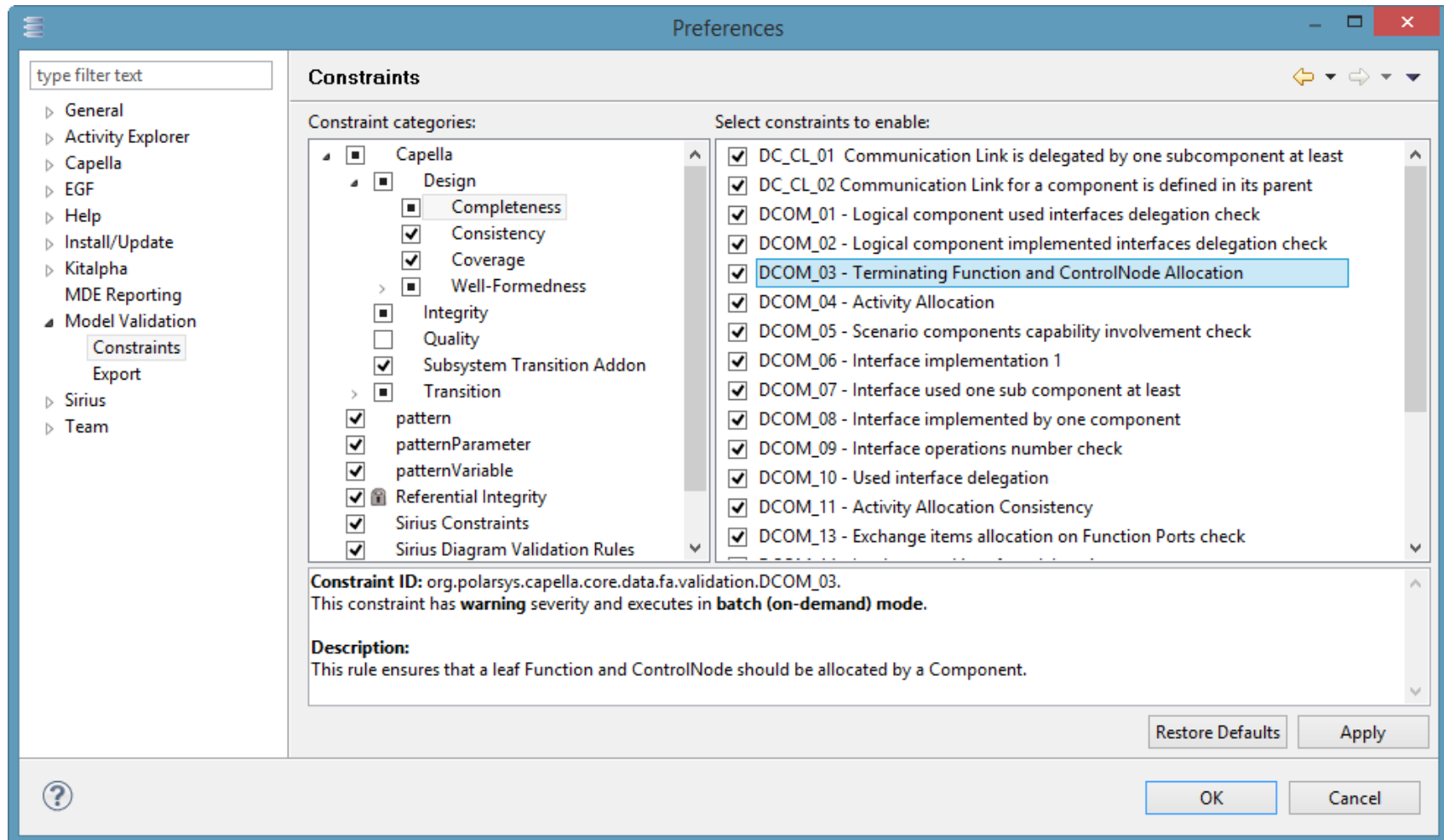
- Acquire Met Data
 - Parent
 - Collect Met Data
 - All Related Diagrams
 - [LAB] Logical System
 - [LES] Acquisition Scenario

Referenced Elements:

- Out Flow Ports
- Outgoing Functional Exchanges
 - collected met data
 - Elaborate Current Situation
- Realized System Functions
 - Acquire Met Data

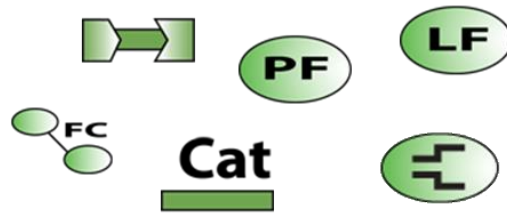
Methodological Guidance

■ Model Checking

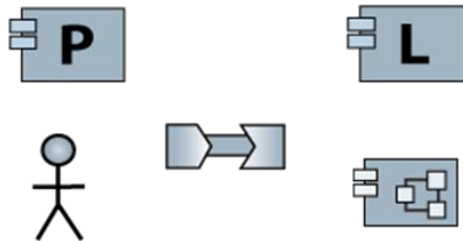


Methodological Guidance

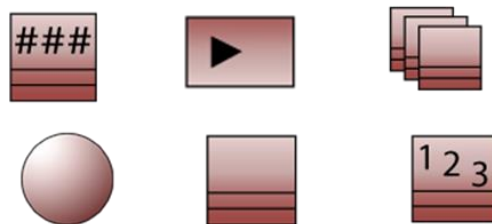
■ Semantic Color Map



Functional analysis
= Green



Components engineering
= Blue



Data models and interfaces =
Pink

Agenda

1. Tooled-Up
Modeling Method

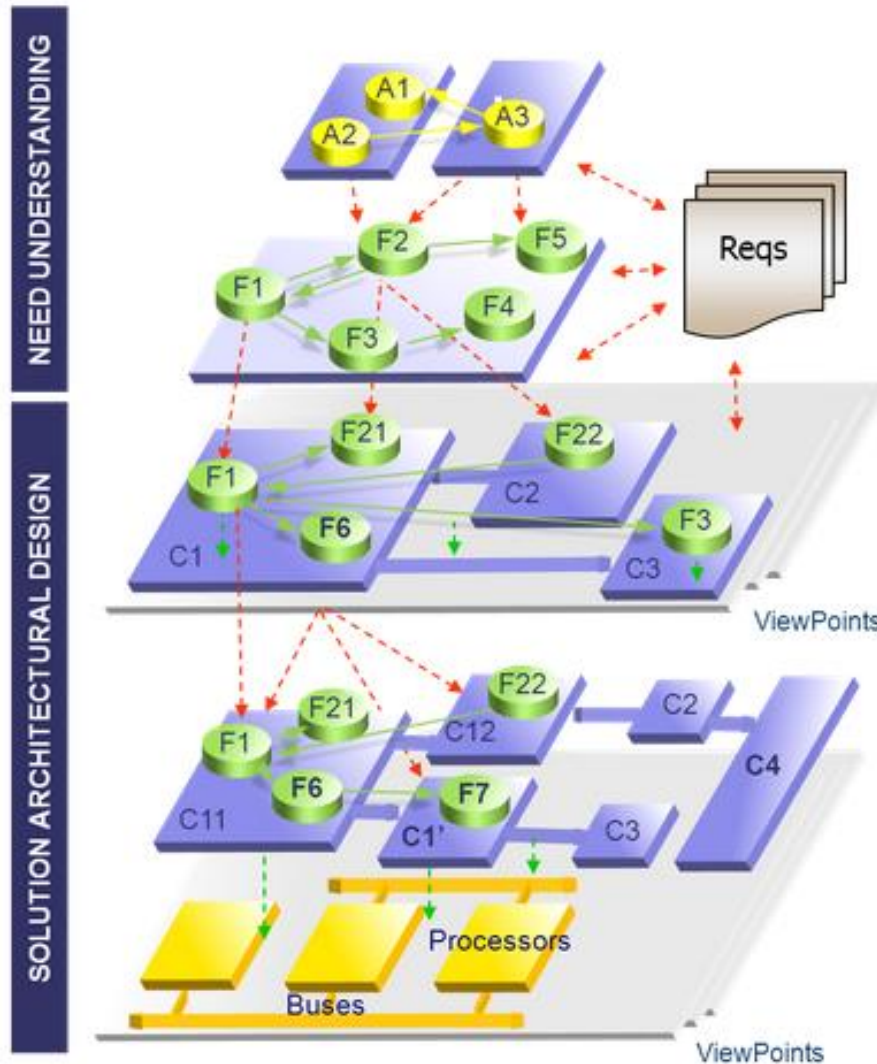


2. Systems
Engineering
Continuity

3. User-Driven
Modeling Tool

4. Open-Source!

ARCADIA: Global View



What the users of the system need to accomplish

What the system has to accomplish for the users

How the system will work to fulfill expectations

How the system will be developed and built

Automated Transitions

- Functions / Actors / etc.

Logical Architecture ▾

System Analysis **Logical Architecture**
Develop System Architectural Design Physical Architecture

- ▶ Transition from System Functions ? ⚙
- ▶ Refine Logical Functions, describe Functional Exchanges ? ⚙
- ▾ 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](#)
- ▶ Allocate Logical Functions to Logical Components ? ⚙
- ▶ Delegate System Interfaces and create Logical Interfaces ? ⚙
- ▶ Enrich Logical Scenarios ? ⚙
- ▶ Transverse Modeling ? ⚙

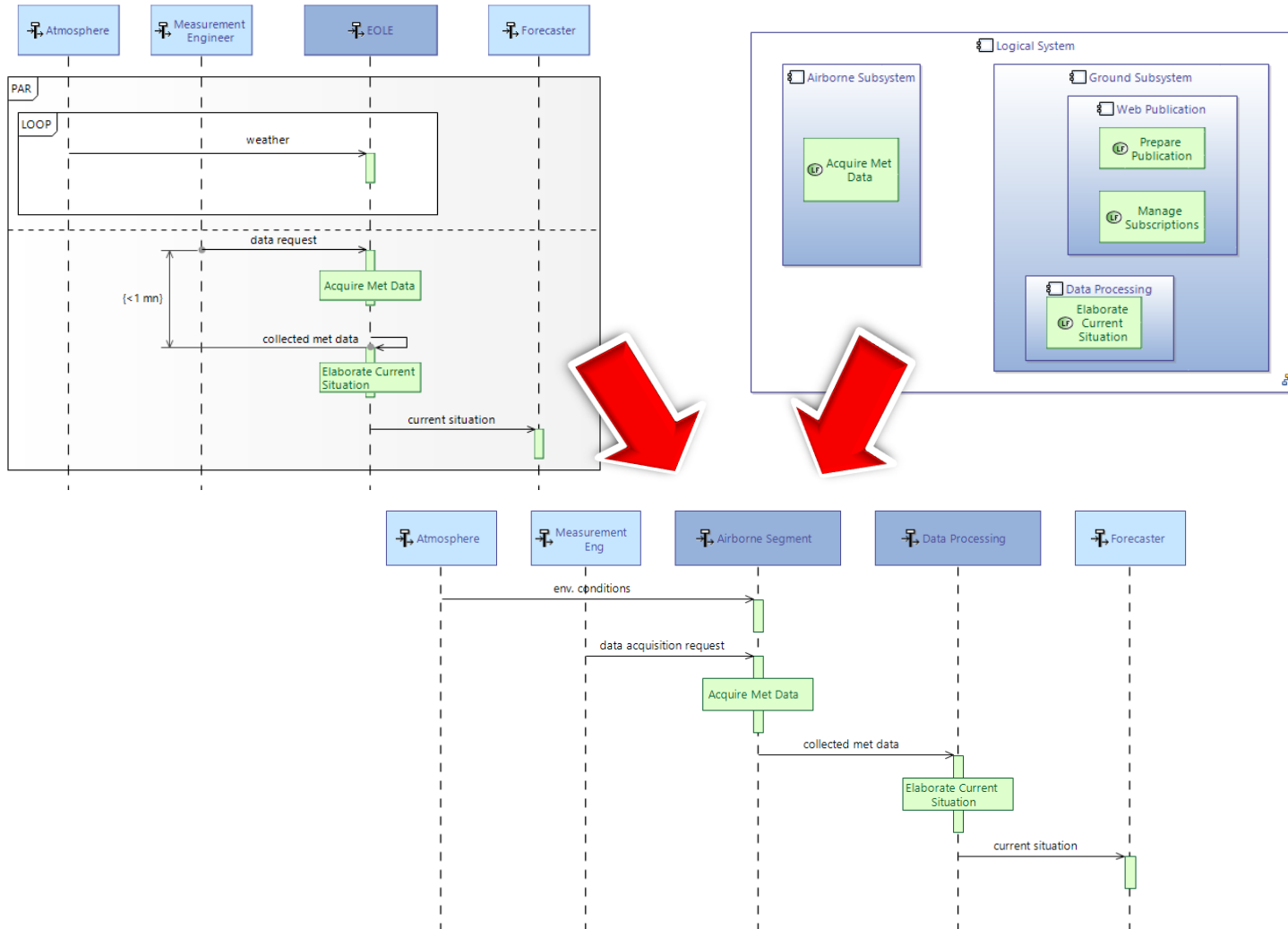
Logical Architecture ▾

System Analysis **Logical Architecture**
Develop System Architectural Design Physical Architecture

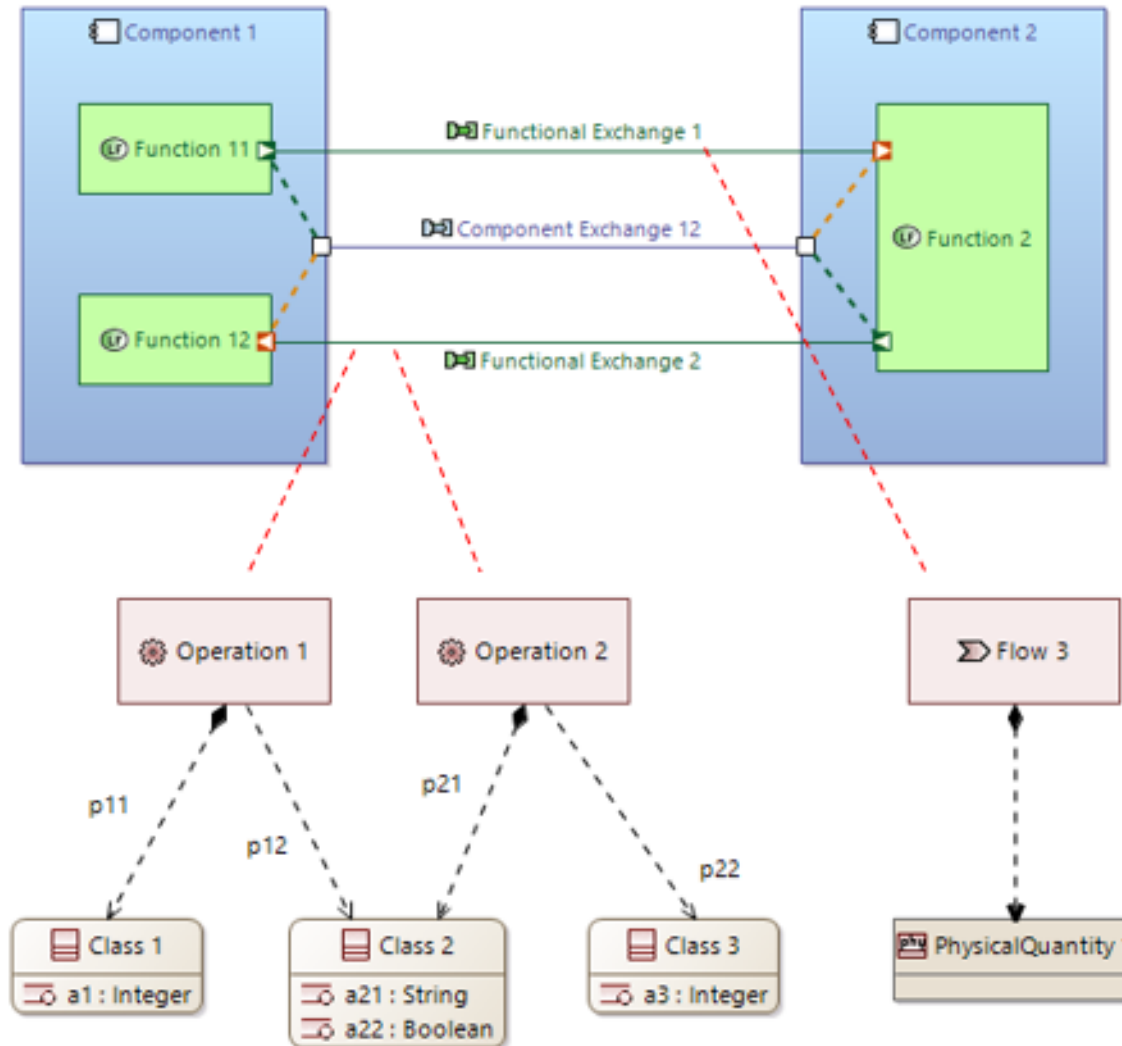
- ▾ Transition from System Functions ? ⚙
 - [Perform an automated transition of System Functions](#)
 - [Create Traceability Matrix](#)
- ▶ Refine Logical Functions, describe Functional Exchanges ? ⚙
- ▶ Define Logical Components and Actors ? ⚙
- ▶ Allocate Logical Functions to Logical Components ? ⚙
- ▶ Delegate System Interfaces and create Logical Interfaces ? ⚙
- ▶ Enrich Logical Scenarios ? ⚙
- ▶ Transverse Modeling ? ⚙

Automated Transitions

- Even Scenario Transition!



Functions / Components / Data



Components



Functions

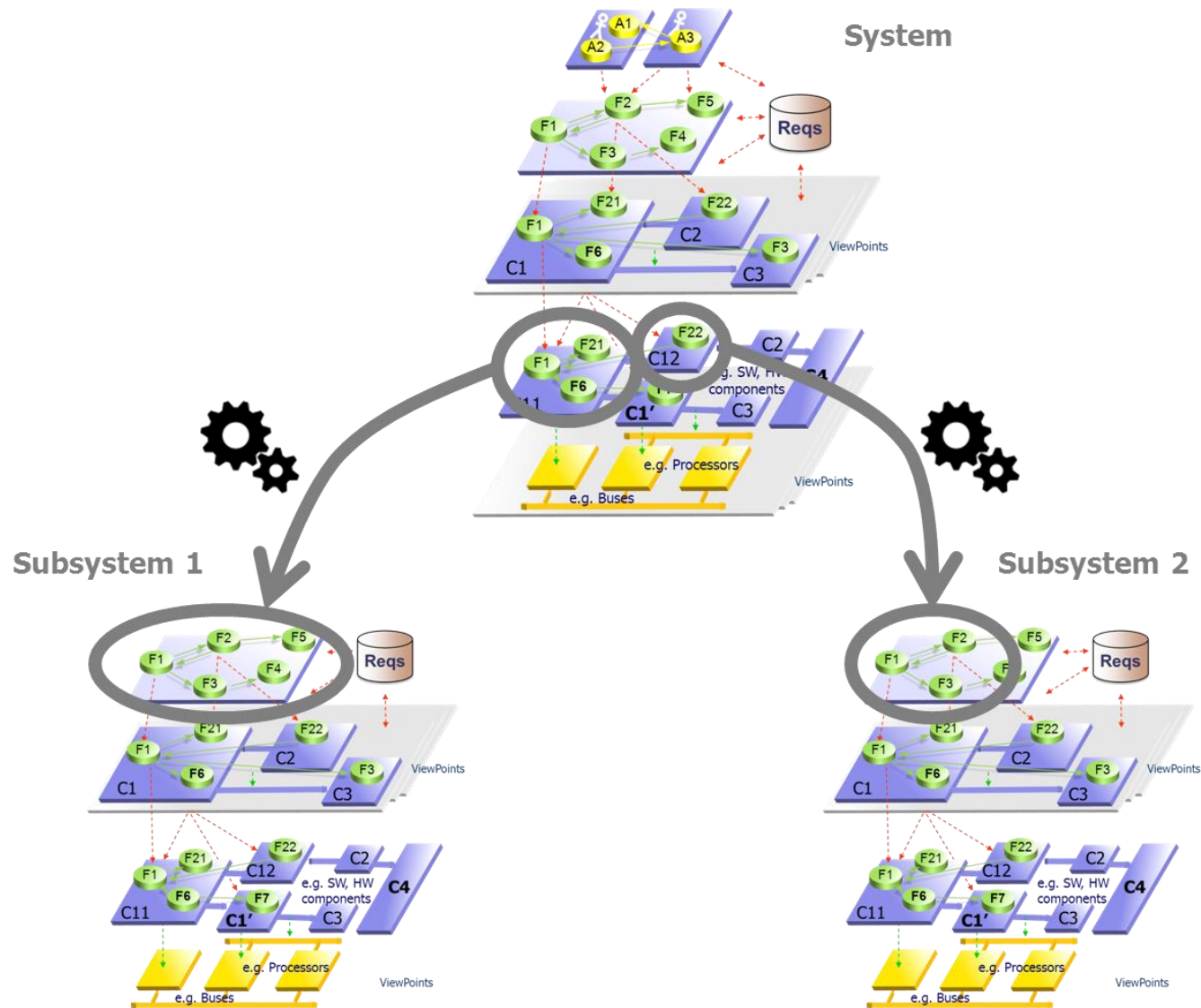


Exchange Items

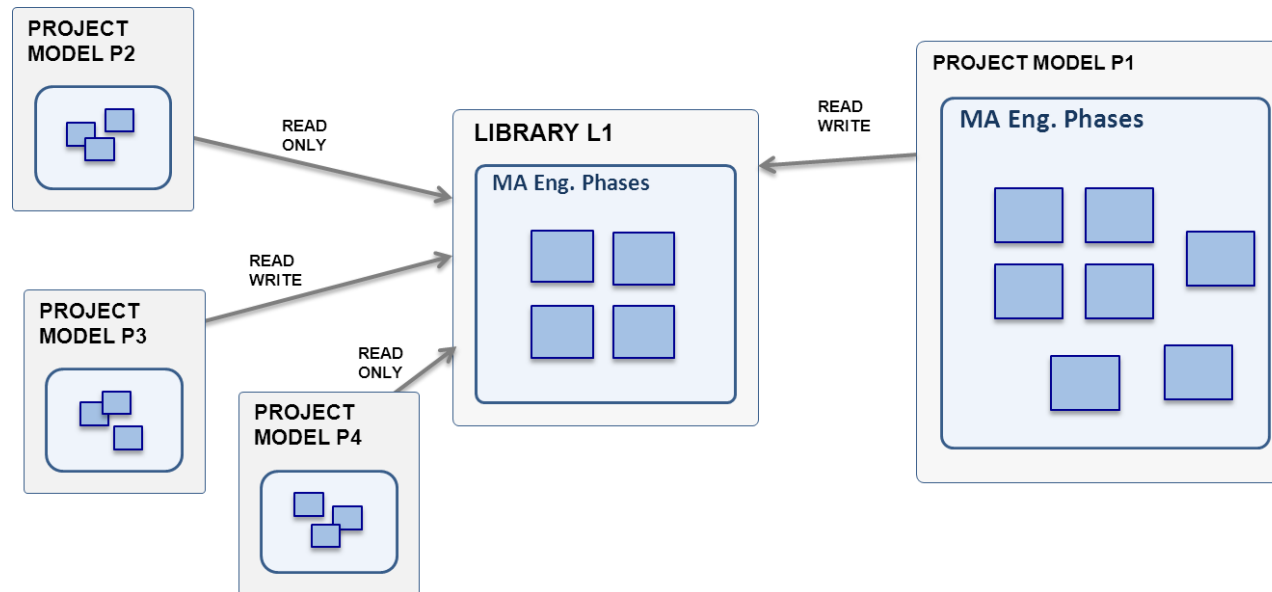
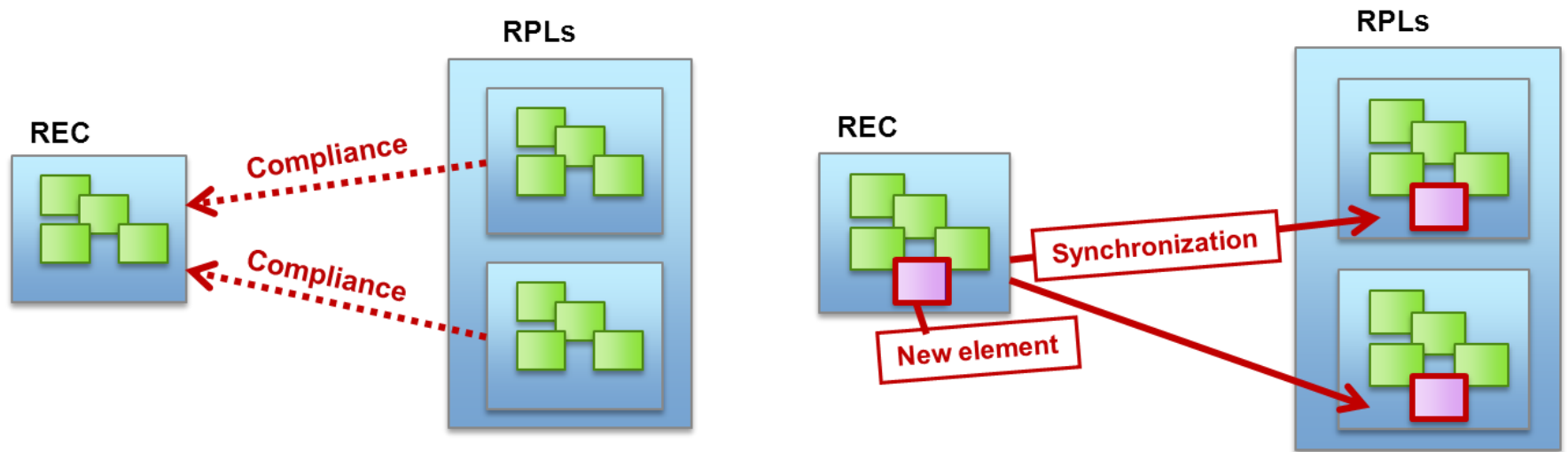


Types

System/Subsystem Transition



Replicable Elements and Libraries



Architecture Evaluation

Viewpoints

The screenshot displays a software architecture tool interface. The main window shows a physical system diagram with components like Atmosphere, Balloon, Ground Station, and various sensors and computers. A blue arrow highlights a data flow from the Balloon's 'Acquire Images' component to the Ground Station's 'Elaborate current situation' component. The bottom of the interface features an 'Architecture Evaluation (basic)' section with a bar chart and a 'Viewpoint Manager' table.

Viewpoint	Property	Value
Basic Price	weight	1
Basic Price	critical	false
Basic Performance	weight	1
Basic Performance	critical	false
Basic Mass	weight	1
Basic Mass	critical	false

Indicator	Measure
Basic Mass	1.0666
Basic Price	0.9567
Basic Performance	0.9567
Synthesis	0.9999

Capella Studio

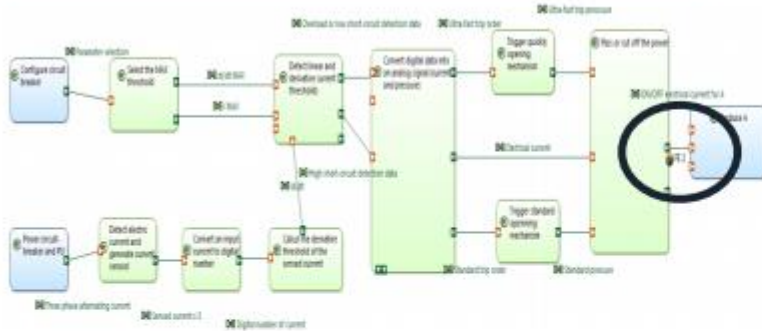


1. Integration of Capella in Kitalpha
2. Definition of a Capella Target Application
3. Integration of Capella-specific generators
4. Extensions of the textual editors
5. Customization of the html documentation generation

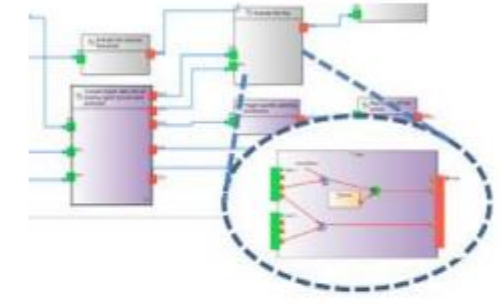
Architecture Evaluation

- Coupling with External Tools
 - Example: Safety Architect

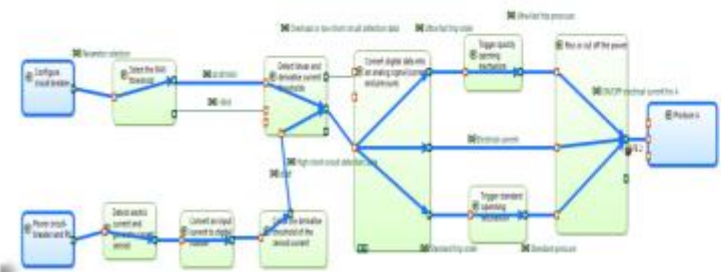
Feared event added to Capella dataflows (viewpoint)



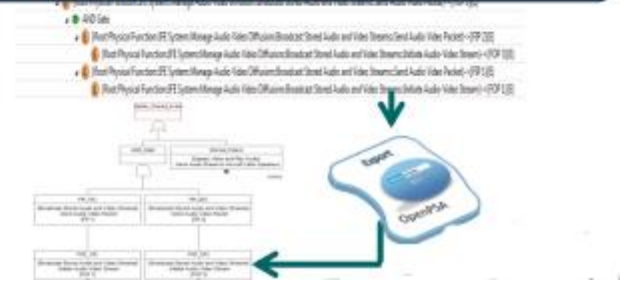
In Safety Architect, analysis of block local failure conditions



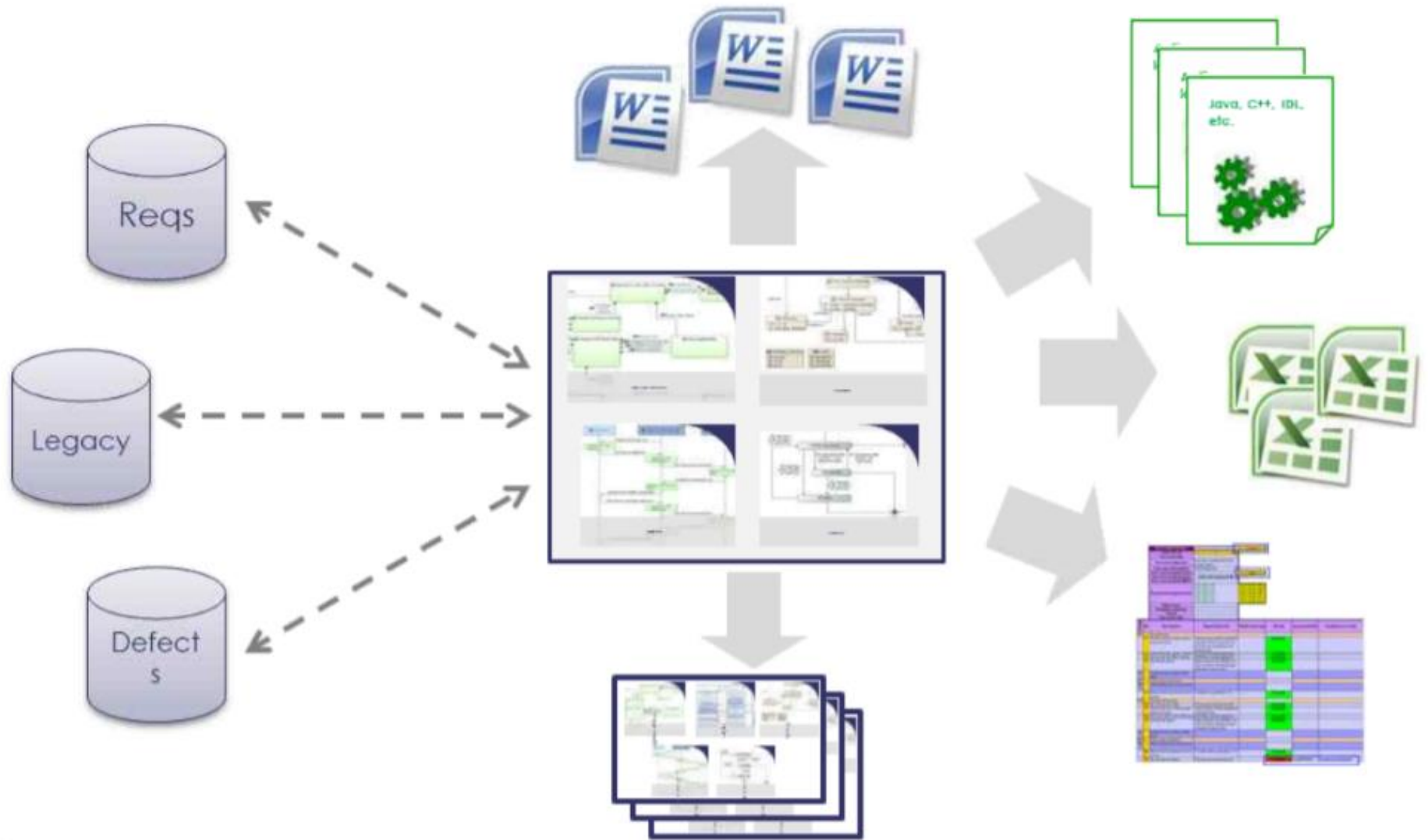
In Capella, visualization of fault trees as critical functional chains



In Safety Architect, automated generation of fault-trees



True MBSE!



Agenda

1. Tooled-Up
Modeling Method

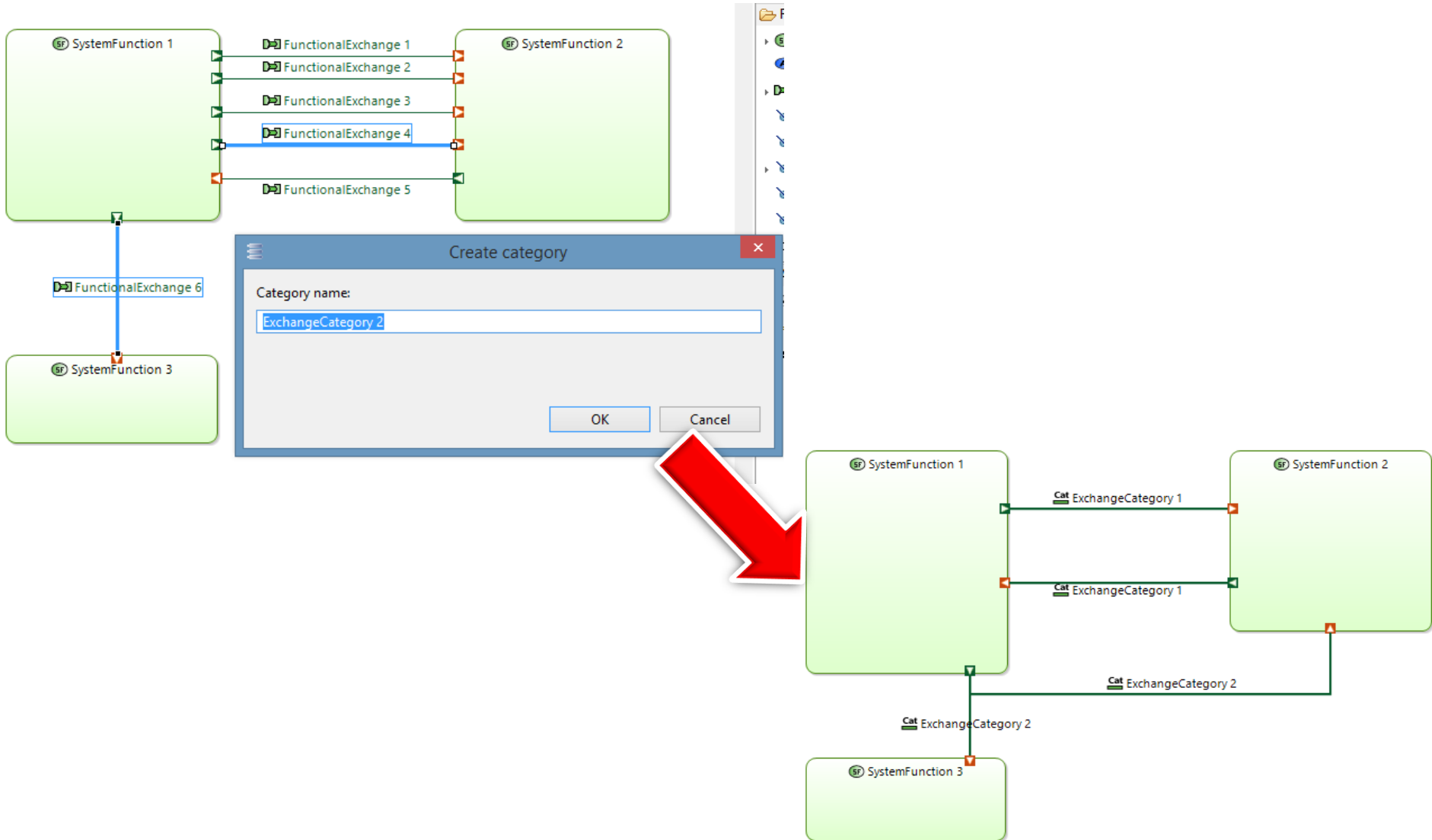
2. Systems
Engineering
Continuity

3. User-Driven
Modeling Tool

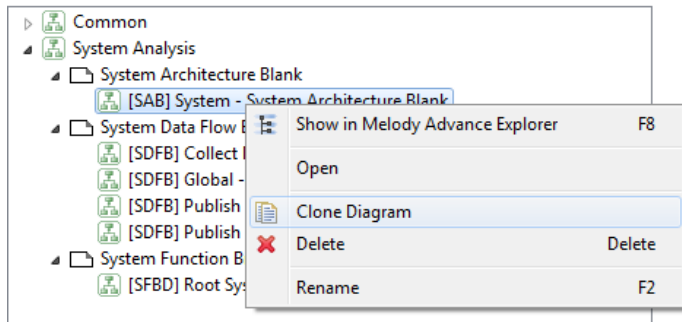
4. Open-Source!



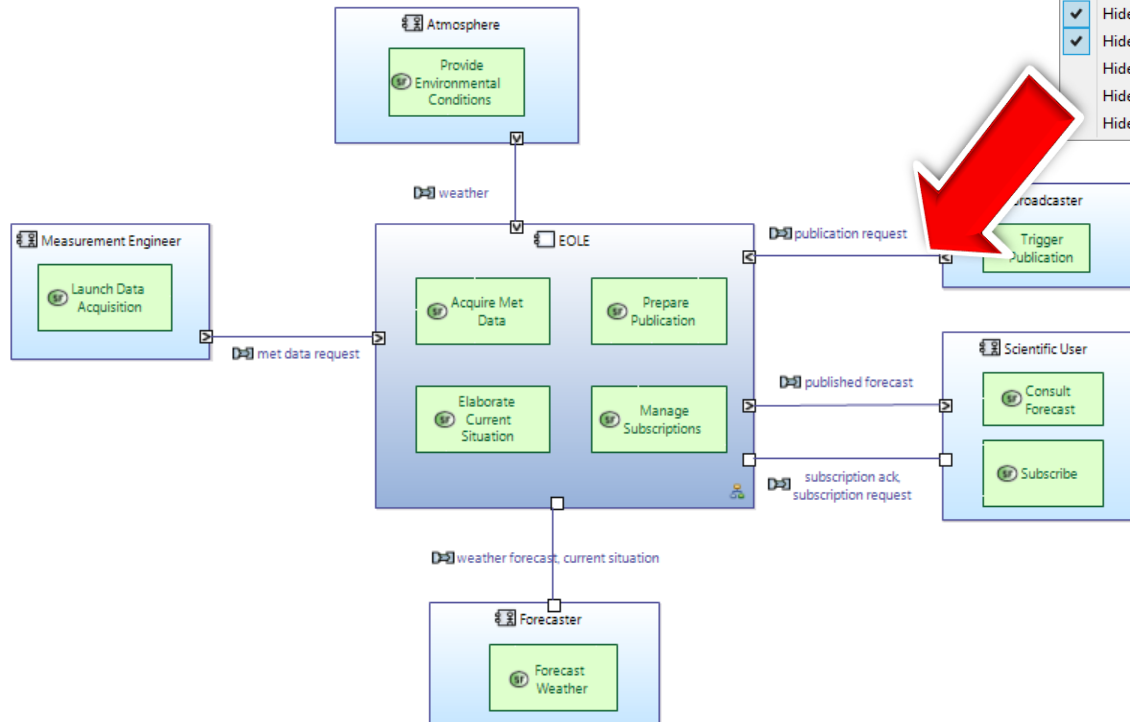
Category



Advanced Diagram Management



- Collapse Component Ports
- Collapse Function Ports
- Hide Function Ports without Exchanges
- Hide Component Ports without Exchanges
- Hide Allocated Functional Exchanges
- Hide Functions
- Hide Functional Exchanges
- Hide Component Exchanges
- Hide Port Allocations
- Hide Allocated Function Ports
- Show Exchange Items on Functional Exchanges
- Show Exchange Items on Component Exchanges
- Show Exchange Items on Component Exchange (without Functional Exchanges)
- Show Allocated Functional Exchanges on Component Exchanges
- Hide cross Functional Exchanges of reusable Components
- Hide Simplified Grouped Component Exchanges
- Hide Simplified Oriented Grouped Component Exchanges
- Hide Functional Exchanges names
- Hide Component Exchanges names
- Hide Physical Links names



Powerful Accelerators!

The image displays a software development environment with two windows. The top window, titled 'test - Overview', shows a high-level 'Logical System' diagram. The bottom window, also titled 'test - Overview', shows a detailed logical architecture diagram for '[LAB] Logical System - Logical Architecture Blank'. This diagram features a central 'Logical System' component connected to several external components: 'Atmosphere' (containing 'Make Weather'), 'Measurement Engineer' (containing 'Launch Data Acquisition'), 'Broadcaster' (containing 'Trigger Publication'), 'Scientific User' (containing 'Consult Forecast' and 'Subscribe'), and 'Forecaster' (containing 'Forecast Weather'). Connections are labeled with 'D=UI' and specific UI elements like 'air', 'Broadcaster UI', 'internet', 'Subscription UI', and 'Forecaster UI'.

On the right side, a 'Palette' window is visible, showing a list of 'Accelerators'. The 'Initialization from existing diagram' accelerator is highlighted in yellow. A red arrow points from this accelerator to a 'Selection Wizard' dialog box. The dialog box contains the following text:

Selection Wizard
Select existing diagram for initialization.

Select a name to find
? = any character, * = any string
type filter text

- [SAB] EOLE
- [SAB] EOLE External View
- [SAB] EOLE Functional View
- [SAB] EOLE Synthetic View

Buttons for '?' (help), 'OK', and 'Cancel' are at the bottom.

A text box next to the 'Initialization from existing diagram' accelerator provides the following description:

Initializes the current diagram according to an existing diagram defined in the previous architecture. It adds realizing elements for each element from the source diagram and preserves layout between diagrams. This tool does not modify the semantic model.

Model Diff / Merge

The screenshot displays the Capella software interface in a comparison mode. The window title is "Capella - Compare ('Airborne' - 'Airborne_Mai') - Capella". The menu bar includes "File", "Edit", "Navigate", "Search", "Project", "Run", "Window", and "Help". The toolbar contains various icons for file operations and navigation.

The interface is divided into three main panes:

- Capella Project Explorer:** Located on the left, it shows a tree view of the project structure. A search box is present with the text "Select a name to find" and a hint "? = any character, * = any string". The tree shows folders for "Airborne" and "Airborne_Mai", each containing "Airborne.aird" and "Airborne.melodymodeller" files. Other folders like "EOLE", "EOLE_Mai", and "In-Flight Entertainment System" are also visible.
- Compare ('Airborne' - 'Airborne_Mai'):** The central pane, titled "Synthesis", shows a hierarchical comparison of the two models. It lists various components and their counts, such as "Airborne_Mai.aird (9)", "[SAB] Airborne Subsystem (434)", "[SES] Acquisition Main Success Scenario (329)", "Metadata_sLGW4DvZEel6e7Y1JruIA (1)", "Airborne (36)", "Library Dependencies (1)", "Safran_Lib [readOnly]", "Airborne_Mai (32)", "Safran_Lib (8)", "Metadata_8JDbcCqNEeeNVaq-pzvbqA (1)", "Viewpoint Reference_8J5I8CqNEeeNVaq-pzvbqA", and "Met_Library (109)".
- Model Views:** On the right, two side-by-side views show the internal structure of the compared models. The left view is for "Airborne/Airborne.aird" and the right view is for "Airborne_Mai/Airborne_Mai.aird". Both views show a similar hierarchy of components, including "Airborne.aird", "[SES] Example Scenario", "[SAB] Airborne Subsystem", "Metadata_8JDbcCqNEeeNVaq-pzvbq", "Airborne", and "Met_Library".

Agenda

1. Tooled-Up
Modeling Method

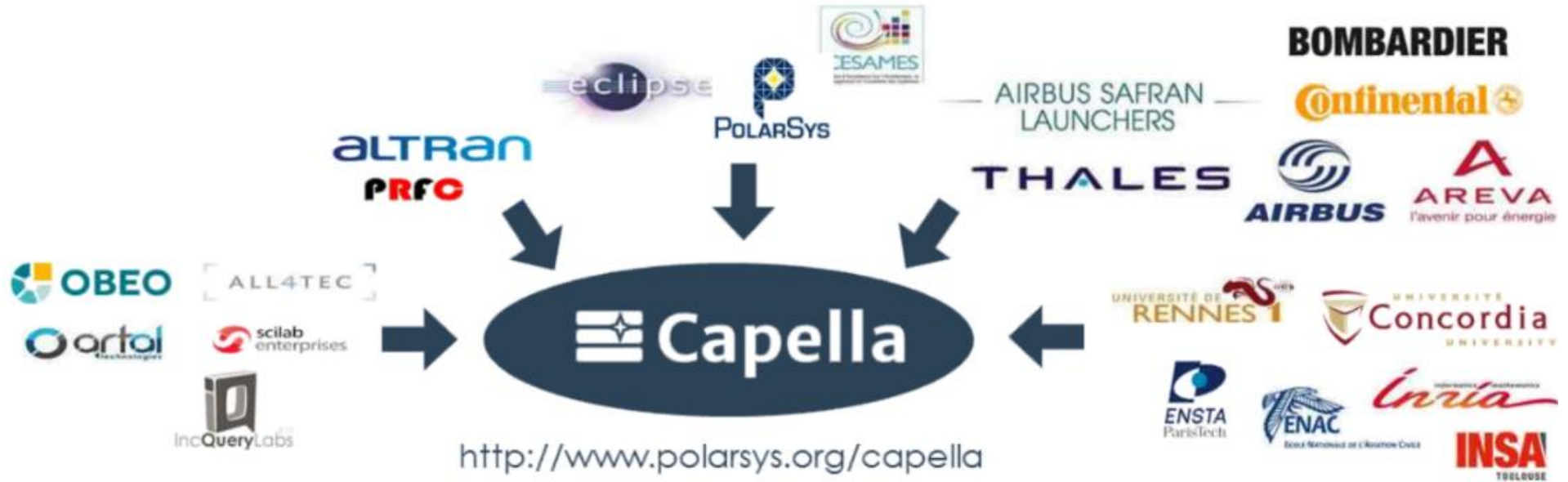
2. Systems
Engineering
Continuity

3. User-Driven
Modeling Tool

4. Open-Source!



Capella Ecosystem



Initial 3-year (French) collaborative project

Larger industry consortium currently being initiated

To Learn More...

Web Sites:

- www.polarsys.org/capella
- [//wiki.polarsys.org/Capella](http://wiki.polarsys.org/Capella)
- [//polarsys.org/forums/index.php/f/13/](http://polarsys.org/forums/index.php/f/13/)

- www.obeo.fr/en/capella-professional-offer

- www.prfc.fr

- www.clarity-se.org/

