
A method best-supported by

Capella

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

January 15th, 2015
This set of slides is a light introduction to Arcadia. The method is today extensively detailed through different Thales documents which are not publicly available: Reference guide, practitioner’s guide, encyclopedia of concepts, etc. Fully publishing publicly the detailed Arcadia method is a major objective of the 3-years Clarity consortium. This will however not be immediate, as a significant work is necessary to remove Thales references and find the best support (books, standardization technical reports, etc.).

The public publishing process will also probably lead to an adaptation of some of the Arcadia existing terminology. Without changing the goals and semantics of the Arcadia current content, a few concepts will most likely be renamed.

For any question about the method and its usage within Thales or to directly exchange with us, please contact arcadia-contact@thalesgroup.com
Contents

- Current Engineering Practices and Gaps
- ARCADIA Goals and Action Means
- ARCADIA Concepts
  - Examples
  - Capella, a workbench supporting the method
  - Engineering Steps
  - Focus on Functional Analysis
  - Focus on Justification and Impact Analysis
  - Focus on Early Validation
- ARCADIA Methodological Approaches
  - Transitions, Requirements, IV&V
- ARCADIA wrt Standards: xAF, SysML, AADL...
- Benefits of ARCADIA
Current Engineering Practices and Gaps

Engineering practices and their limits
The System Engineering « Ecosystem » Challenge: Collaboration

* RAMT: Reliability Availability Maintainability Testability

This document and any data included are the property of Thales. They cannot be reproduced, disclosed or used without the company's prior written approval.
The System Engineering « Ecosystem » Challenge: Collaboration

* RAMT: Reliability Availability Maintainability Testability

This document and any data included are the property of Thales. They cannot be reproduced, disclosed or used without the company’s prior written approval.
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
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
- ...

Diagram:

- Requirements
- Tests description
- Product Breakdown
- Change Requests
Definition and subsystem specification

Requirement validation process

IV&V : integration, verification, validation

Customer Textual Requirements

System Requirements

Un-formalised Design

Product Breakdown

Sub-Contractor Textual Requirements

Subsystem Design

Tests description

Coherency?

Requirements

IV&V Management

Test Campaign

Test Results

Text Req

Text Req

Test Case

Test Case

Text Req

Test Case

Test Case

This document and any data included are the property of Thales "Company" S.A. They cannot be reproduced, disclosed or used without the company's prior written approval.
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
ARCADIA Goals and Action Means

Solving the walls issue
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
One Need Definition for all

Specialities know-how confronted to architecture

Need & Architecture driving IVVQ

One global method, adaptable/adapted to each domain

Specialty engineering: safety, perf, security, …
Seven Cardinal Virtues in System Engineering

① Bad understanding of operational use

② Incoherent reference & decisions between engineering specialties

③ Poor continuity between engineering levels

④ Late discovery of definition & architecture problems

⑤ Underestimated Architecture impact / benefit

⑥ No anticipation of IVVQ, no functional mastering

⑦ 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
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
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**
   **SPECIALTY TOOL** CAPELLA
From Requirements Allocation to Architecture Mastering

« Requirement to Box allocation »

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

Early validation of the architecture
Mastering and optimisation of product and IV&V
The Long Way to Master Complexity...

... while major challenges and benefits lie here

Many focus too much on this...

and here

Adaptation to Real Life

Domain-Specific Know-How

Methods, Processes

Common Solution

Flexibility

Usability

Cost

Security

Perf.

Tools Integration

Tools

Formalism, Language

Tools

Tools

Tools

Tools

Tools

Complex Systems Design & Management 2011
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
An Example of Modelling & Early Validation: In-Flight Entertainment System

Playing videos on demand
Listening to music
Surfing the web
Gaming...
Operational Analysis: Example

What the users of the system need to accomplish

Actors & Users
Activities, missions, capabilities
Dynamic behaviour
What the system has to accomplish for the Users

Actors & System Functional dataflow interfaces
Dynamic behaviour

Modes & States
Dynamic Behaviour
Scenarios

System function
Data exchange
Actor/user
How the system will work so as to fulfil expectations

Functional allocation to components
First trade-off analysis (see multi-viewpoints)
How the system will be developed and built

Software Vs Hardware allocation
Functional allocation check
Interface definition/justification
Trade-off analysis (see multi-viewpoints)
What is expected from each designer / sub-contractor

- Requirements
- Interfaces
- Operational use case scenarios
- Expected behaviour & functional dataflow
- Allocated non-functional constraints (latency, criticality…)
- Allowed implementation resources
  
  "Sub-Contract"

EPBS: End Product Breakdown Structure
Self-Protection System Example (Bird Eye)

Operational Analysis

Data & Interface Analysis

Logical & Physical Architectures

System Analysis

June 2010:
40 op. act.
150 functions
400 components
130 diagrams

4 men.months
Exampe of Physical Architecture Overview
ARCADIA Concepts

Introduction to Arcadia engineering steps
Operational Analysis Model

System Functional and Non-Functional Need Model

Logical Architecture Model

Physical Architecture Model

Product Breakdown
Behaviour Description Means

Operational Analysis

System Functional & NF Need

Logical Architecture

Physical Architecture

Product Breakdown

Dataflow: Functions, op. activities, interactions & exchanges

Functional chains, Operational processes through functions & op. activities

Data model: Dataflow & scenario contents; Definition & justification of interfaces

Scenarios: Actors, system, components, interactions & exchanges

Modes & states: Of actors, system, components
Operational Analysis

System Functional & NF Need

Logical Architecture

Physical Architecture

Product Breakdown

Dataflow: Functions, op. activities interactions & exchanges

Component wiring: All kinds of components

Breakdown: Of all concepts

Allocation:
- op. activities to actors,
- functions to components,
- behav. Components to impl. Components
- dataflows to interfaces,
- elements to configuration items
The Engineering Triptych, ‘Three Legs Better Than One’

Requirements Management
- User operational Requirements
- Customer/Stakeholders Requirements
- System/SW Requirements
- Components Requirements
- Component Requirements

Need Analysis & Modelling
- Operational Scenarios
- System/SW Scenarios
- System/SW Scenarios
- Component IVVQ Scenarios
- Component IVVQ Scenarios

Architecture Building
- Operational Capabilities & Activities
- System/SW need Functions
- System/SW design Functions
- System/SW design Functions
- Component allocated Functions

Traceability/allocation
Traceability/justification
Allocation

Operational Analysis
Funct. Need Analysis
Logical Architecture
Physical Architecture
Product Breakdown
EPBS – End Product Breakdown Structure

Capella

This document and any data included are the property of Thales. They cannot be reproduced, disclosed or used without the company's prior written approval.
ARCADIA Concepts

Capella workbench, a tooled supporting the method
Capella solves some of the weaknesses of COTS* or OSS*:

- **ARCADIA Method support & guidance** for modelling
  - Method Steps, encyclopaedia, rules, diagrams...

- **User-oriented** Semantics
  - Engineer concepts rather than abstract/profiled language

- **Support for « modelling in the large »**
  - Performance on large models, ergonomics, modelling aids...

- **Support for viewpoint extensions**, modelling & analysis
  - Model extensions, diagrams extensions, viewpoint management...

- « Semantic » Import/export capabilities (excel, SysML, AF, ...)

- **Yet ARCADIA is also deployed using other tools**
  - Excel/Access, Rhapsody, System Architect/DoDAF...
  - with reduced capabilities, however

---

* COTS: Commercial Off The Shelf
* OSS: Open Source Software
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 Operational Activities

Define the Interactions between Operational Activities

Define Operational Entities

Allocate Operational Activities to Operational Entities

Operational Analysis

Define Operational Capabilities

Define Operational Entity Scenarios

Define Operational Activity Scenarios

Define Operational Processes

Operational Analysis Workflow and Main Diagrams

Define Operational Processes
ARCADIA Concepts

System Need Analysis
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’
- **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

1. Define System Capabilities
2. Define Functional Chains
3. Define Scenarios
4. Define the States and Modes
5. Transition from Operational Analysis
6. Refine System Functions
7. Describe Function Data Flows
8. Assign Functions To System and Actors
9. Define the Data Model

Define the System Analysis Workflow and Main Diagrams.

Assign Functions To System and Actors

Define System Capabilities

Define Functional Chains

Define Scenarios

Define the States and Modes

Transition from Operational Analysis

Refine System Functions

Describe Function Data Flows

Define the Data Model

System Analysis

This document and any data included are the property of Thales. They cannot be reproduced or communicated without the company's prior written approval.
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

- Define Logical Components
- Transition from System Analysis
- Describe Logical Data Flows
- Refine Logical Functions
- Refine Functional Chains
- Assign Functions To Components
- Describe Data Flow Scenarios
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 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 ...
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

- Describe Data Flow Scenarios
- Transition from Logical Architecture
- Refine Physical Functions
- Physical Architecture
- Assign Functions To Components
- Describe Physical Data Flows
- Define Physical Components
- Refine Functional Chains

This document and any data included are the property of Thales. They cannot be reproduced, disclosed or used without the company’s prior written approval.
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 Concepts

Focus on Functional Analysis, different engineering approaches
**Top-down functional breakdown:**
- Describe need as a few interrelated high level functions
- Refine each function, and associated exchanges
- Hierarchical, recursive approach

**Bottom-up requirement-driven:**
- Often used to formalise textual requirements
- “translate” each requirement into elementary functions, exchanges, and constraints on them
- Then synthesise higher level views by grouping elementary (leaf) functions into mother functions
- And synthesise exchanges by grouping them into categories
Use case - driven:

- Start building scenarios to illustrate system use & external interactions (or components and internal interactions)
- Then define functions at each end of exchanges
- Then enter a bottom-up approach to synthesise functions

Operational analysis – driven:

- For each operational activity or interaction, consider system and actor functions to support it
Different Ways to Build a Functional Analysis: Examples

Functional Chain – driven:
- Define major system traversal expectations
- Deduce associated functional chains, populate them with functions
- Enrich functional analysis, reusing and linking existing functions
- Then enter a bottom-up approach to synthesise functions

Several ways could (should?) be mixed and interleaved
- Other ways can be considered as well

All ways should converge to the same final model contents, no matter how the building steps were ordered
Architecture Frameworks / IDEF0 -like?

- Uses of a same function in different diagrams may differ or oppose each other (e.g., Enterprise Architect)
- No explicit definition of inputs and outputs independently from diagrams
- Poorly adapted to Reuse of functions & Use case consolidation

➡ Definition of in/out ports on functions, to express “direction for use” of the function
➡ Function/Ports definition is shared among all uses & diagrams
➡ Functional exchanges are also shared
SysML blocks concept -like?

😍 Huge work to manage & update delegation of ports
😍 No direct link between leaf functions
😍 Not adapted to bottom-up approaches and model evolution

➡️ Any function can be linked to any other
➡️ Functional exchanges of the parent are just moved (drag & drop) towards the relevant child function in charge of managing the exchange
➡️ Automated graphical synthesis at parent level
Activity diagrams?

- Mix data flow and sequence flow
- Same limits as IDEF0
- Poorly adapted to Architecture Definition (no control flow in interfaces!)
- Control flow depends on context, while dataflow is intangible

- Restriction to “Dataflow” concept and diagrams:
  - Dependencies between functions, as expressed by (oriented) functional exchanges linking their ports,
  - Nature of data, information, signals, and flows... exchanged between functions specified on exchanges.
  - No pure sequence flow if no data is exchanged
  - Scenarios (sequence diagrams) can express context-dependent ordering or precedence constraints
ARCADIA Concepts

Focus on consistency and impact analysis, interface definition and justification
Connect elements

In a coherent manner!
Routing data inside Components
ExchangeItems group data to be used together (e.g. message)

Audio profile Exchange Item:
- Sound profile
- balance
- volume

Process audio signal

Customise sound parameters

Possibly for several similar uses

Audio profile Exchange Item:
- Sound profile
- balance
- volume

Process audio signal

Coherency required!

EI1 (d11, d12)
EI2 (d21)
EI3 (d31, d32)
EI1 (d11, d12)
EI2 (d21)
EI3 (d31, d32)
EI3 (d31, d32)
Same for Components, adding Interfaces in order to group exchangeItems

Sharing Interfaces

Coherency required!
ARCADIA Concepts

Early Validation: Multi-viewpoint approach for collaborative engineering and non-functional analysis
The Product Architecture must deal with potentially contradictory Constraints, which impact Breakdown:

- Safety
- Performances
- Complexity of internal interfaces
- Ease of System Integration
- Cost, sub-contracting
- ...

→ Building an appropriate Architecture means finding the **most acceptable Compromise** between these **Viewpoints**

Then a detailed design & check according to each viewpoint is required

→ Fine-grain Analysis per viewpoint must
- **start from architecture model**, and
- update/validate first hypotheses
Early Validation: Specialities Know-how Confronted to Architecture

Specialty engineering: safety, perf, security, ...

Sub-contractors

Multi-viewpoint trade-off analysis (see ISO 42010 standard)
Architecture Definition Lifecycle Process

Operational Need Analysis

System Functional & NF Analysis

Simulation Means

Architecture Design (Logical, Physical)

Check

Dedicated Analysis Means (RAMST...)

IVVQ Means

Architecture Early Validation (resource cons, perf, fault propagation...)

Close the Loops!

Global Architecture Design & Coherency Mastering
Support of multi-viewpoint Trade-off Analysis

- Express N-F constraints & Need inside operational/functional need models

- Capture domain know-how on common architecture for each Viewpoint:
  - Dedicated concepts (model extensions)
  - Architecture checking rules & algorithms
  - Dedicated diagrams and graphical annotations

- Analyse each candidate architecture against all viewpoints, locate defects and correct

- Quickly Iterate

Supported in Capella
1. Automatic analysis
- CPU overloaded
- Bus overloaded
- Latency too high

2. Causal analysis
- Tool locates problems
- Quantitative analysis

3. Architecture improvement
- Lighter protocols
- Higher bandwidth
- Hardware processing

Require additional Capella viewpoints
1° Automatic analysis:
- Rule: "No single source for major failure condition"
- Not met for video

2. Causal analysis
- Tool locates problems
- Failure propagation algorithm

3. Architecture improvement:
- Second redundant server

Require additional Capella viewpoints
Confrontation rules for multi-viewpoints trade-off

Histogram view will be available in Capella in 2015 (see Kitalpha)
Other example of Modelling & Validation:
On-board Electrical Power System

Energy & Thermal system
Of a commercial Aircraft
Three Interleaved, Multi-Physics Models

**Power Model**
- Generation
- Distribution

**Thermal Model**
- Conditioning
- Pressurisation
- Equipments Cooling

**Command & Control Model**
(coming soon)
Power Systems Example: Dedicated Viewpoints

Power & Thermal performance
depending on flight phase consumption, incl. Overloaded components
detection based on power computation, linked to thermal Model

Safety / Integrity
incl. Failure containment, redundancy rules & analysis,
failure scenarios & propagation, monitoring efficiency, shielding...

Reliability & Availability
incl. Reliability computing, reconfiguration issues, flight delay...

Spatial (3D) arrangement
Early identification of spatial arrangement constraints impacting
the architecture

And also: Mass, Cost, Reliability...
"Model Once, Use Many": The Blue Line Vision

One model, many users...

...and many other uses...

Engineering documents: SSS, IRS, SSDD, ICD, SRS, ...

User & Maintenance Documents

Electronic Interface & data generation

Test bench specification

Metrics, Risk Management

Wiring, Development & Production Means...

3D Computer Aided Design

...and more
ARCADIA Methodological Approaches

Transition, relationship to requirements, IV&V
Support of Consistency Between Engineering Levels

- **Automated Transition between Engineering Levels**
  - Iterative, conservative
  - Coherency control
- **Mastering complexity through multiple abstraction levels**
Recursive Application to Each Engineering Level

### Level n

1. Operational Need Analysis
2. System/SW Need Analysis
3.1 Logical Architecture
3.2 Physical Architecture
4 - Contracts & IVVQ Definition

### Level n+1

1. Operational Need Analysis
2. System/SW Need Analysis
3.1 Logical Architecture
3.2 Physical Architecture
4 - Contracts & IVVQ Definition

**Component Integration Contract**

**Constraints**

**Need Analysis restricted to component**

**Component Situation in Level n Physical Architecture**

**Breakdown**

- Systems of Systems
- Complex Systems
- Sub-systems
- Equipment
- Sub-assemblies & Platforms

**Process**
Example of progressive building

1. At bid time to sketch price:
   - Quick functional analysis
   - First sizing of I/O needs

2. At bid time to secure price:
   - Coarse-grain architecture to evaluate risk, reuse opportunities

3. At product line level to find competitive advantage:
   - Operational & capability analysis to enrich product

4. At design time to secure Bid architecture:
   - Functional to components mapping
   - Multi-viewpoint analysis (safety, perf., IVV...)
   - Check with operational need

5. At detailed design time:
   - Completion & link of models where risky
   - Fine grain analysis

6. At development time:
   - Generation of interface files & wiring data
   - Allocation of resources to components...
New model-based engineering approaches such as ARCADIA seek (among others) to overcome textual requirements limitations

The need is formalized in a shareable form, that can easily be analyzed and validated
- Operational and functional analysis
- Traceability links with textual requirements

Textual requirements are complemented (and not replaced) and validated by models and their use

The solution is formalized, traced and justified (partially validated) by the model architecture

Previous traceability links are now based on a unifying model and an explicit and verifiable process that secures engineering
For Customers who require it, textual requirements are still the main vector

- For the Customer, Functional description is an explanation and / or additional support deepening specifications

- Traceability towards engineering artifacts (architecture, tests ...) is still ensured by User Requirements (UR)

- It is made internally through the model: requirements <--> model <--> artifacts

- System Requirements (SR) add to the UR only those requirements that are strictly necessary to communicate and validate the need

- Based on the model as a negotiation support
Internally, models carry most of the description of need and solution

- Anything that can be efficiently expressed in the model is formalized that way ("modeled requirements")
  - In this case, it is unnecessary to create or refine textual requirements (would be redundant)
  - Internal requirements (textual) are added where necessary:
    - Either to express a constraint or an expectation, more precisely than the model,
    - Or if it is difficult to represent and capture a specific need in the model

- The customer requirements (UR) remain traced in the model and towards engineering artifacts, for justification purposes

- Engineering, subcontracting and IVVQ are driven by the model
  - (To be adjusted according to the maturity of subcontractors)

- A posteriori check of coverage / satisfaction of these requirements is done
Comparing Approaches for Requirement Engineering

Textual Requirements driven

- Customer Textual Requirements
- System Requirements
- Sub-Contractor Textual Requirements
- Subsystem Design
- Un-formalised Design

Tests description

Coherency?

Textual Reqs + Model driven

- System Model
- Solution
- Product Breakdown
- Subsystem Model
- Need
- Solution
- Sub-Contractor Textual Requirements

(Derived, reconstructed link)

This document and any data included are the property of Thales "Company" S.A. They cannot be reproduced, disclosed or used without the company's prior written approval.
Test campaigns are constructed from model scenarios and functional chains

- Refinement (detailed scenarios, not nominal ...), complements, details (ranges, expected results ...)
- Traceability links
  - Between test campaigns and delivery versions
  - Between tests and scenarios / functional chains

Validation of textual customer requirements is achieved by exploiting the indirect links

- Tests - scenarios / CF - functions – requirements
Requirements

Model

IVV Management

(Derived, reconstructed link)
Using ARCADIA Engineering Models to Drive IV&V

Define IVV Strategy

Focus on Functional Content and Architecture

Master Development Ups and Downs

Control Maturity of Deliveries

Optimize IVVQ Globally (incl. Enabling Systems / Test Means)

Operational Need, Functional Contents

System Components

Mission System
Radar
Receiver
Software/HW

Test Benches
ARCADIA wrt Standards: xAF, SysML, AADL…

Yet another Formalism?
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 goes beyond Formalisms & Languages:

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

AADL     SysML, UML2     MARTE

- Operational Analysis
- System Need Analysis
- Logical Architecture
- Physical Architecture
- EPBS
- Transition to sub-system, SW, HW

Common or Similar Concepts:
- Blocks, parts, (sub)components
- Data, constraint properties
- Ports, flows
- States & modes
- Profiles & libraries...
- Bus, connection
- Processors, Memory... (extension) → AADL
- Threads, processes... (extension)

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 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
Early Validation: How to Validate Architecture vs Need

Need consistency & coherency
Traceability links

Impact analysis:
Architecture vs operational & functional need confrontation
Traceability & implementation links

Architecture Vs non-functional need confrontation
Viewpoint analysis traceability & implementation links

Requirements
System
Functional & NF Need
Model
Logical Architecture
Model
Physical Architecture
Model
Operational Analysis
Model
Product Breakdown

Early Validation: How to Validate Architecture vs Need

Need consistency & coherency
Traceability links

Impact analysis:
Architecture vs operational & functional need confrontation
Traceability & implementation links

Architecture Vs non-functional need confrontation
Viewpoint analysis traceability & implementation links
**Need & Architecture Driving Integration Verification Validation**

- **Definition of test & IV&V Scenarios**
- **Deliveries definition based on operational & functional capabilities**
- **Definition of test benches & simulation needs**
- **Impact analysis**
  - Need evolution
  - Component delivery delay
  - Integration trouble
- **Non-regression optimisation & mastering**

---

This document and any data included are the property of Thales. They cannot be reproduced, disclosed or used without the company's prior written approval.
Summary of Expected Benefits

**Definition & Design**
- Better adequacy to need
- Less rework on architecture
- Capitalisation and reuse

**IVVQ**
- Less engineering change requests
- Less problems / change reports
- Less rework

**Key Terms**
- SSS
- PIDS
- ICD
- SSDD
- IRS
- SRS
- ECR
- PCR

This document and any data included are the property of Thales. They cannot be reproduced, disclosed or used without the company's prior written approval.
Summary of Contribution to Expected Benefits

**Better adequacy to need**
- Thanks to: Customer Need Analysis justifying Architecture

**Less rework on architecture**
- Thanks to: Early Check of Architecture, non-functional Viewpoints modelling

**Capitalisation and reuse**
- Thanks to: Formalised Architecture, Viewpoints & Rules

**Less engineering change requests**
- Thanks to: Early Check against Operational Need

**Less problems / change reports**
- Thanks to: Integration Contract + Architecture designed for IVVQ

**Better development management**
- Thanks to: Integration Contract Formalisation
Different purposes & steps in ARCADIA deployment

Check your own priorities and select!

Then think about full-scale

ARCADIA full scope

Increasing ROI & Engineering Strength

This document and any data included are the property of Thales "Company" S.A. They cannot be reproduced, disclosed or used without the company’s prior written approval.
Some ARCADIA Golden Rules

- Do not model anything if you don’t know for what purpose
  - Build models according to the way you will exploit them
  - And according to users / addressees

- Do not model in details if you are not able to keep the model up to date
  - Adjust stopping criteria accordingly

- If you want quick Return on Invest, keep focussed on your major problems / challenges first

- Favor modelling for several usages - “Model once, use many”
  - In order to maximise ROI and motivate for maintenance