ARCADIA ENGINEERING LANDSCAPE

Introducing Arcadia Scope

Jean-Luc Voirin

Table of Contents

1 Scope of this document

ARCADIA is a tooled method devoted to systems & architecture engineering, supported by Capella modelling tool.

It describes the detailed reasoning to

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

It can be applied to complex systems, equipment, software or hardware architecture definition, especially those dealing with strong constraints to be reconciled (cost, performance, safety, security, reuse, consumption, weight…).

It is intended to be used by most stakeholders in system/product/software or hardware definition and IVVQ as their common engineering reference and collaboration support.

ARCADIA stands for ARChitecture Analysis and Design Integrated Approach.

This document is a brief introduction to Engineering Concerns addressed by Arcadia, describing what features and services can be expected from Arcadia in the field of Engineering.

It could be considered both as a very quick introduction to key aspects of engineering,

and as a short set of user needs & expectations for Arcadia.

2 Arcadia Reference Documents

An in-depth introduction and description of Arcadia, with explanations on the method, on the language, illustrated by detailed examples of application, can be found in the Arcadia reference book:

> *Jean-Luc Voirin, 'Model-based System and Architecture Engineering with the Arcadia Method', ISTE Press, London & Elsevier, Oxford, 2017*

A presentation of Arcadia main principles and concepts can be found in the following online documents, including this one:

- Arcadia Engineering Landscape: an introduction to Engineering as supported by Arcadia
- Arcadia User Guide: a first level description of Arcadia approach and main engineering Tasks
- Arcadia Reference - Activities: an in-depth description of Arcadia tasks and activities
- Arcadia Reference - Data Model: data created and exploited by these activities
- Arcadia Reference - Capabilities: main processes supporting engineering
- Arcadia Language - MetaModel: a more formal description of Arcadia language concepts
- Arcadia Q&A: real life questions and answers on deploying Arcadia

See table 'Summary of reference Documents Contents' next page.

For easier navigation capabilities (including in diagrams, between activities and data, etc.), a web version can be browsed here.

Advanced practitioners in modelling and Arcadia can also access the Arcadia-compliant Capella model of Arcadia, from which this material is automatically extracted, here.

Summary of reference Documents Contents		Book	Landscape	User Guide	Reference - Activities	Reference - DataModel	Reference - Capabilities	Language - MetaModel	Q&A
History	Why was the method created and tooled? For which purpose? With which benefits?	✓							
Philosophy	What are its objectives and expected scope? What are its specificities?	✓	(✓)						
	How does it address Engineering Issues and Challenges?	✓		(✓)			(✓)		
	What kind of major levers does it use to address them?	✓		(✓)			(✓)		
Principles and approach	What are the drivers of each core perspective ….? How to build each of them?	✓		(✓)	(✓)				
	How to address Major engineering Issues using Arcadia and these perspectives?	✓					✓		
Details for implementation	What are the detailed processes to build each of the core perspectives?	(✓)			✓				
	How and where are engineering data elaborated and used to address major engineering challenges?	✓			(✓)	(✓)	✓		
	What is the formal definition of the Arcadia language & concepts?	✓				(✓)		✓	
	Examples and samples of models?	✓							
Hints for Deployment	Which major questions arise in projects applying Arcadia?								✓

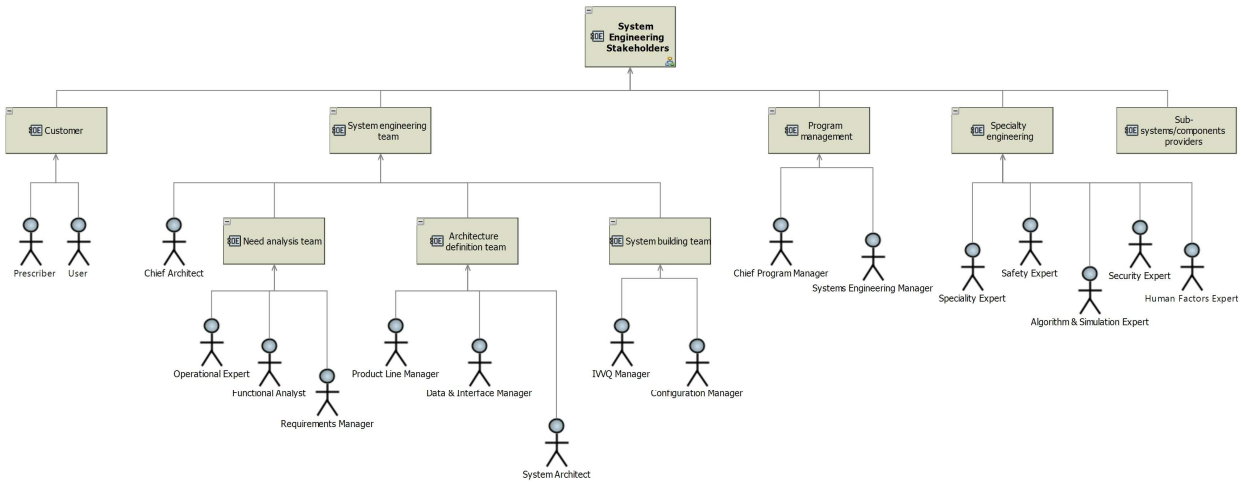
✓ : fully detailled (✓) : simplified or partial

3 Who is interested in applying Arcadia?

This is a description of some key Stakeholders contributing to systems engineering, as addressed by Arcadia.

Warning: the definition of stakeholders is given only as an example, each engineering organisation having its own roles, activities and workshare. furthermore, this description is by no means exhaustive; it is just a sample of some stakeholders that may take benefit from Arcadia approach and outputs.

Note : this description may apply to other levels of (system, software, hardware…) engineering for the architecture definition and verification part, to some extent; it has to be adapted to each discipline and domain according to their own context.



Below is a list of engineering tasks that each stakeholder is expected to perform; this list is not exhaustive, but each of these tasks could take benefit from applying Arcadia.

Customer

Prescriber

- Express end users Need
- Find the best ownership cost of required capabilities
- Deliver external constraints on solution (incl. interfaces etc)
- Express Deliveries content expectations
- Check compliance of foreseen solution against Need
- Check compliance of final solution against Need

User

- Provide feedback

System engineering team

Chief Architect

- Understand the customer & users need to addess
- Check Need consistency & completeness
- Check feasibility & cost of Requirements incl. impact on design & IVVQ
- Find most structuring/constraining requirements for operational purpose
- Evaluate contribution of system functions etc in user need (value analysis)
- Define system architecture(s) in order to confine/manage complexity
- Define functional & non-functional expectations
- Find best trade-off between requirements & constraints (incl. non-functional)
- Early define & validate properties of the architecture wrt non-functional constaints
- Master & minimise Engineering complexity
- Define an architectural frame & constraints to master components development & integration
- Define contractual requirements for components
- define EPBS (end Product Breakdown Structure)

Need analysis team

Operational Expert

- Define & Validate Users goals, tasks, scenarios & needs

Functional Analyst

- Define & Validate services expected from the system

Requirements Manager

- Define & Validate requirements, their consistency and completeness

Architecture definition team

Product Line Manager

- Characterise market & customers segmentation, define product offer
- Confront a customer need to product offer & reference configurations
- Define project configuration & contents
- Adapt architecture to reuse and product policy
- Define product line variability
- Define reference configurations

Data & Interface Manager

- Ensure coherency & consistency of exchanged data
- Allocate external interfaces to components
- Define internal interfaces between components
- Check interface consistency for reused components

- Justify interface definition
- Define wirings & assemblies
- Negotiate with suppliers interface definition

System Architect

- Act as delegated from Chief Architect

System building team

IVVQ Manager

- Ensure realism & relevance of IVVQ scenarios & tests
- Define IVVQ strategy & deliveries
- Optimise non-regression tests
- Define test benches & simulation means
- Analyse impact of delivery delay

Configuration Manager

- Define baselines contents
- Manage changes

Sub-systems/components providers

- Get specifications of components to develop
- Demonstrate adequacy of component design to specifications
- Deliver validated components for integration

Specialty engineering

- Activities from Common Role: Non functional constraints engineering

Speciality Expert

- Activities from Common Role: Non functional constraints engineering

Safety Expert

- Activities from Common Role: Non functional constraints engineering

Algorithm & Simulation Expert

- Define realistic use scenarios & environment
- Define realistic system behaviour model
- Deliver quantitative & qualitative efficiency results

Security Expert

- Activities from Common Role: Non functional constraints engineering

Human Factors Expert

- Activities from Common Role: Non functional constraints engineering

Program management

Chief Program Manager

- Capitalise Know-how & Outcome
- Build engineering management plan and monitor its progress
- Master Cost & Schedule

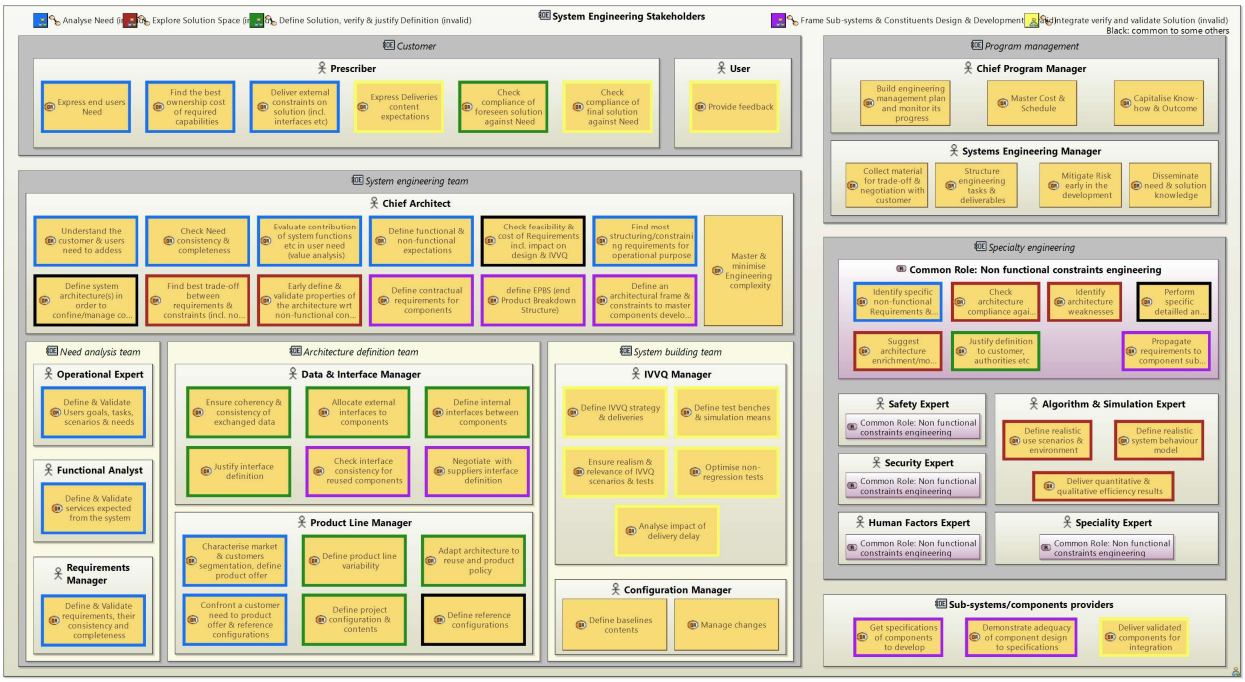
Systems Engineering Manager

- Collect material for trade-off & negotiation with customer
- Structure engineering tasks & deliverables
- Mitigate Risk early in the development
- Disseminate need & solution knowledge

Common Role: Non functional constraints engineering (Generic Role)

- Identify specific non-functional Requirements & constraints
- Suggest architecture enrichment/modification
- Justify definition to customer, authorities etc
- Check architecture compliance against specific requirements
- Propagate requirements to component sub-contractors
- Identify architecture weaknesses
- Perform specific detailled analysis

Engineering Stakeholders and tasks are summarised in the figure below.

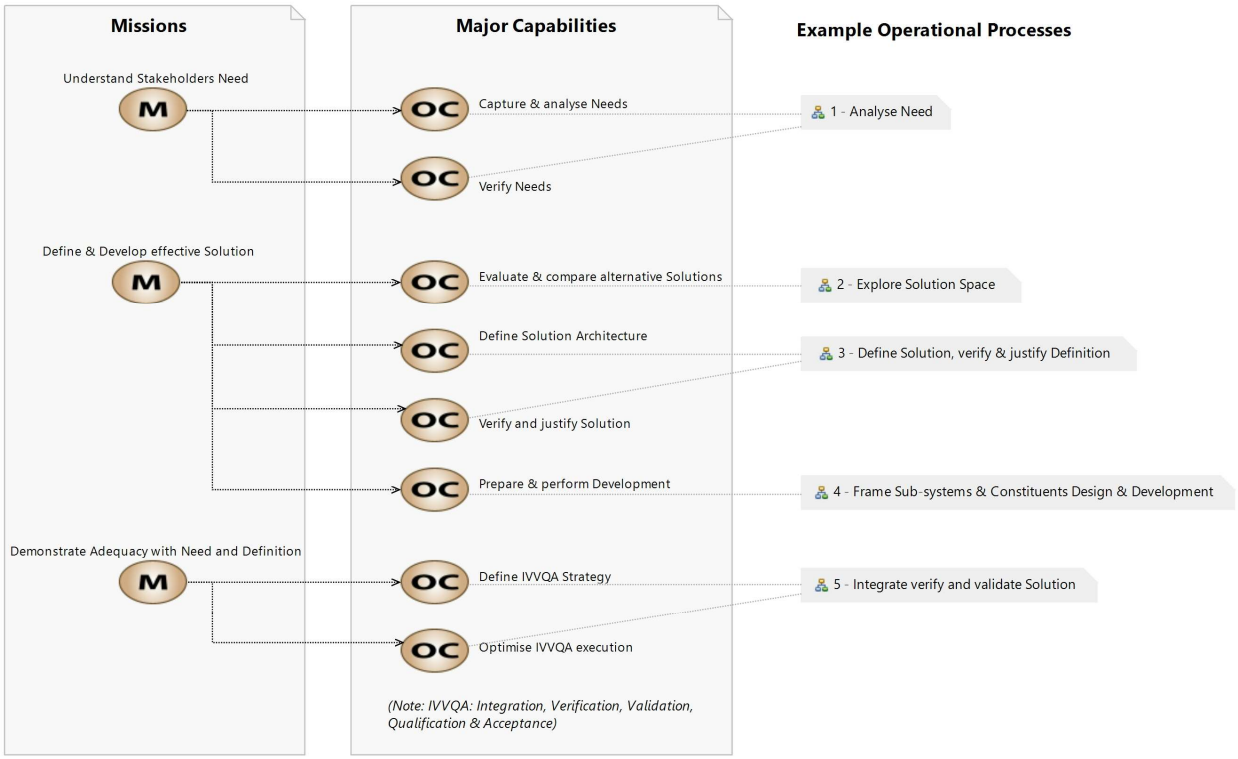


4 Which Engineering Purposes are considered?

This diagram shows key objectives (missions (M) and operational capabilities (OC)) of engineering stakeholders in the scope of Arcadia.

A few operational processes (on the right) illustrate how stakeholders are involved in each process through their own tasks.

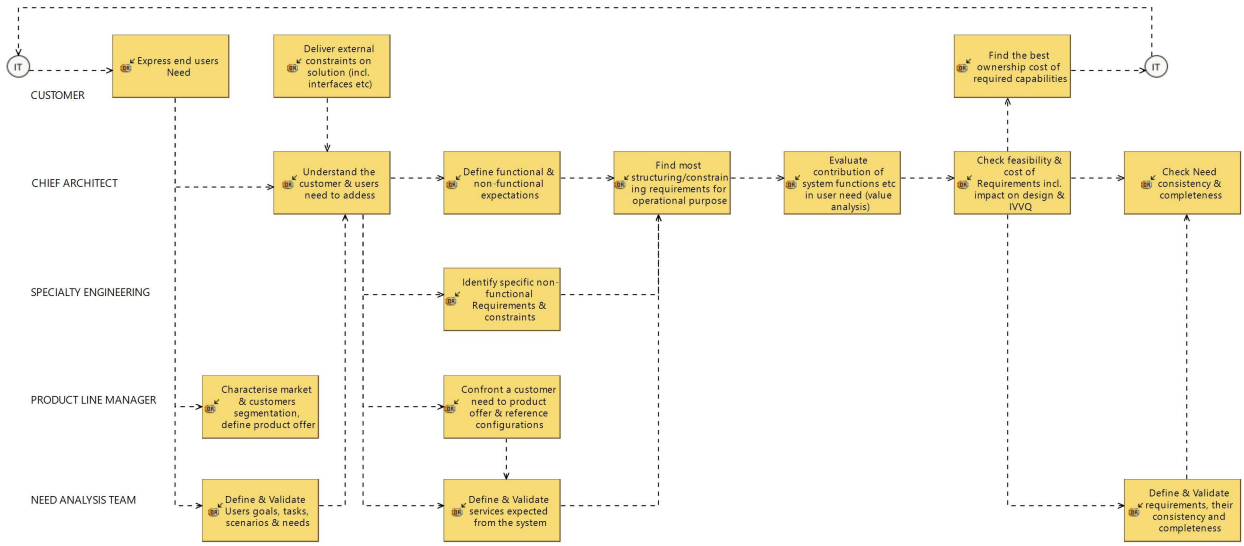
For illustration purpose only.



The former engineering processes are sketched below, to illustrate how stakeholders tasks can contribute to each capability.

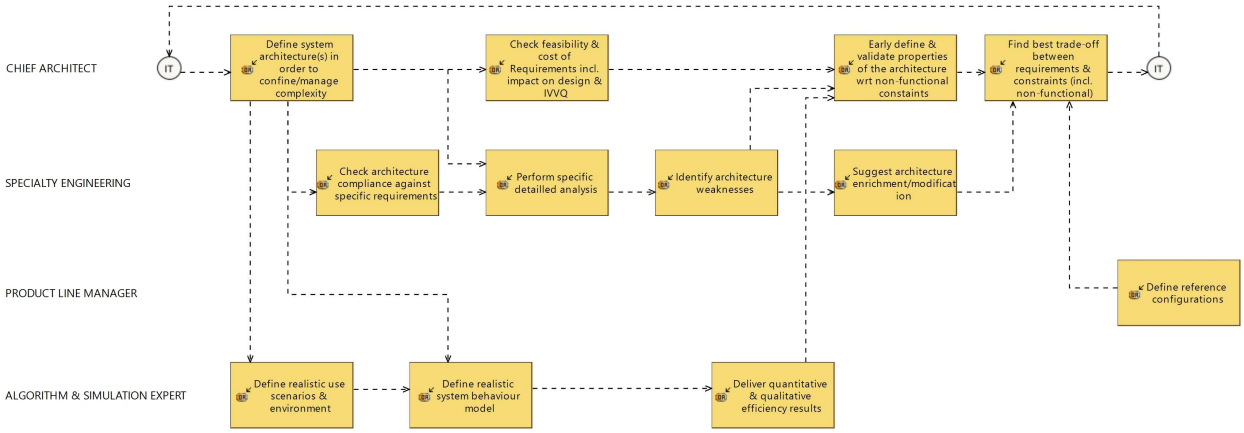
1 - Analyse Need

This process illustrates the collaborative work between engineering stakeholders in order to capture, understand, analyse the need of end users (and beyond), in order to build a verified and agreed representation of needs.



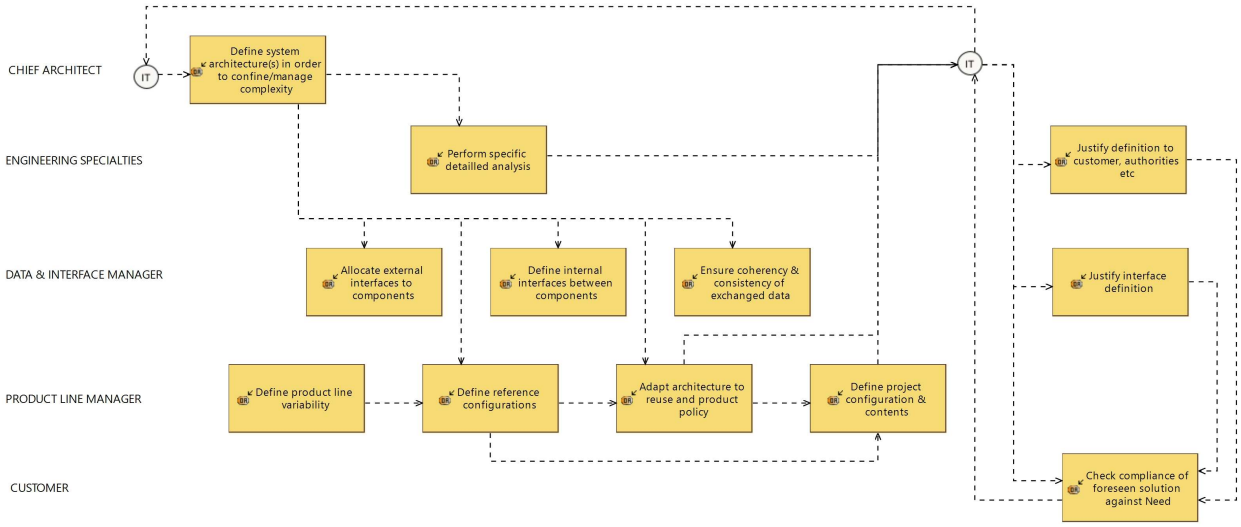
2 - Explore Solution Space

This process illustrates the collaborative work between engineering stakeholders in order to elicit and elaborate solution alternatives, analyse each of them to compare and identify best alternatives, as a basis for further solution definition.



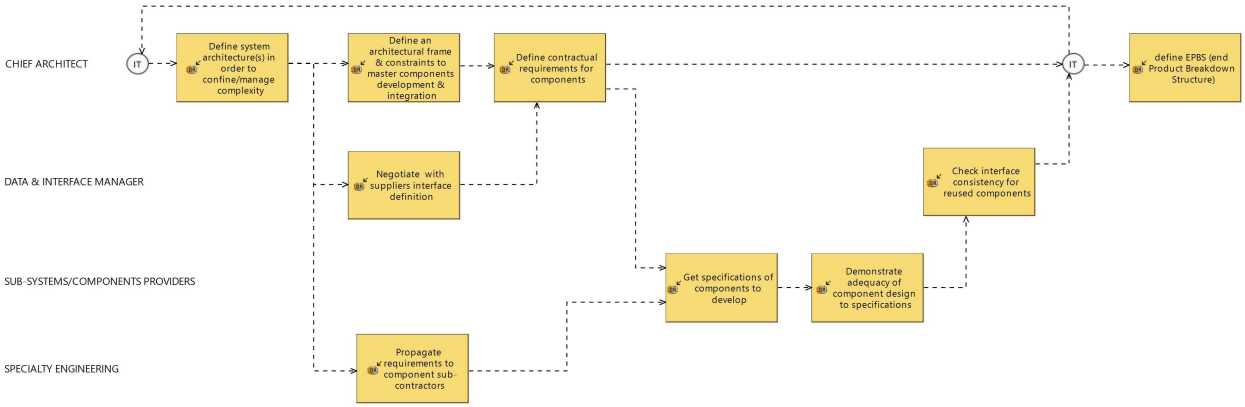
3 - Define Solution, verify & justify Definition

This process illustrates the collaborative work between engineering stakeholders in order to consolidate most relevant solution alternatives, verify and select the best compromise, in order to build a verified and agreed representation of solution for further engineering.



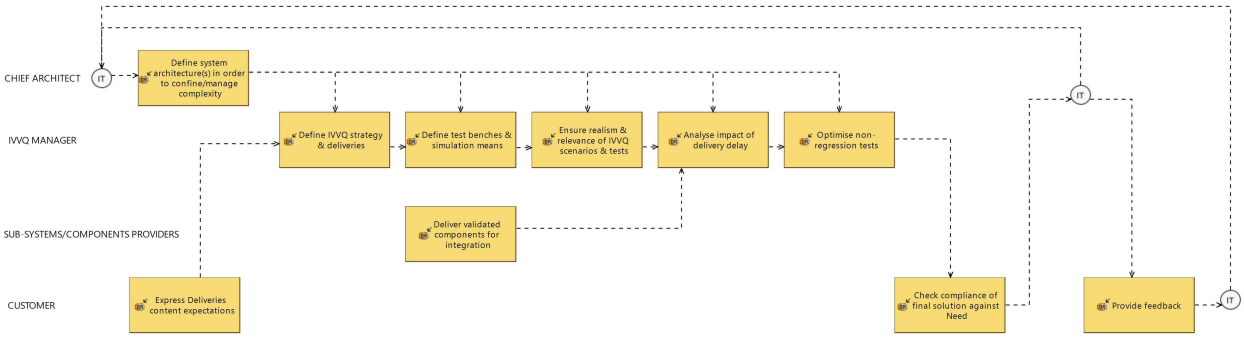
4 - Frame Sub-systems & Constituents Design & Development

This process illustrates the collaborative work between engineering stakeholders in order to reach an agreement on each sub-system or constituent contribution to the solution, as a basis to drive further lower level solution definition, design and development.



5 - Integrate verify and validate Solution

This process illustrates the collaborative work between engineering stakeholders in order to verify that the solution produced by integrating developped or purchased system constituents, is conform both to design, need and operational usage expectations.

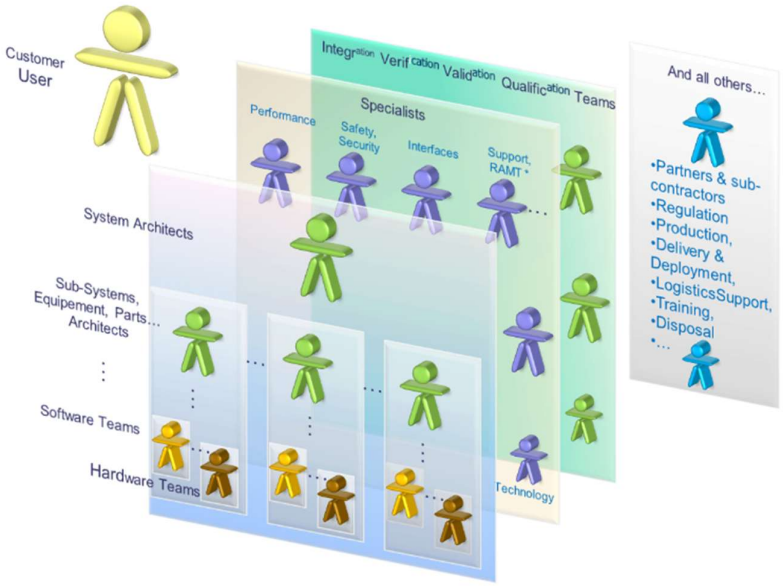


5 What kind of assistance can Arcadia provide to Engineering?

5.1 Arcadia Scope and Principles

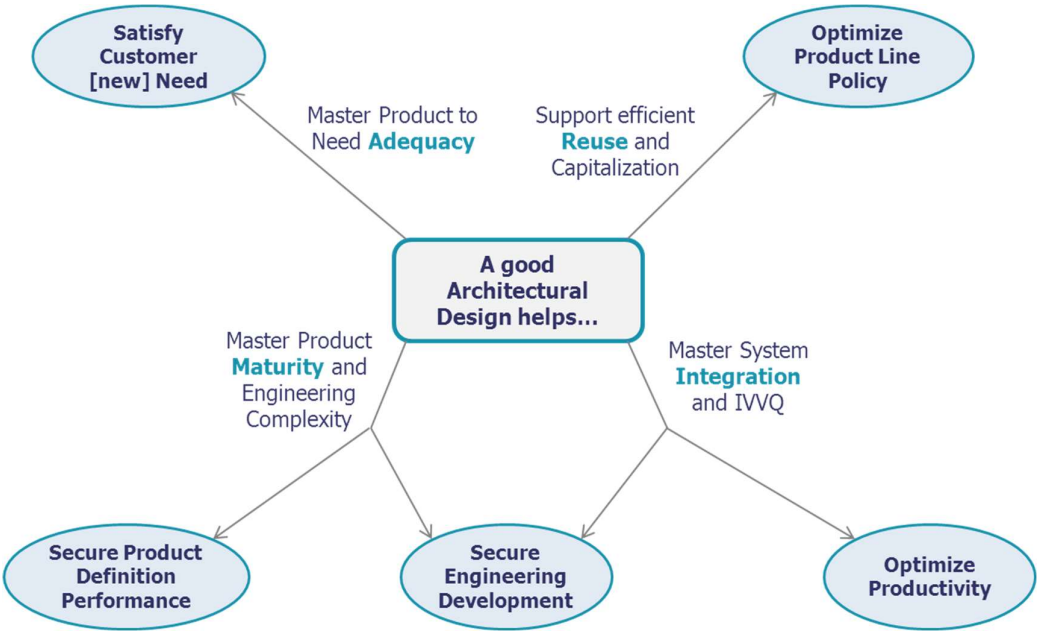
Arcadia, a model-based Architecture engineering method

Modern systems are subject to increasingly higher constraints regarding expected behavior and services, safety, security, performance, environment, human factors, etc. All these constraints are under the responsibility of different stakeholders, which need to be reconciled during the solution architectural design and development process.



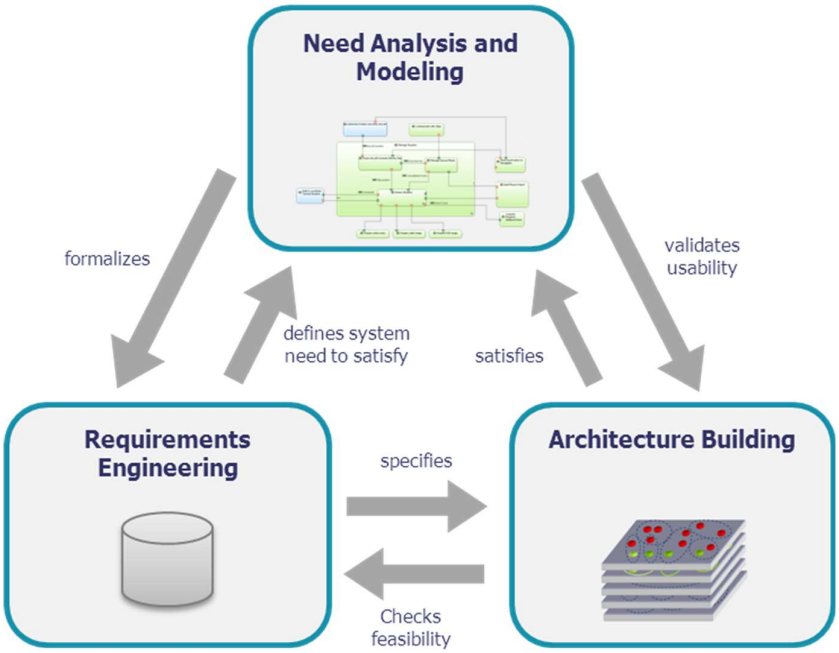
Architecture definition is a major part of engineering activities, and notably includes analyzing operational needs, structuring and decomposing the system, software, or hardware assets in order to

- Provide significant information for decision-makers and managers
- Ease the mastering of need, complexity, design and development
- Structure engineering in a well-defined, justified, technical frame
- Guide designers and developers to respect the product definition drivers.



Arcadia is a **model-based engineering method for systems, hardware and software architectural design**. It has been developed by Thales between 2005 and 2010 through an iterative process involving operational architects from all the Thales business domains. Since 2018, Arcadia is registered as Z67-140 standard

Arcadia promotes a viewpoint-driven approach (as described in ISO/IEC 42010

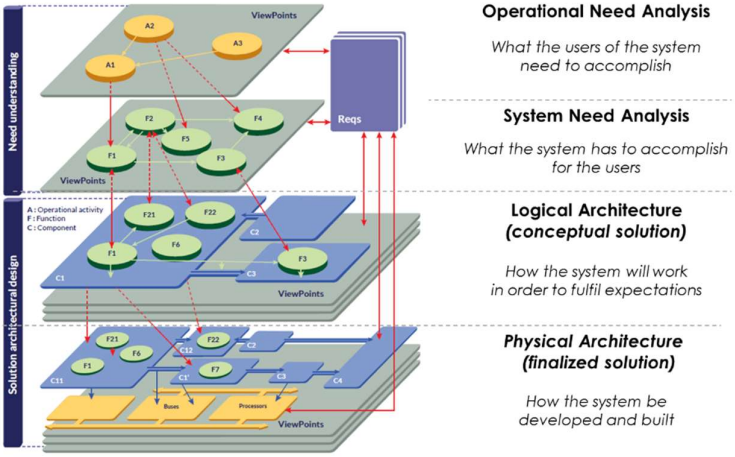


Some purposes that drove Arcadia definition were:

- Model-based and tool-supported approach and language, to address scalability and complexity. The reference modelling tool supporting Arcadia is the Capella open source solution (see https://www.eclipse.org/capella
- Supporting collaboration and co-engineering of most stakeholders (incl. customers, specialties (safety, security, RAMT etc.), disciplines (software, hardware, mechanics etc.), sub-contractors, integrators)
- Welcoming domain-specific and specialty/disciplines specificities and added value
- Adapted to multiple and diverse lifecycles, workshares, complexity and size issues.

The ARCADIA definition is driven by a few structuring principles, as depicted in the figure below and illustrated in next paragraphs:

- Extended functional analysis to define both need and solution behavior;
- Separation of need analysis and solution architecture definition;
- Separation of operational need analysis and definition of system contribution to this need (system need analysis);



- Separation of functional/behavioral description, and structural decomposition;
- Differentiation between the structuring of behavioral/logical components and physical hosting components.

This favors separation of concerns, so as to adapt to different lifecycles, provides capabilities for impact analysis, and allows efficient management of architecture alternatives, reuse, etc.

Noticeable features of Arcadia

- Models supporting enterprise-wide collaboration and co-engineering
  - An Eclipse Capella™ model is built for each Arcadia engineering phase. All of these models are articulated through model transformation, and related by justification links; they are processed as a whole for impact analysis, notably in case of required evolutions.

  - Collaboration with engineering specialties is supported by modelled engineering viewpoints to formalize constraints and to evaluate architecture adequacy with each of them
  - Collaboration with customer and subsystems engineering relies on co-engineered models (e.g. physical architecture), automatic initialization of need model for sub-systems, and impact analysis means between requirements and models of different engineering levels.
  - Integration, verification, validation and qualification (IVVQ) are driven by user capabilities, functional chains and scenarios in the model, rather than by textual requirements
  - Elaboration of product line variabilities and configurations is optimized and assisted based on operational market segmentation, commercial portfolio contents and architecture constraints/adaptations to product policy, all described in the model.
- Tailored for architectural design
  - A domain-specific language (DSL) was preferred in order to ease appropriation by all stakeholders, usually not familiar with general-purpose, generic languages such as UML or SysML.
- Dealing with complexity and size
  - Abstraction levels are in the DNA of Arcadia. Capella advanced mechanisms have been developed to mask and confine complexity, deal with model maintenance, large-scale modelling, model evolution and reuse.
- Field-proven in real industrial situations
  - Arcadia is currently applied in various domains and organizations, in many countries, on very large or small projects, by thousands of users. A continuous challenging, improvement and adaptation of both the method and its supporting workbench has favored a very fast dissemination.
- Open to domain-specific added value
- Adapted to several lifecycles and work sharing schemes

Adaptation of Arcadia to Dedicated Domains, Contexts, Etc.

Beyond the transverse, common architectural design work, each organization, in the field of its own business, constraints and know-how, should **tailor the method steps by adapting them to their own domains, products and programs**. This includes:

- Definition of a reference architecture (including architecture drivers) for each key product and software element

- Definition of appropriate viewpoints adapted to the domain, product and architecture
- Definition of complementary dedicated engineering rules
- Selection of relevant architectural patterns for the domain, product, and technologies considered
- Setting up of models, based on the reference architecture and viewpoints, and basis for simulation, early validation, automation of the design process (key for productivity gains)
- Definition of adjustment rules for each of its contexts
- Dissemination in the engineering teams (training, coaching)...

Adaptation to different Lifecycles

The recommended method described in this document takes best benefit from a **top-down approach**:

- Starting from operational and system need to define and validate requirements
- Building a "technology neutral" logical architecture dealing with non-functional constraints
- Then specifying technical functions and services of a physical architecture to implement it in the best way

Yet many constraints which need to be taken into account arise from the industrial context:

- Technical or technological limits
- Available technology, COTS
- Existing legacy to be reused
- Product policy imposing the use of given hardware boards, software components...
- Industrial constraints such as available skills, the necessity to sub-contract, and export control...

This is the reason why Arcadia can be applied according to several lifecycles and work sharing schemes. Great care has been taken in the method, the language and the Capella workbench to not impose one single engineering path (e.g. top-down) but to be adaptable to many lifecycles: **Incremental, iterative, top-down, bottom-up, middle-out**, Etc.. The method is inherently **iterative**.

Examples of iterations or non-linear courses are:

- Need analysis starting from requirements, due to a lack of operational knowledge (a kind of reverse engineering of operational need)

- Requirements analysis anticipating logical or even physical architecture, to check for feasibility by defining/confronting to an early architecture
- Logical architecture anticipating (part of) physical architecture, e.g. to check for performance issues
- Physical architecture adapting to subcontracting constraints, or built from assembling reusable, existing components
- Components contract definition iterating on physical architecture to secure integration and refine contract parameters

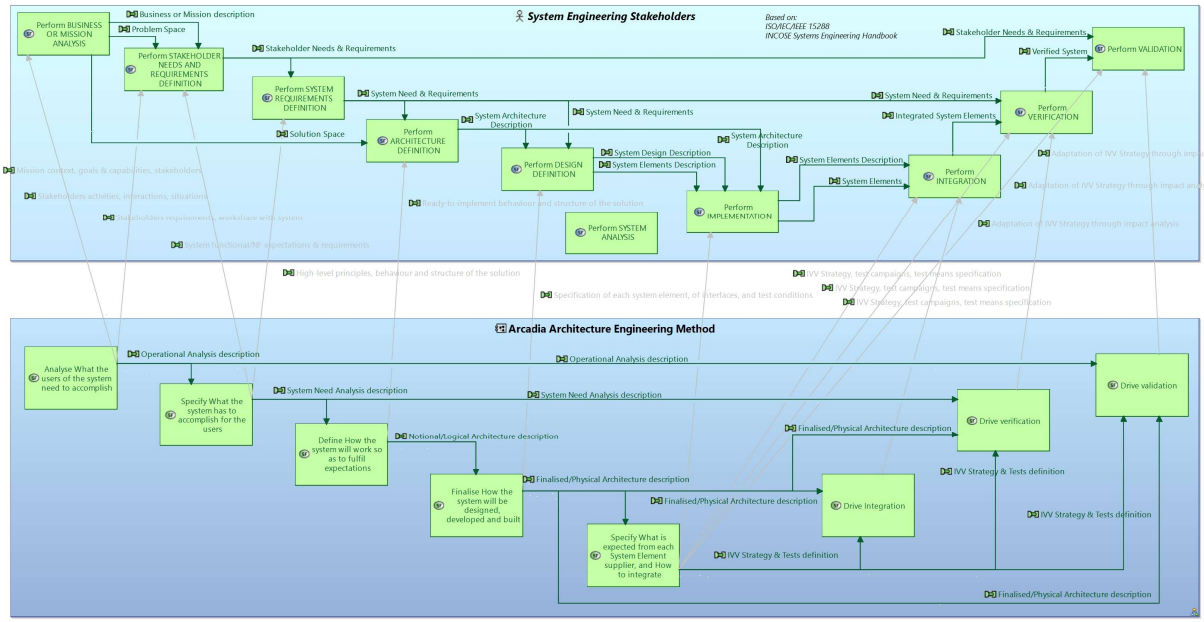
Next paragraphs give a first description of major arcadia perspectives, for a given engineering level (system, sub-system, software or hardware part…).

5.2 Arcadia core Perspectives and capabilities

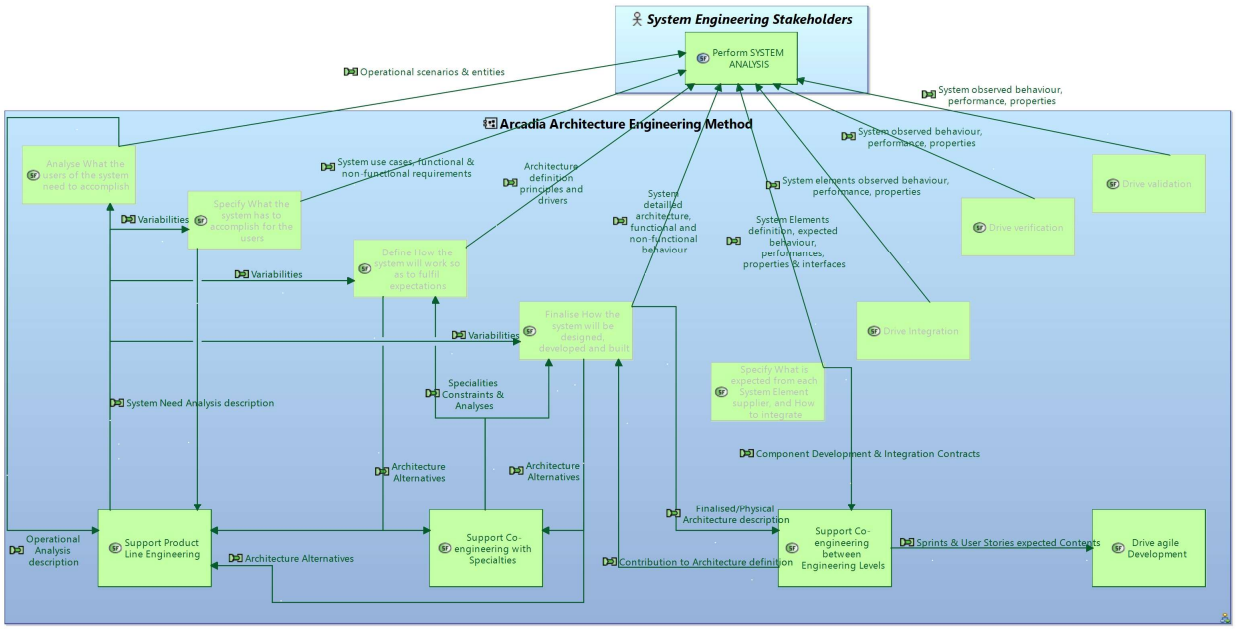
This figure defines main *partial* support that Arcadia can bring to engineering processes, through the main perspectives brought by the method to stakeholders.

Arcadia perspectives and services are described in the lower left part of the figure, while the upper part freely depicts core engineering tasks, based on the so-called Technical Processes of ISO/IEC/IEEE 15288:2015 'Systems and software engineering — System life cycle processes' standard, including their presentation in INCOSE Systems Engineering Handbook. Vertical arrows illustrate contribution of Arcadia to IEEE processes.

Note: Arcadia does not address the full contents of all these tasks and processes, notably for business or mission analysis, ideation, need & solution space exploration, alternatives elicitation. See details for more information on Arcadia scope.



This figure introduces complementary services expected from Arcadia to support collaboration between engineering stakeholders and teams.



Each of the former expected contributions of Arcadia to engineering tasks is briefly described below.

Analyse What the users of the system need to accomplish

The first perspective on system engineering brought by Arcadia shall focus on analyzing the stakeholders needs and goals, their expected missions and activities, far beyond (and often before) customer requirements. This analysis also contributes ensuring adequate system need understanding with regard to its real operational use and IVVQ conditions, but it does not consider the solution or system per se.

Outputs of this engineering activity shall mainly consist of an "operational architecture" which describes and structures the stakeholders need in terms of actors/users, their operational capabilities and activities (including operational use scenarios with dimensioning parameters, and operational constraints such as safety, security, lifecycle, etc.).

Specify What the system has to accomplish for the users

This perspective shall focus on the system itself, in order to define how it can contribute to satisfy the former operational needs, along with its expected behavior and qualities. The main goal at that point is to check the feasibility of stakeholders requirements (cost, schedule, technology readiness, etc.) and if necessary, to provide means to renegotiate their content.

Outputs of this engineering activity shall mainly consist of system functional need descriptions (system capabilities, functions or services, functional chains, and scenarios), interoperability and interaction with the users and external systems (functions, exchanges plus non-functional constraints).

Define How the system will work so as to fulfil expectations

This perspective shall aim at building a notional component breakdown of the system at a coarse-grain level. Based on solution-oriented functional and non-functional analysis describing the designed behavior (functions, interfaces, capabilities, functional chains & scenarios, modes & states…), build one or several decompositions of the system into logical components. Its limited complexity level helps in exploring the solution alternatives.

All major (non-functional) constraints (safety, security, performance, IVV, cost, non-technical, Etc.) shall be taken into account and compared to each other so as to find the best trade-off. This approach is viewpoint-driven, where viewpoints formalize the way these constraints impact the system architecture.

Outputs of this engineering activity shall consist of the selected logical architecture which is described by components and justified interfaces definition, functional behavior, scenarios, modes and states, formalization of all viewpoints and the way they are taken into account in the components design. Since the architecture has to be validated against the need analysis, links with requirements and operational scenarios are also to be produced.

Finalise How the system will be designed, developed and built

This perspective shall define the "final" detailed architecture of the system at this level of engineering, ready to be developed according to implementation, technical and technological constraints and choices. The tradeoff around resources (e.g. power, communication, computation etc.) is addressed by introducing hosting physical components for implementation.

The same viewpoint-driven, functional-based approach as for logical architecture building is used. The model at that point is considered ready to develop by downstream engineering teams.

Outputs of this engineering activity shall consist of the selected physical architecture which includes global behavior, components to be produced, formalization of all viewpoints and the way they are taken into account in the components design. Links with requirements and operational scenarios are also produced.

Specify What is expected from each System Element supplier, and How to integrate

The fifth and last perspective shall be a contribution to EPBS (End-Product Breakdown Structure) building, taking benefits from the former architectural work, to enforce components requirements definition, and prepare a secured IVVQ (Integration Verification Validation Qualification).

All choices associated to the system/SW chosen architecture, and all hypothesis and constraints imposed to components and architecture to fit need and constraints, shall be summarized and checked here. IVV Strategy, including phasing/versioning, releases contents, integration trees, test means and enabling systems shall be defined based on the former perspectives models. Test campaigns are defined based on capabilities and scenarios.

Outputs from this activity shall be mainly "component Integration contracts" collecting all necessary expected properties for each constituent to be developed, on one side, IVV strategy and Test Campaigns/procedures on the other side.

Drive Integration

Arcadia shall provide support to:

- base IVV Strategy on operational capabilities, functional chains of the model;
- build IVV Schedule based on model dependencies;
- define sunny days/rainy days scenarios, impact analysis of IVV ups and downs and retroaction on versioning;

- contribute to test means and enabling systems definition from system and actors description in physical architecture.

Drive verification

Arcadia shall provide support to:

- base IVV Strategy on operational capabilities, functional chains of the model;
- build IVV Schedule based on model dependencies;
- define sunny days/rainy days scenarios, impact analysis of IVV ups and downs and retroaction on versioning;
- contribute to test means and enabling systems definition from system and actors description in physical architecture.

Drive validation

Arcadia shall provide support to:

- base IVV Strategy on operational capabilities, functional chains of the model;
- build IVV Schedule based on model dependencies;
- define sunny days/rainy days scenarios, impact analysis of IVV ups and downs and retroaction on versioning;
- contribute to test means and enabling systems definition from system and actors description in physical architecture.

Support Product Line Engineering

Arcadia shall provide support to:

- segment market & options thanks to need analysis,
- justify and check product variability Vs customer need;
- analyse consequences on architecture and check for compatibility;
- drive variability definition, justification and checks based on architecture analysis;
- derive projects architecture and assets based on selected options;
- secure reuse of existing components by multi-model confrontation.

Support Co-engineering with Specialties

Arcadia shall provide support to:

- Embed non-functional constraints of specialties (safety, security, RAMT etc.) in model, support risk analysis;
- analyze solution based on specialty golden rules, and bridge with specialty detailed analysis and assets;
- perform a multi-viewpoint approach to find best architectural compromise in short loop.

Notably regarding Supervision Engineering, Arcadia shall provide support to:

- define system modes and states, situations faced by the system, associated behaviour, configurations and expectations (e.g. reliability, resilience…);
- analyze system behavior in critical situations and consequences of failures, startup and shutdown phases, reconfigurations, etc.;
- verify concurrent modes & states logics and architecture adequacy.

Support Co-engineering between Engineering Levels

Arcadia shall provide support to:

- lead interface definition/justification and ensure global consistency by global functional analysis;
- define and check coherency of definition of required subsystems;
- master complexity by separate coupled models and formalized requirements, justification and impact analysis;
- deal with reusable components.

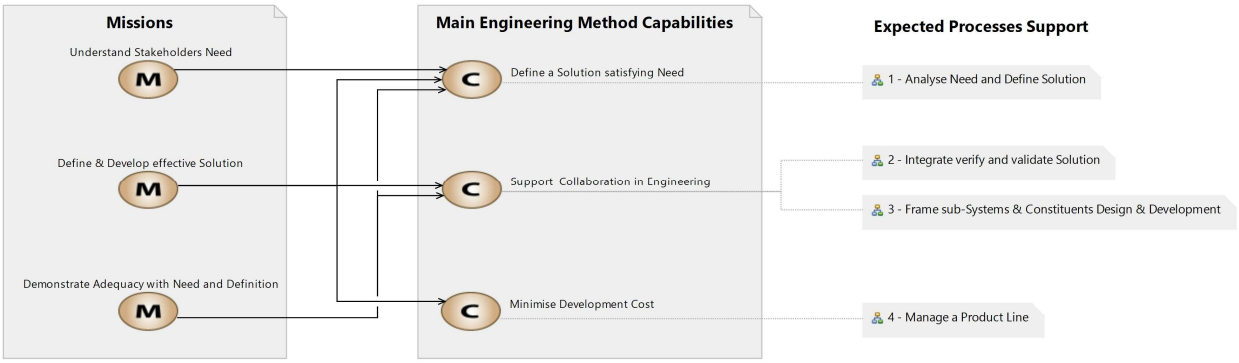
Drive agile Development

Arcadia shall provide support to:

- drive and master incremental engineering and design, software agile and DevSecOps sprints and user stories from capabilities and functional chains/scenarios;
- maintain a view of final target architecture view, and update it according to sprints results & evaluations

6 How do Arcadia perspectives and services orchestrate?

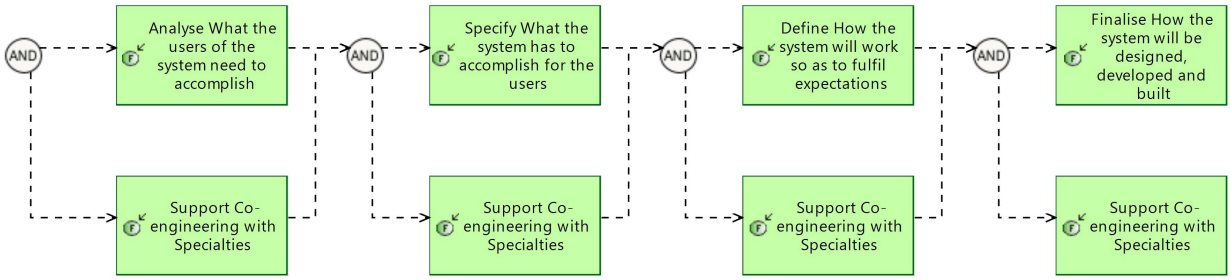
This figure presents focused capabilities expected from Arcadia according to engineering objectives, illustrated by simplified processes involving the former Arcadia services.



The very simplified engineering processes sketched below illustrate how Arcadia-supported tasks can contribute to each capability and the way they cooperate.

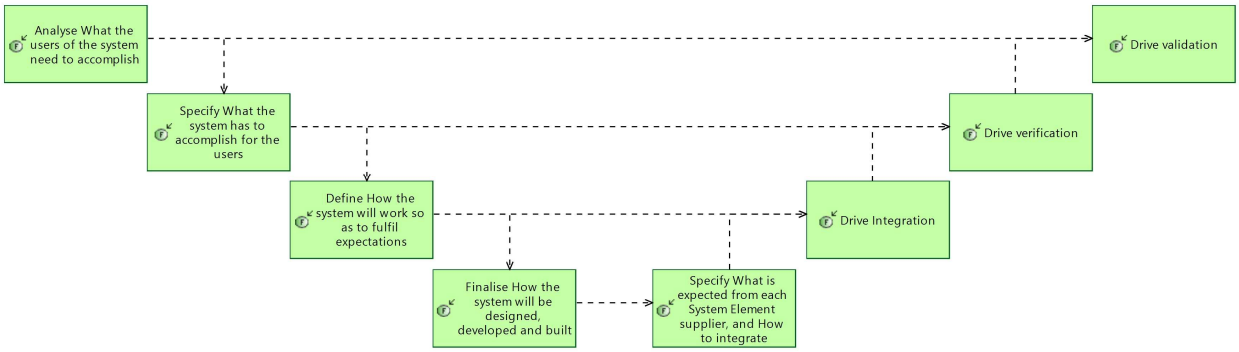
1 - Analyse Need and Define Solution

This process illustrates an expected way to build a solution compliant with need and expectations, in co-engineering.



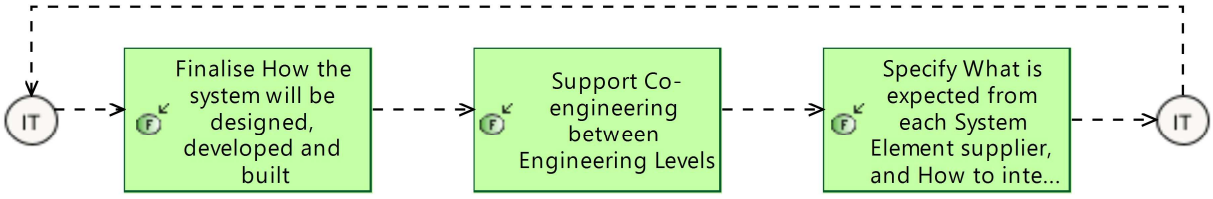
2 - Integrate verify and validate Solution

This process illustrates an expected way to drive integration, verification and validation of the solution, based on need and solution definition outputs.



3 - Frame sub-Systems & Constituents Design & Development

This process illustrates how need and solution definition outputs should drive the definition, design and development of solution constituents.



4 - Manage a Product Line

This process illustrates how need and solution definition outputs should drive the definition, design and development of the product line, which in turn should shape solution definition and adaptation to product/project.