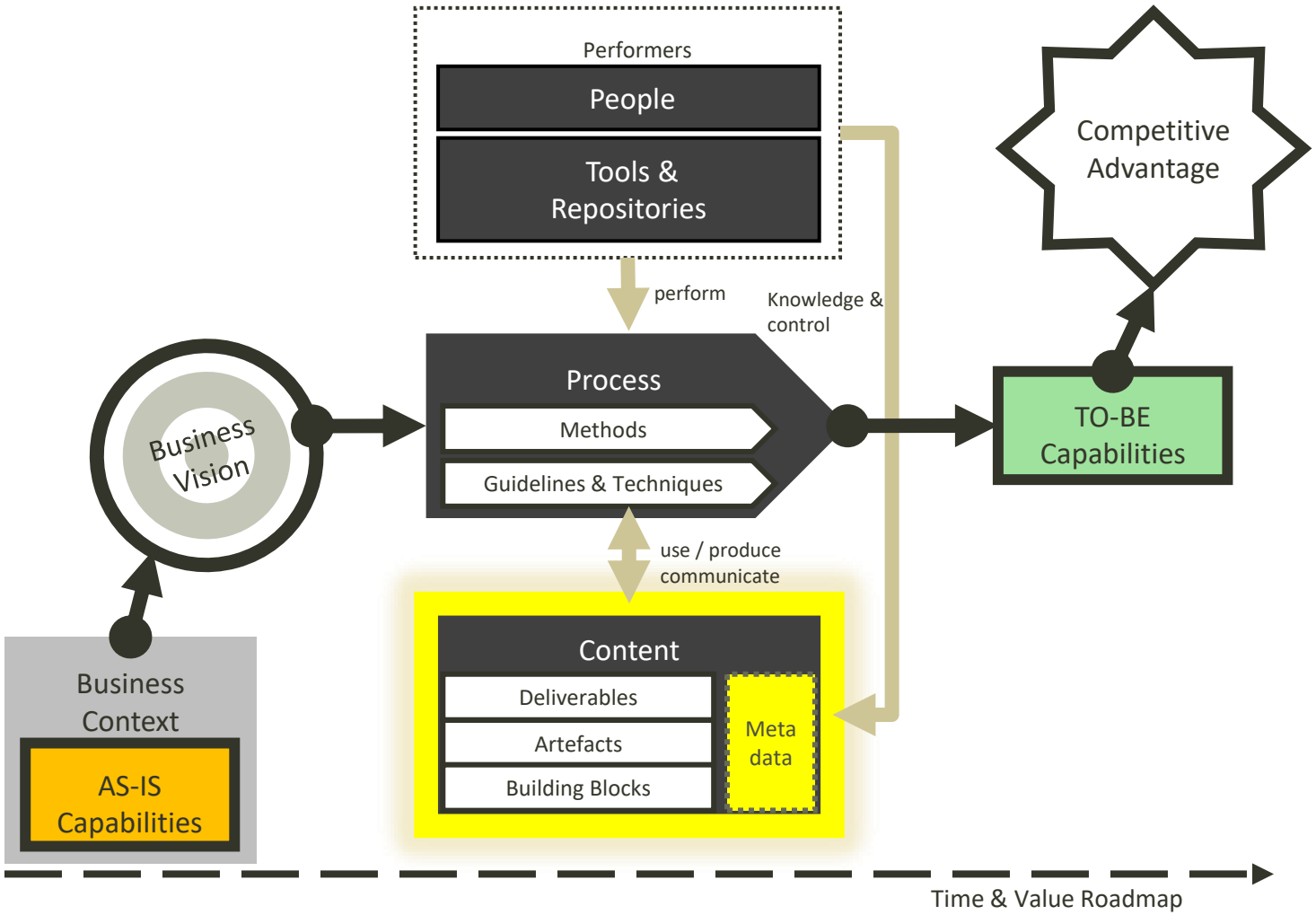
Unified Framework for Driving Transformations

Modeling Language

One single strategy and architecture modeling language

derived from pure systems semantics

Modeling Language



What the strategy and architecture modeling language needs to represent…

- Any system involving any combination of people, software and physical equipment (an enterprise, a country, a software, a farm, nature…)
- The vision for the system to be (strategy definition & execution)
- Architecture changes for realizing the vision

- The strategy and architecture views needed
- throughout a precise and actionable process for driving transformations

- Several levels of detail for managing complexity
- Time as another dimension of the exact same model instances

- Manageable portfolios of items including applications, organizations, equipment, information and material

- with precise and unambiguous semantics (single metamodel)
- with a single graphical representation that is similar to what most modelers are used to
- so that multiple disciplines can use the same language and effectively collaborate

What we don’t want as a language for strategy and architecture

- A language nicely designed to represent code e.g. UML
- A language nicely designed to represent configuration data e.g. BPMN (for workflow engines)
- A language designed without any clear vision
  - of what the language should semantically represent
  - of which precise and actionable process the language needs to be used for
- A set of languages that were not designed to be integrated, and which potentially combine all the above characteristics

Creating a language that makes sense…

Enterprise Architecture

Strategic Management

Performance Management

Business Process Management

Customer Experience Management

Organizational Design

Data Management and Analytics

Technology Management

Risk Management

Governance

Compliance and Regulatory Management

Transformation Disciplines

Viewpoint

Influence

L1

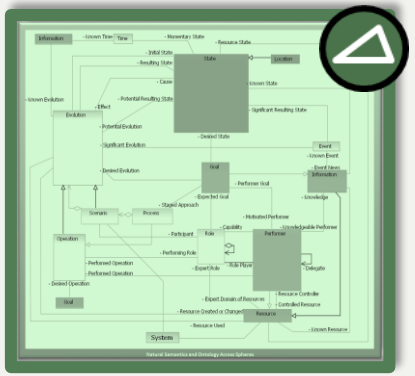
L2

L3

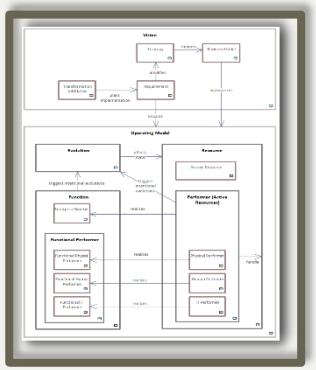
Enterprise Description + Vision + Changes

Drive Transformations

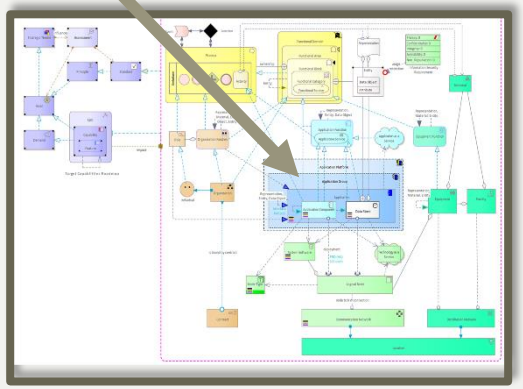
Language items used in viewpoint



Systems Semantics



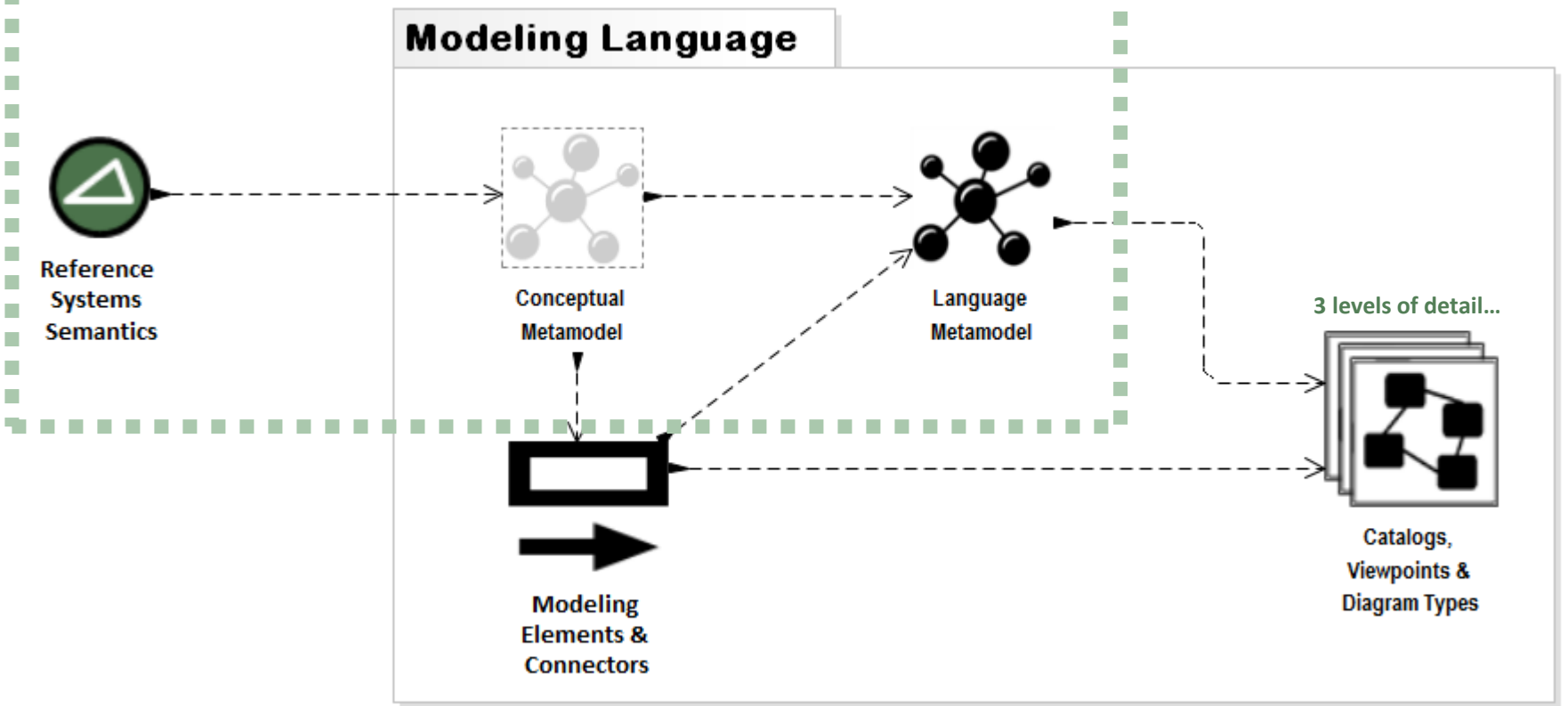
Conceptual Metamodel



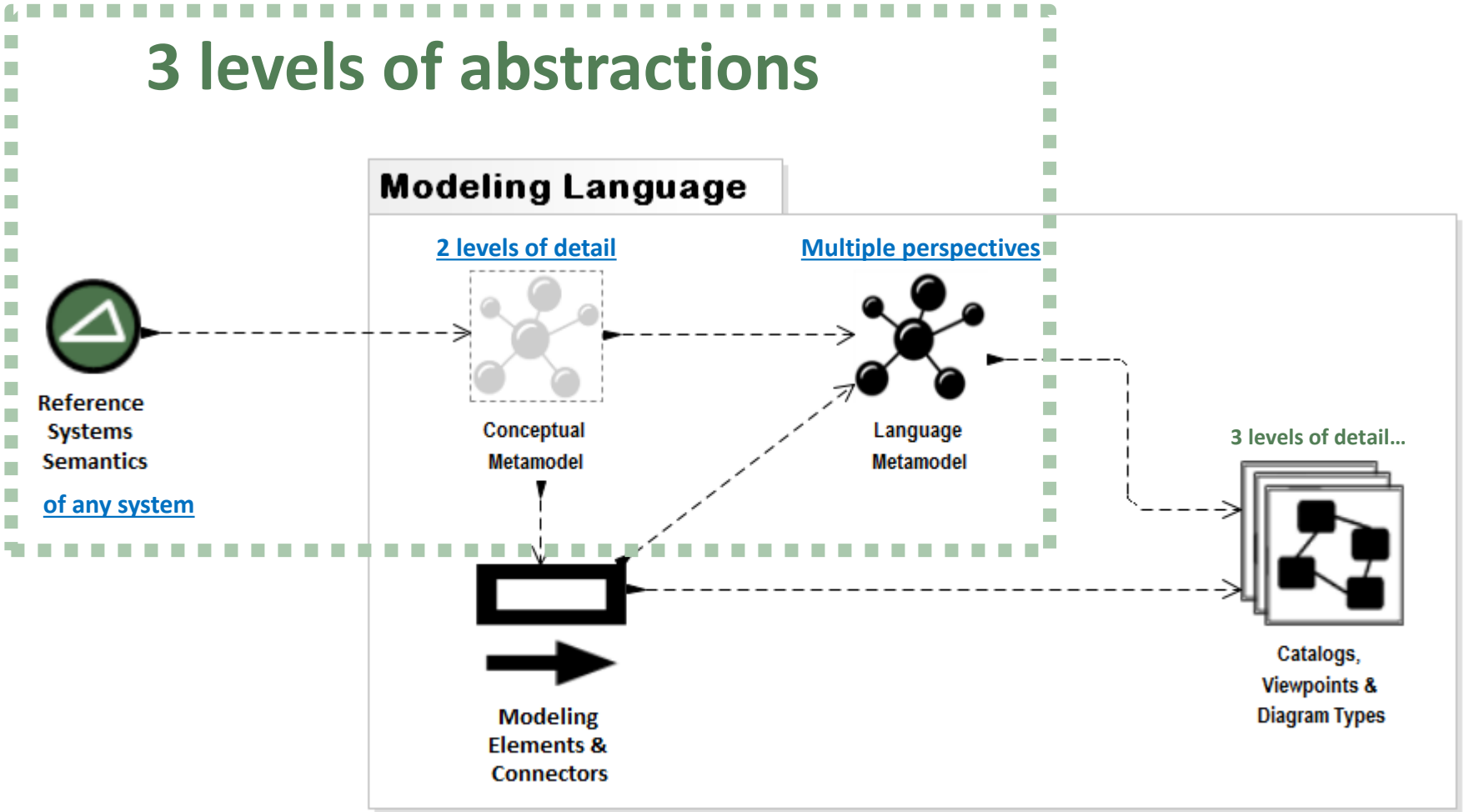
Language Metamodel

The detailed Labnaf modeling language was designed step by step using

3 levels of abstractions



The detailed Labnaf modeling language was designed step by step using



The Labnaf Modeling Language in practice…

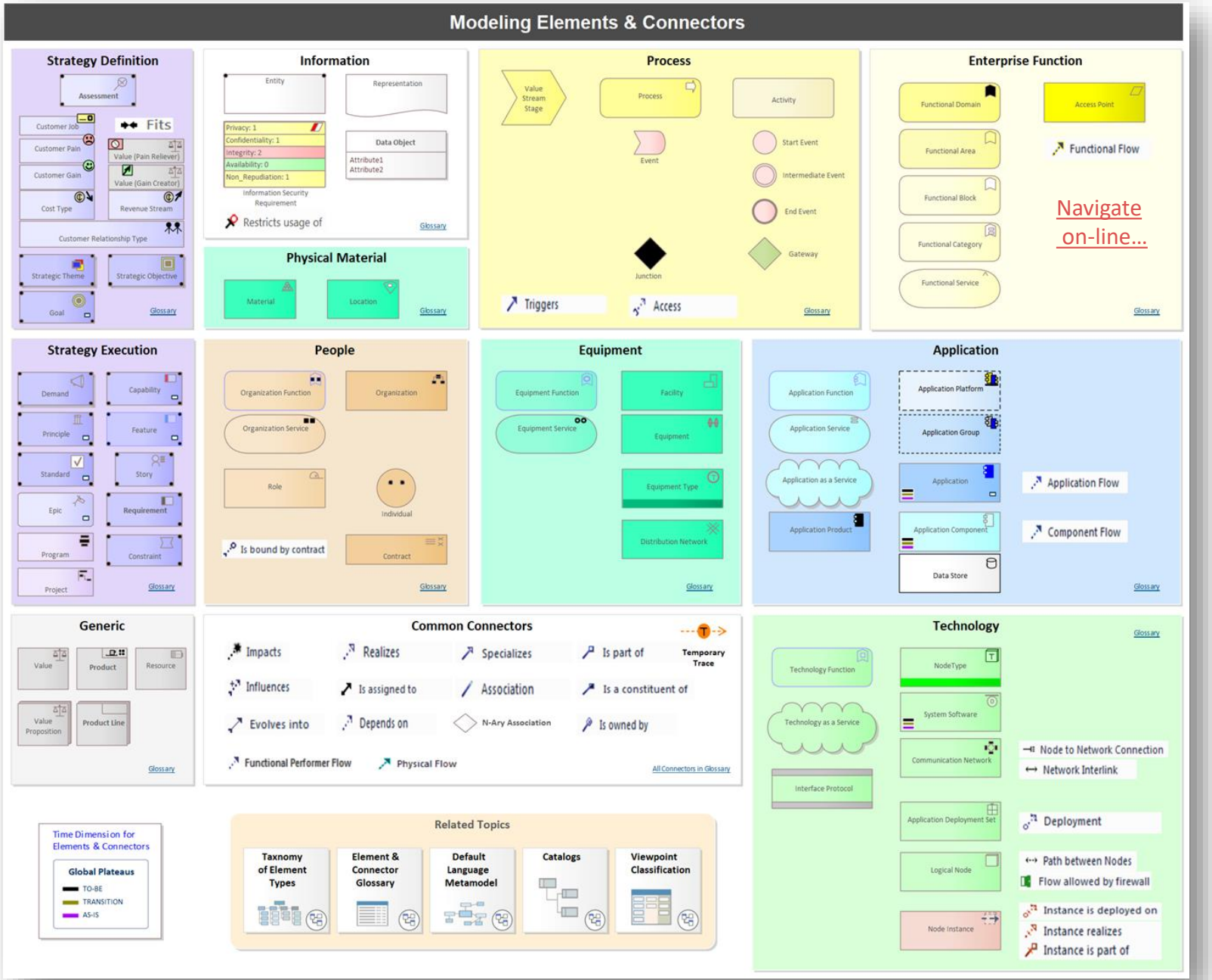
The Modeling language covers the entire process of driving transformations.

It simplifies and rationalizes the modeling experience by unifying transformation disciplines into a single homogeneous set and by concentrating on what is needed for the work at hand.

The resulting set of simplified and integrated modeling toolboxes features precise and unambiguous modeling elements and connectors.

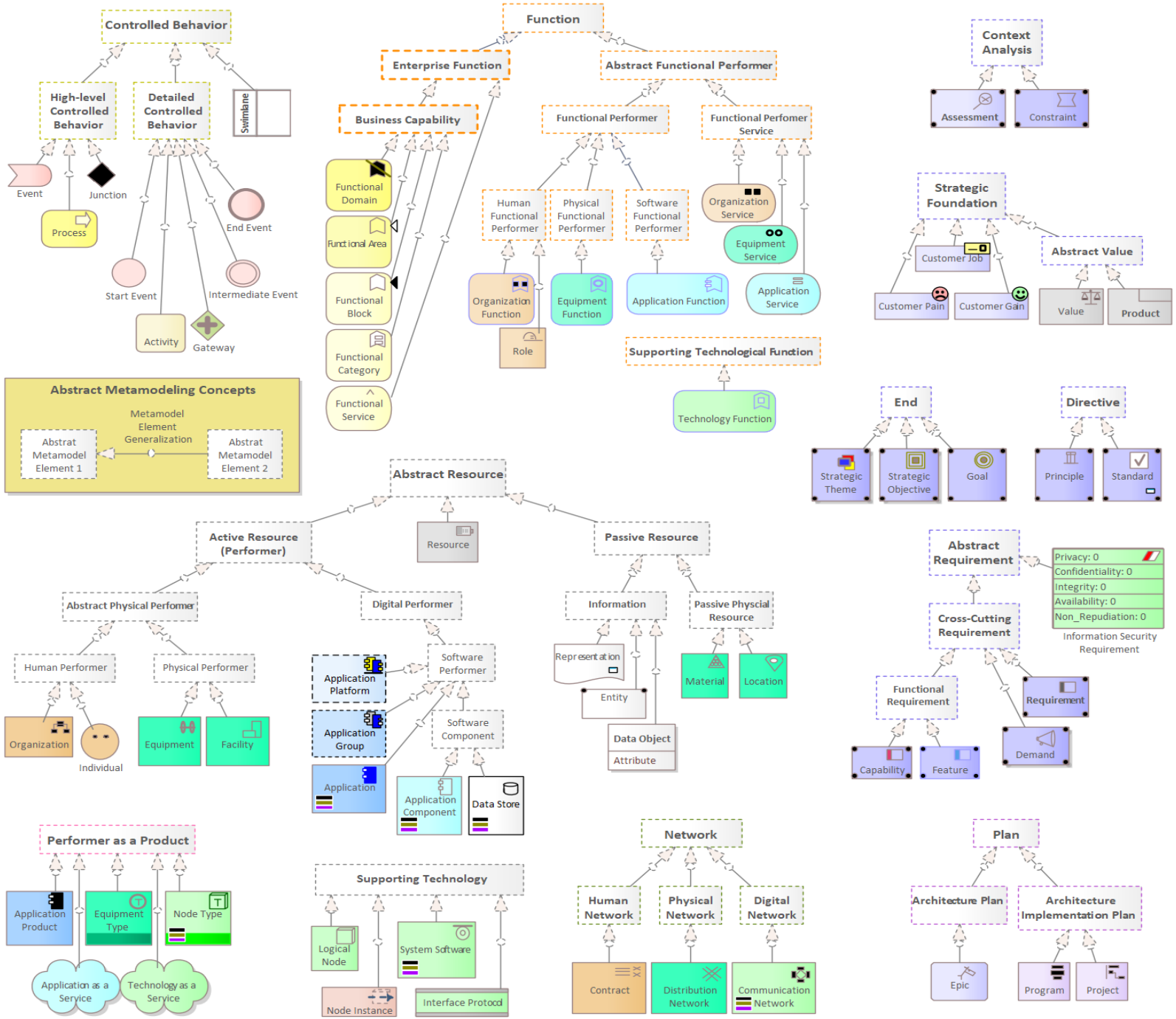
The language is implement using the Sparx Enterprise Architect Modeling tool platform.

Labnaf Element and Connector Types

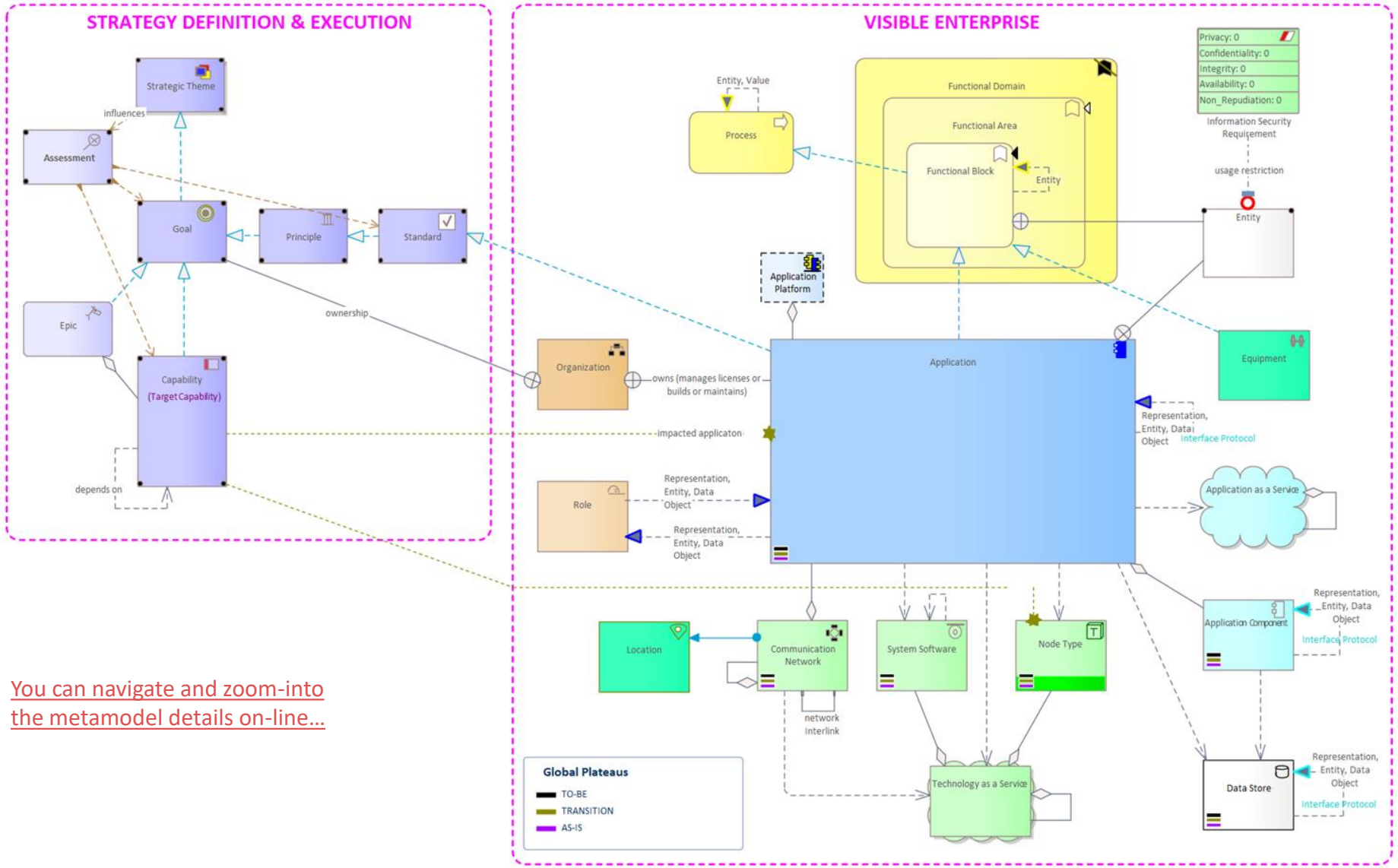


Taxonomy of Element Types

Part of the standard metamodel

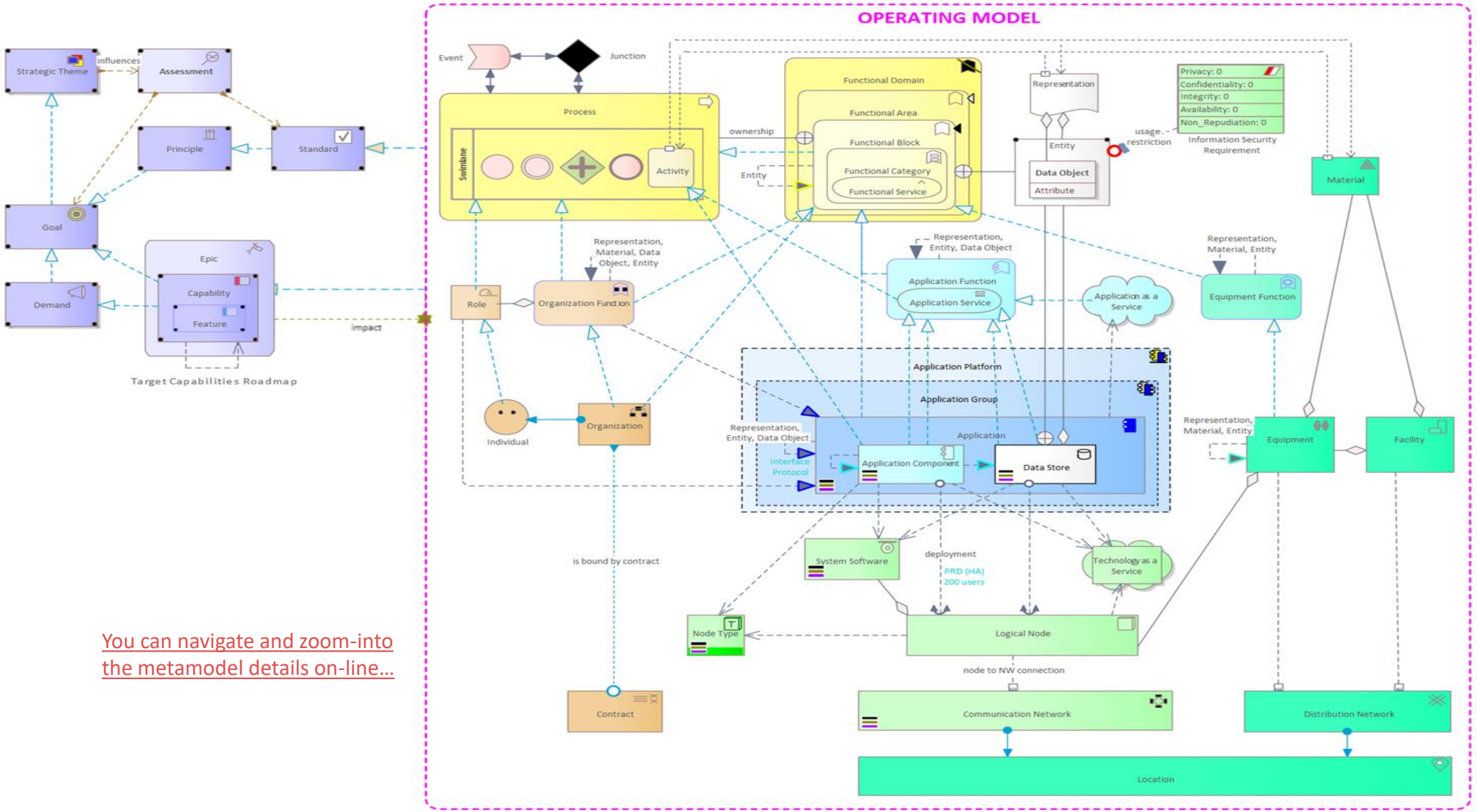


A Subset of the Standard Metamodel including Strategy & **Enterprise** Architecture



You can navigate and zoom-into the metamodel details on-line…

A Subset of the Standard Metamodel including Strategy & **Solution** Architecture



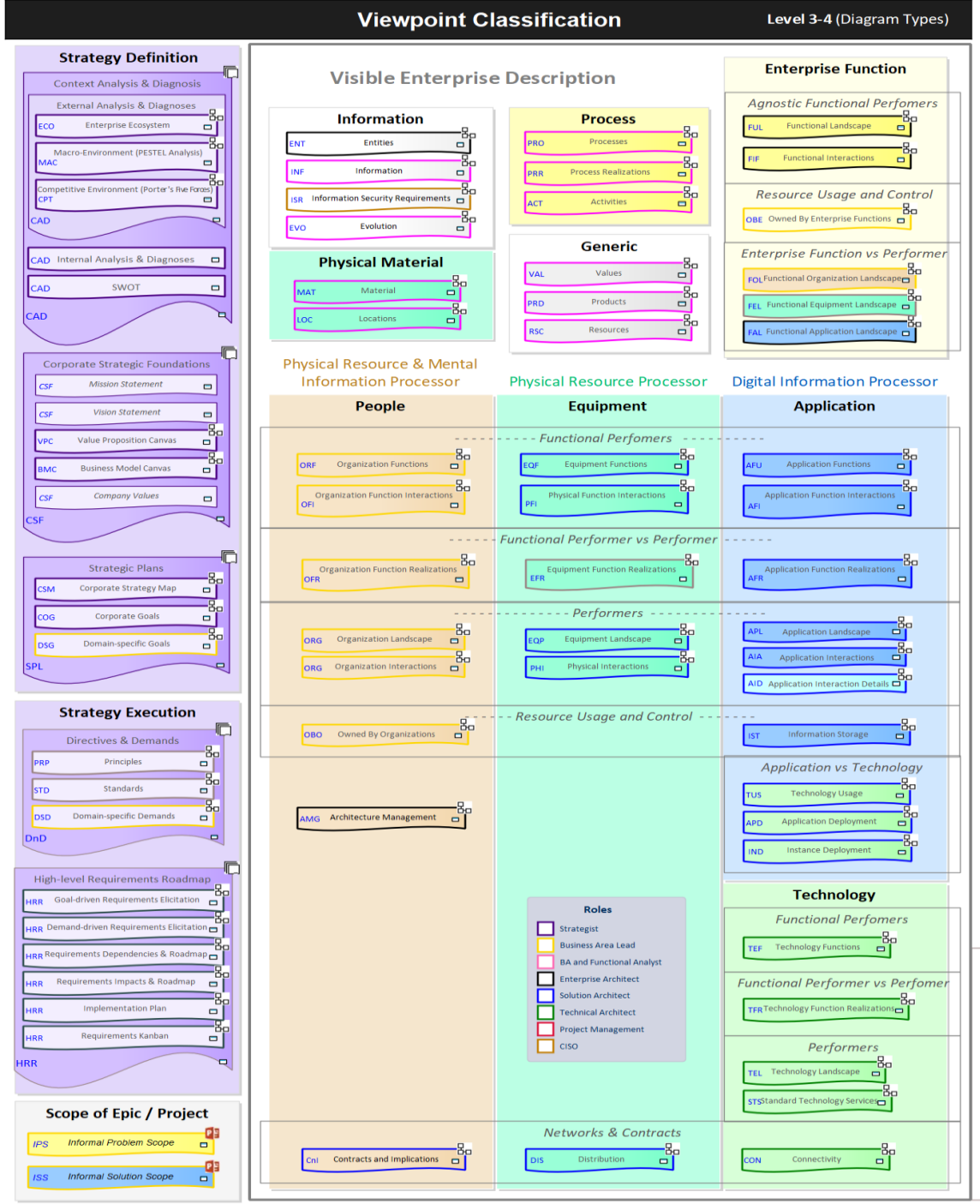
You can navigate and zoom-into the metamodel details on-line…

These are the predefined Labnaf viewpoints

A **Viewpoint**

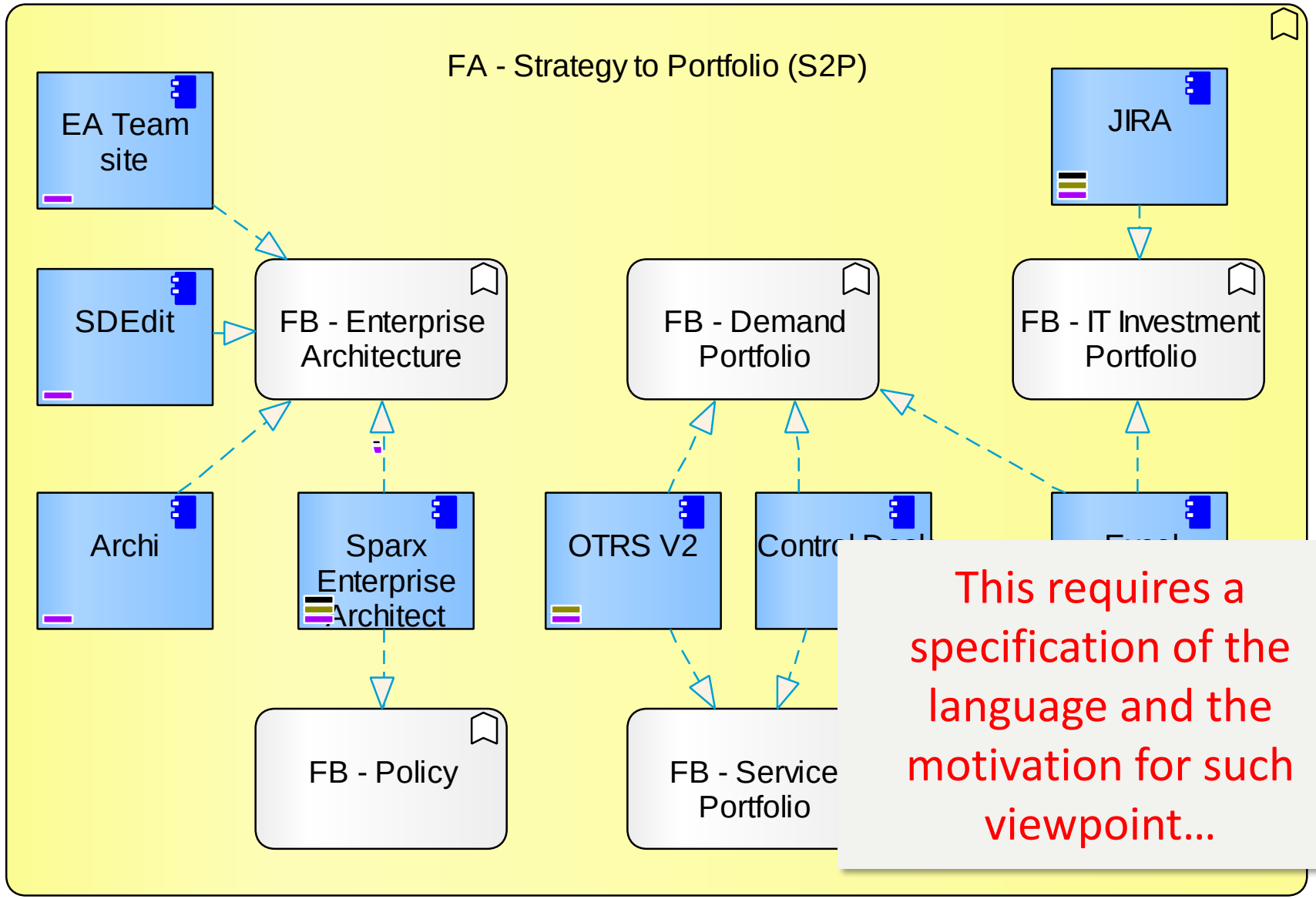
is a type of view

Navigate on-line…



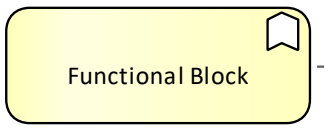
Viewpoint are organized following architecture perspectives

Sample “Functional Application Landscape” View



This requires a specification of the language and the motivation for such viewpoint…

Prescriptive language for "Functional Application Landscape" views



A Functional Block is a level 3 business function that belongs to some functional area.
The granularity and scope of a Functional Block is defined by identifying

- some homogenous set of information that the Functional Block is mastering
- a group of activities that fulfill the purpose of the functional block, that belong to some processes of same nature and that produce and use the information mastered by that Functional Block

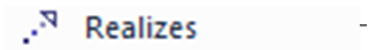
A business function is a behavior element that groups behavior based on a chosen set of criteria e.g. required business resources and/or skills, competencies, knowledge, etc.

Inspired by Archimate

An application

- Is a self-contained unit of functionality as perceived by end-users
- Can be clearly mapped to some functional blocks
- Has its own specific set of application attribute values
- Is used by and billable to one or several Organizations
- Is owned by a single Organization
- Can be part of an Application Platform or an Application Group
- Encapsulates Applications Components and Application Interfaces
- Can exist at one or many specific points in time called "plateaus". Possible plateaus are AS-IS, TRANSITION and TO-BE.

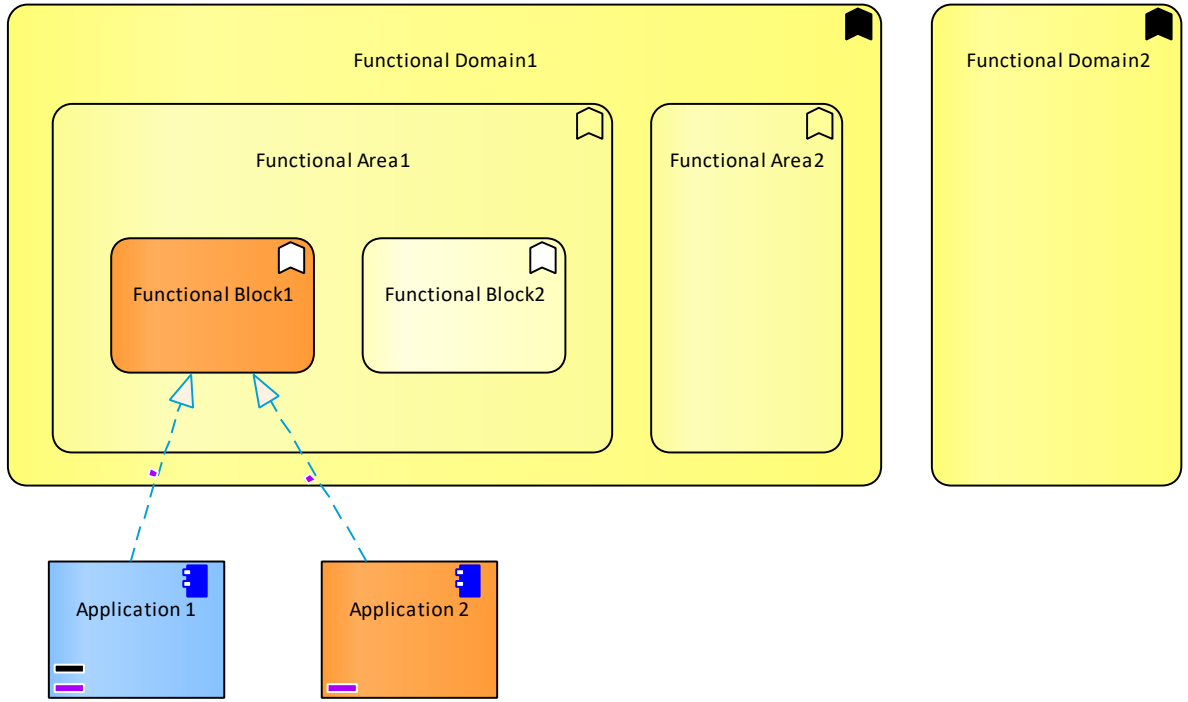
The name of an application component should preferably be a noun.



A **Realization** relationship indicates which concrete entities ("how") realize which abstract entities ("what"). The realization relationship is used in a business operational sense (e.g., a role realizes a swim-lane of activities), but also in an IT context (e.g., an application realizes a functional block).

Inspired by UML & Archimate

Motivation for creating “Functional Application Landscape” views



This answers the following questions

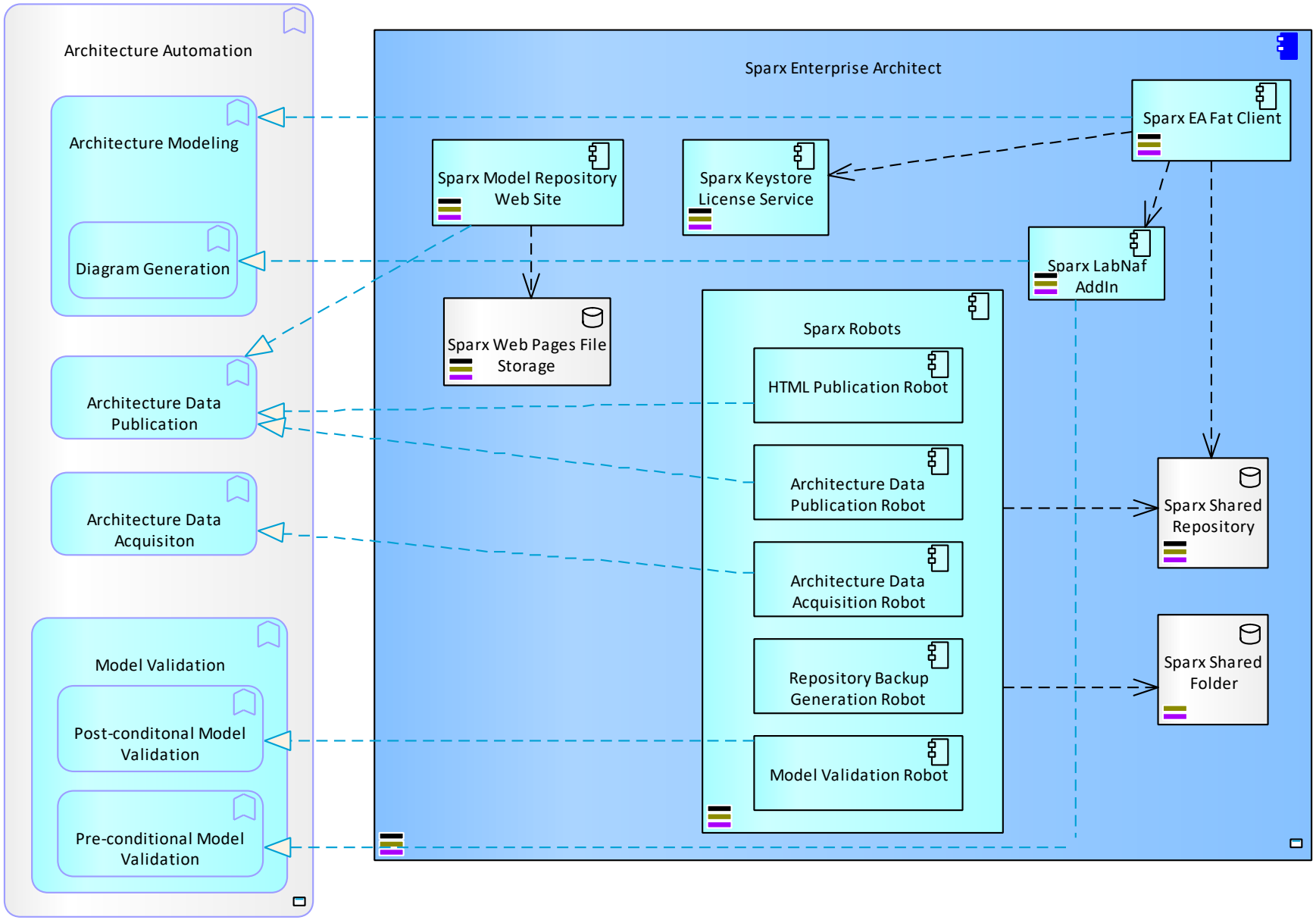
When used as enterprise-wide viewpoint

- Which **applications** support which **functional blocks**?

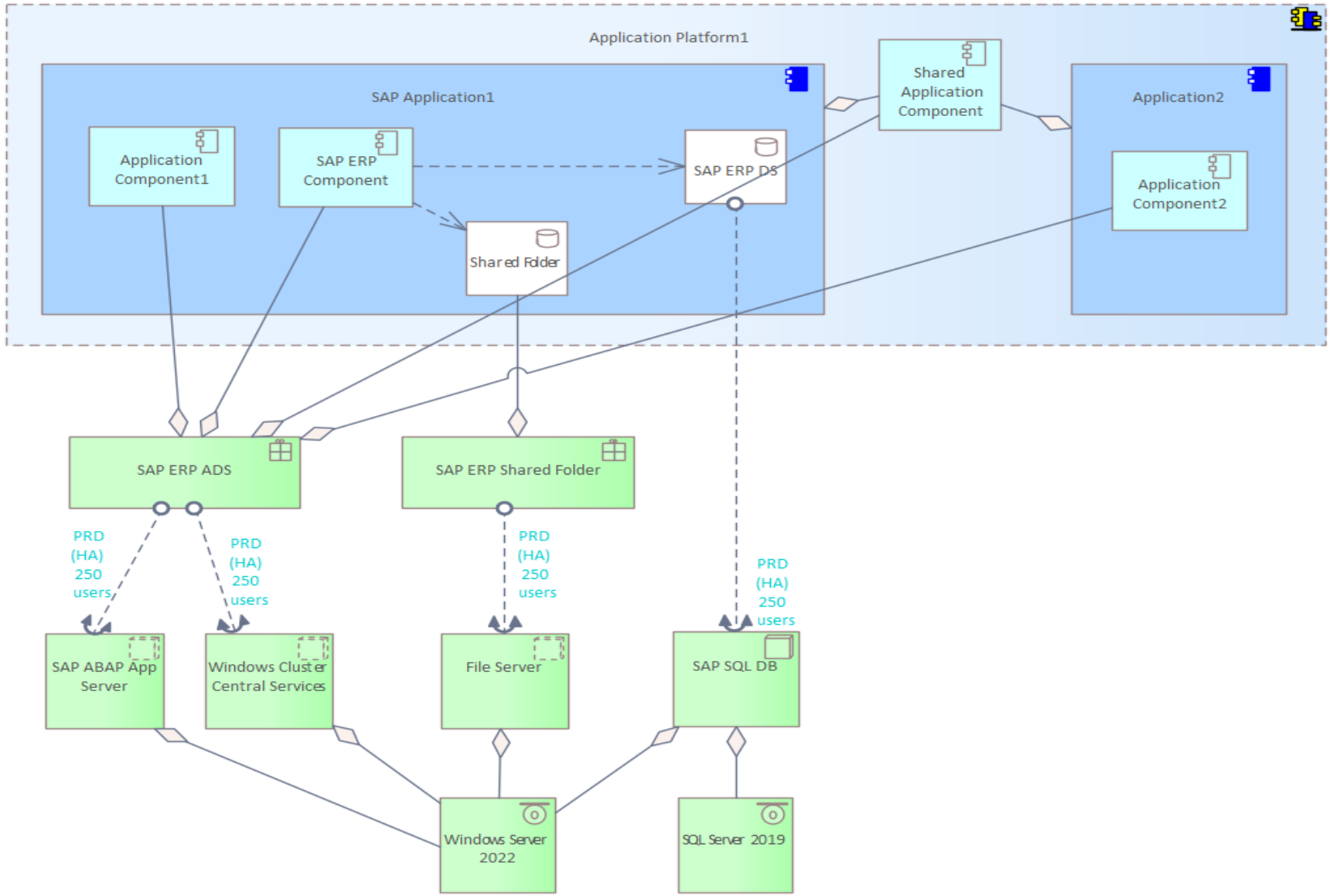
When used as project-specific viewpoint

- Which **applications** are/will automate the **functional blocks** inside the scope of this project?

Sample “Application Landscape” View



Sample “Application Deployment” View



Elements & connectors

Vision
- Assessment
- Customer Job
- Customer Pain
- Customer Gain
- Revenue Stream
- Cost Type
- Customer Relationship Type
- Strategic Theme
- Strategic Objective
- Goal
- Principle
- Standard
- Demand
- Capability
- Feature
- Story
- Requirement
- Constraints
- Epic
- Program
- Project

Information
- Entity
- Representation
- Data Object
- Information Security Requirement

Physical Material
- Location
- Material

People
- Organization Function
- Organization Service
- Role
- Organisation
- Individual
- Contract

Generic
- Product
- Resource
- Value

Process
- Process
- Event
- Junction
- Swimlane
- Activity
- Start Event
- Intermediate Event
- End Event
- Gateway

Equipment
- Equipment Function
- Equipment Service
- Equipment
- Equipment Type
- Facility
- Distribution Network

Enterprise Function
- Functional Domain
- Functional Area
- Functional Block
- Functional Category
- Functional Service
- Access Point

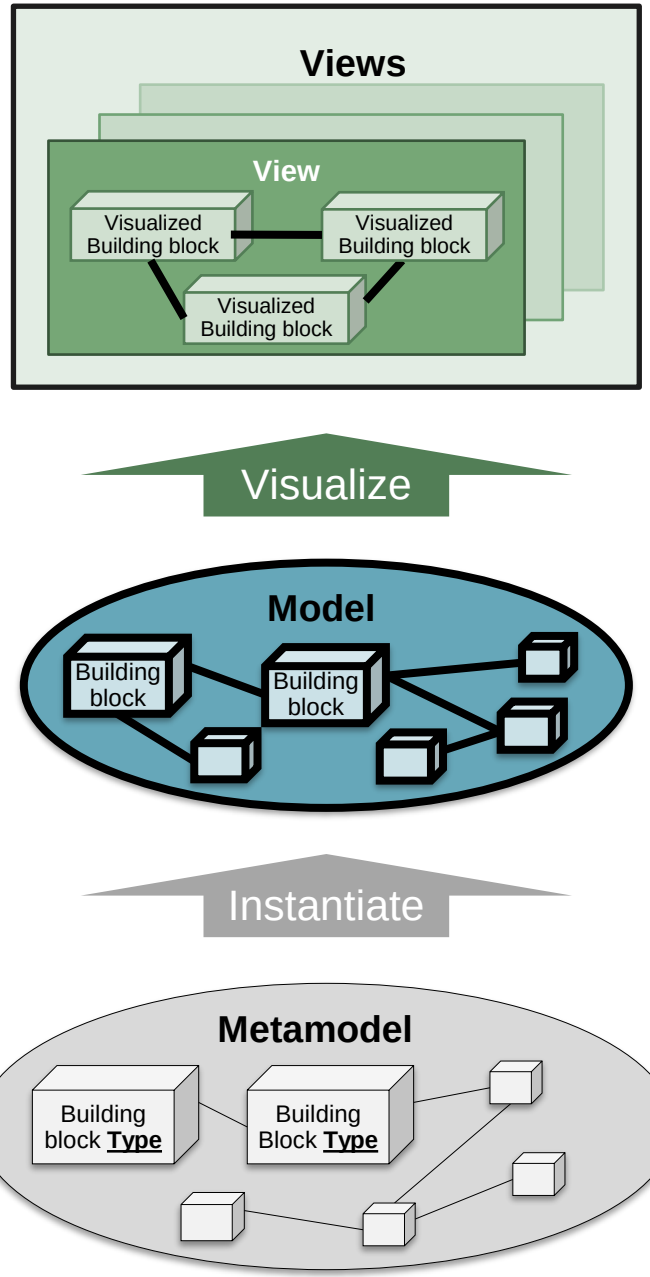
Application
- Application Function
- Application as a Service
- Application Service
- Application Platform
- Application Group
- Application
- Application Component
- Data Store
- Application Product

Technology
- Technology Function
- Technology as a Service
- Node Type
- System Software
- Communication Network
- Interface Protocol
- Application Deployment Set
- Logical Node
- Instance Deployment Set
- Node Instance

All Connectors
- Access
- Attach to flow
- Is part of
- Association
- Application Flow
- Is assigned to
- Component Flow
- Is a constituent of
- Is bound by contract
- Depends on
- Deployment
- Evolves into
- Fit
- Functional Performer Flow
- Flow is allowed by firewall
- Functional Flow
- Specializes
- Impacts
- Influences
- Instance is part of
- Instance is deployed on
- Instance realizes
- Network Interlink
- Node to Network Connectic
- Is owned by
- Path between Nodes
- Physical Flow
- Realizes
- Triggers
- Restricts usage of

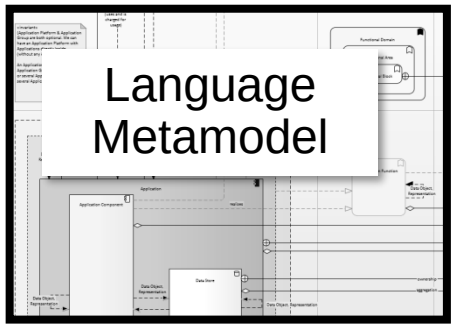
When presented to end users, use **verbs** instead of connector names when the direction of the connector is ambiguous

We use a Metamodel to define the different Types of building blocks and relationships

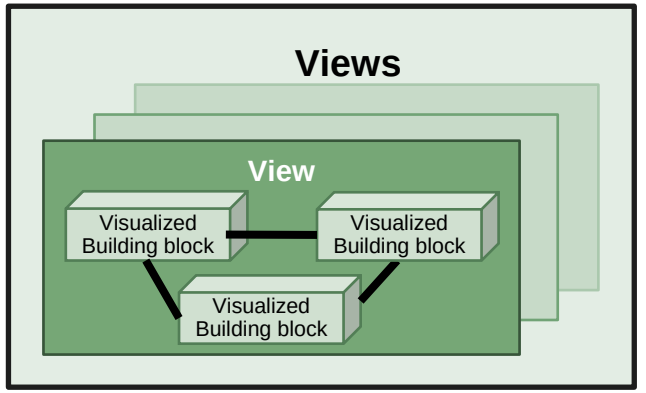


Metamodel

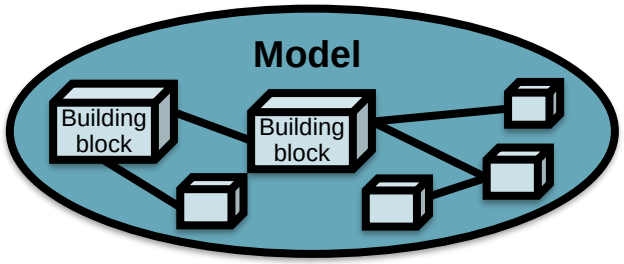
- The metamodel identifies all model element types (e.g. function, process, application, data object…).
- It also defines all possible relationship types between these elements (e.g., applications support functions).



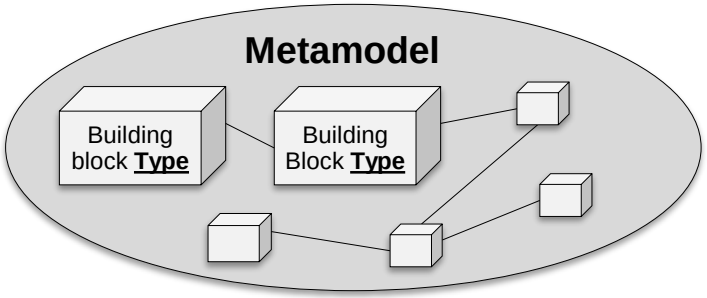
Instantiate



Visualize



Instantiate

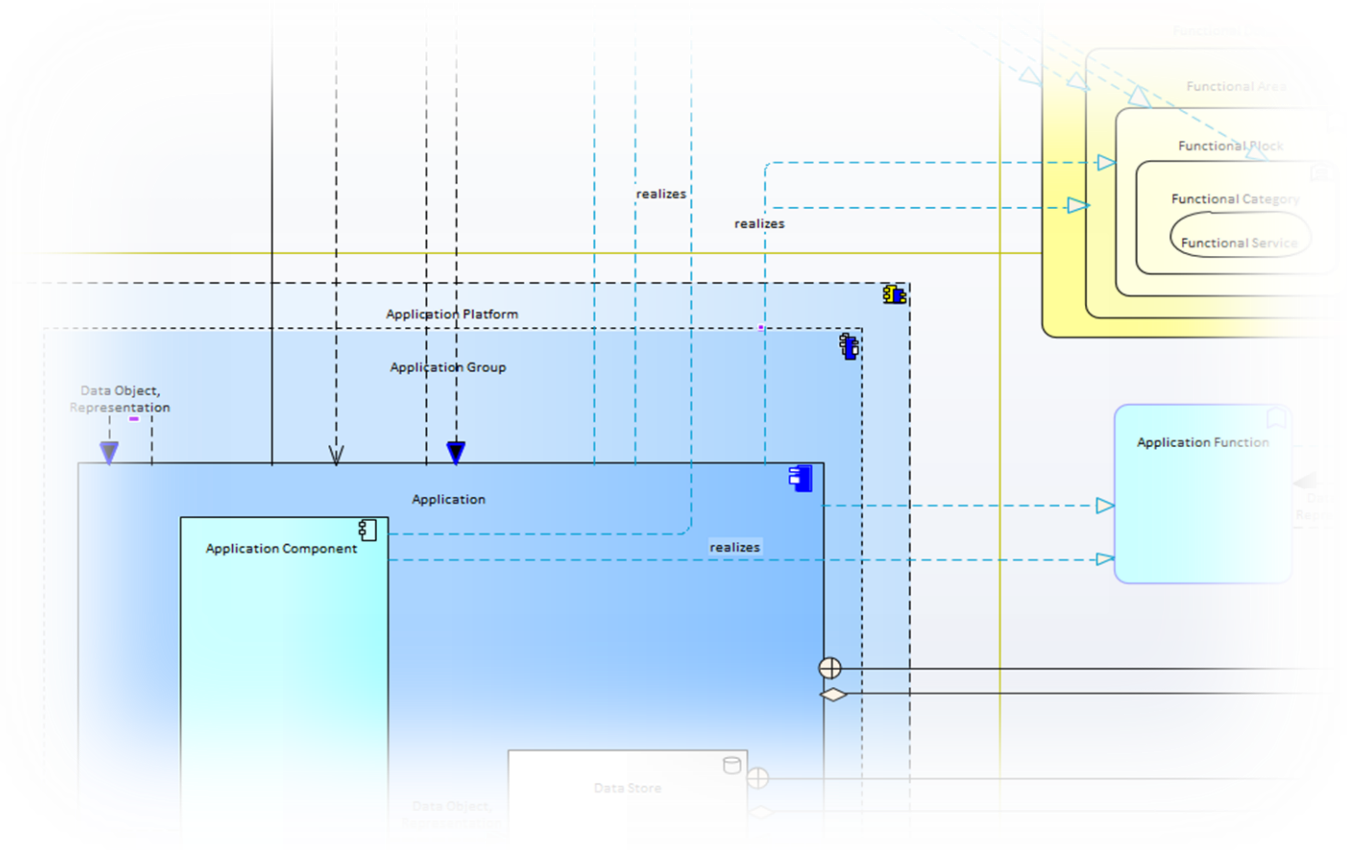


We use a Language Metamodel to define the different Types of visualized building blocks and relationships

Implement as a modelling language

Language Metamodel

See on-line documentation



For further information about the language…

Detailed documentation and contact information are available here:

www.Labnaf.one