Digital Architectural Framework



Digital Architecture

Framework

DAF v. 6.0

June 2025

# About Digital Architecture Framework (DAF)

The Digital Architecture Framework (DAF), formerly known as "Delta Architectural Framework," is a set of modeling tools for Digital Engineering aiming to enable collaboration between roles such as Enterprise Architects, Solution Architects, and Business Analysts in designing the enterprise by creating high-quality models that can be transformed into artifacts such as code and documentation.

Figure 1: Value Streams covered by DAF

The DAF meta-model is based on industry standards such as UML, ArchiMate, TOGAF, RUP, SPEM and best practices for requirement management, and offers a baseline of more than 50 concepts and properties that can be customized to meet specific needs. While DAF can be used on its own, it is not intended to represent a new standard but rather serves as an accelerator that allows for quick tailoring of existing standards.

It may accelerate the delivery of projects by supporting collaboration of different roles and by creating traceability between strategy and execution.

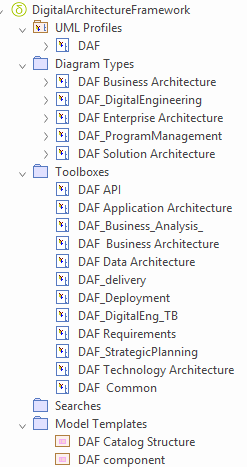


Figure : content of the DAF MdG

It's implemented as MDG technology in Sparx Enterprise Architect, including:

* UML profile with a set of customized icons
* multiple Toolboxes, organized by roles
* Custom diagrams, organized by roles
* Model patterns, accelerating the creation of DAF diagrams.
* Model Template with the structure of Catalogs
* Shape scripts
* Quick linker
* A set of 100+ diagram examples showing how to use DAF to cover Enterprise Architecture, Solution Architecture, Design Architecture and Business Analysis. Those examples include all the diagrams in the TOGAF 9.1 specification.
* A structure based on TOGAF to host an enterprise repository
* A set of scripts, performing model transformations
* A set of Rules for model Validation

# License

DAF is Open Source, released under the Creative Common

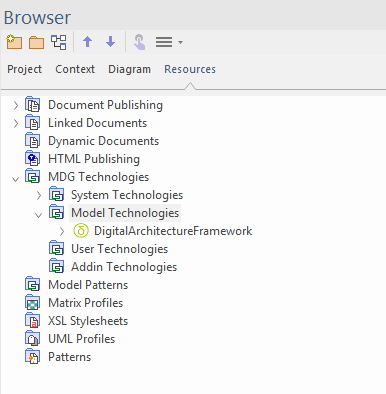
Creative Commons Licence

[Digital Architecture Framework](https://github.com/FreeTAKTeam/DigitalArchitectureFramework)© 2025 by [Giu Platania](https://www.linkedin.com/in/giu-platania)is licensed under [Creative Commons Attribution 4.0 International l](https://creativecommons.org/licenses/by/4.0/?ref=chooser-v1)

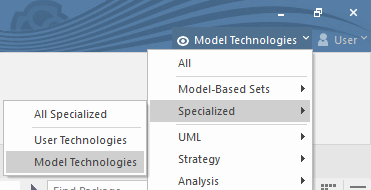
[This license requires that reusers give credit to the creator. It allows reusers to distribute, remix, adapt, and build upon the material in any medium or format, even for commercial purposes.](https://creativecommons.org/licenses/by/4.0/?ref=chooser-v1)

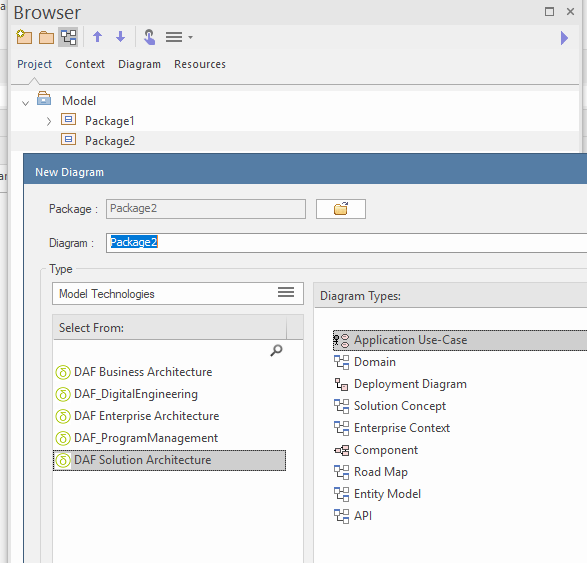
# Installation

1. **Obtain the DAF MDG file**: Ensure you have the xml file defining the DAF MDG Technology. This file contains the definitions for stereotypes, toolboxes, profiles, and diagram types.
2. Save the file to an accessible directory on your computer.
3. Launch **Sparx Enterprise Architect**.
4. Open or create a project where you want to use the DAF MDG Technology.



1. In the browser, open the tab Resources and navigate to **MDG technologies> Model Technologies** , right click and select **Import Technology**.
2. Locate and select the .xml file for DAF.
3. Select **import to Model**
4. Click **OK**  to load the MDG Technology into Sparx EA.
5. **optional: Enable the DAF MDG Technology**
   * Go to **Specialize > Technologies > Manage Technologies**.
   * In the list of available MDG Technologies, find "DAF" and check the box to enable it.
   * Click **Close** to save your settings.
6. **Suggested: set your perspective focus on DAF**



1. Test a DAF diagram
   * Create a new diagram
   * Select one DAF Diagram from the options  
     

# Concepts

## From Reality to Abstraction

DAF is a Metamodel (M2) that describes the elements of a Model (M1) that is an abstraction of a real system (M0).

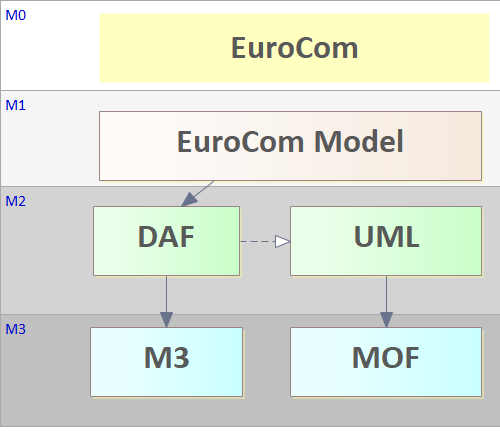


Figure : The hierarchy in this diagram follows the MDA principle of abstraction, where each level (M0 to M3) adds structure and abstraction to the level below, ensuring a consistent and scalable modeling approach.

* **M0: EuroCom:** The M0 layer represents the real-world organization or domain of interest, here as "EuroCom." This is the concrete instance of the domain that the models aim to represent, analyze, and improve. It consists of actual data and entities in the operational environment.
* **M1: EuroCom Model**: The M1 layer contains the model of the M0 domain. In this case, the "EuriCom Model" represents an abstraction of the EuroCom organization, describing its structure, behavior, or other aspects at a higher abstraction level. The M1 model acts as a blueprint that specifies how the M0 system is represented.
* **M2: DAF and UML**: The M2 layer defines the metamodels that describe the structure and semantics of the models in the M1 layer The arrow from DAF to UML indicates influence, highlighting how DAF integrates concepts from UML.
* **M3: M3 and MOF**: At this level, the fundamental rules and concepts for defining modeling languages are established. The M3 layer represents the meta-metamodels that describe the foundation for defining metamodels like UML or DAF. M3 is a custom meta-metamodel that can describe DAF or other metamodels. M3 is used instead of MOF (Meta-Object Facility): A standard meta-metamodel defined by the OMG to provide a formal foundation for modeling languages like UML.

## From the Model to a view

There is a formal way of separating the idea of “one model” from the way we look at it, develop it, and communicate it.



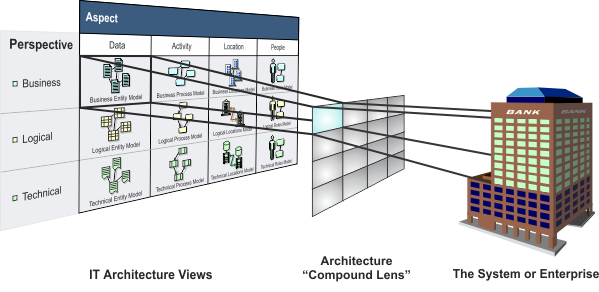
The Digital Engineering approach aims to enable the separation of concerns across different models, metamodels and viewpoints, allowing for a more flexible, maintainable and understandable representation of the system, making it possible for different stakeholders to focus on the aspects of the system that are relevant to them.

A "**metamodel**" is a model of a model, which defines the structure and constraints of the models that conform to it. It describes the concepts, properties, and relationships that a model can contain, as well as the constraints and rules that apply to them. Metamodels are often expressed in formal languages, such as MOF or eMOF.

The concept of a "**model**" refers to a representation of a system or part of it in a specific format that adheres to the metamodel (e.g. UML or ArchiMate). These models can be used to define the architecture of a system. The ISO/IEC/IEEE 42010 standard defines "*architecture*" as "*the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles guiding its design and evolution*."

.

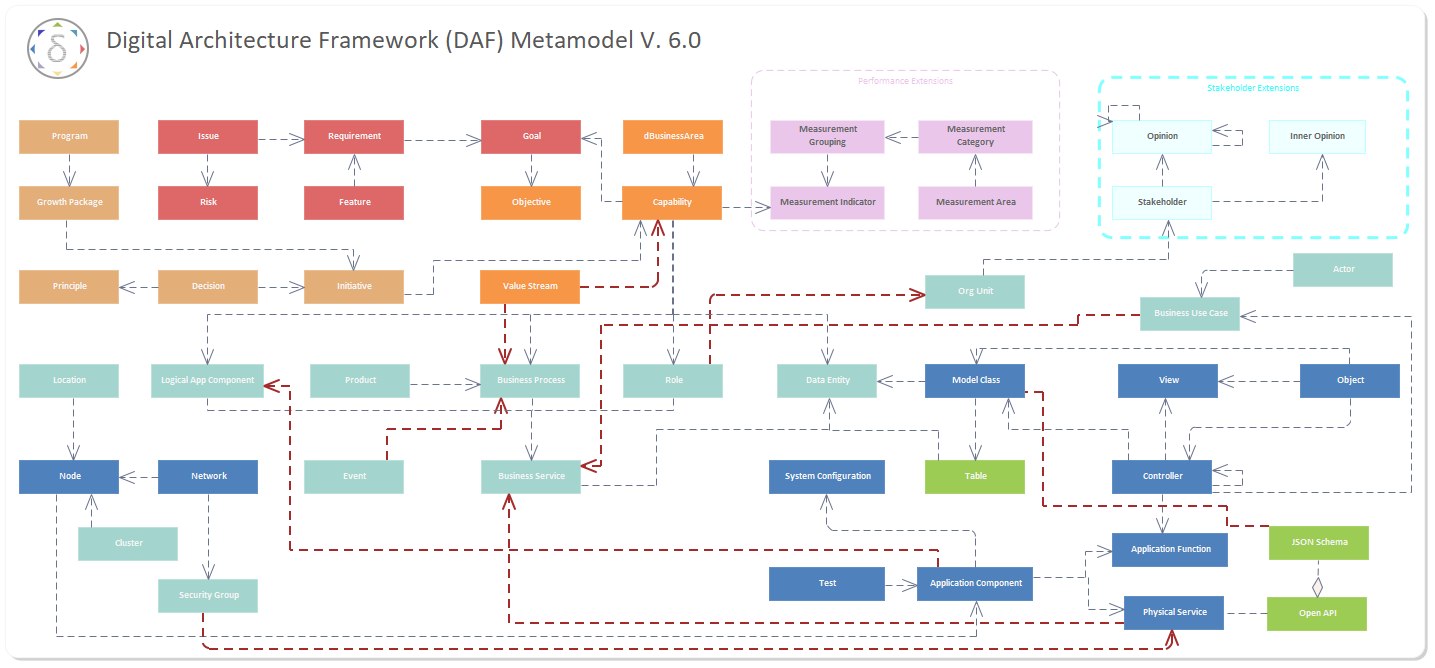
A "**Model Layer**" is a set of models that conform to a specific metamodel and are used to represent a specific aspect of the system being modeled. For example, one model layer could be used to represent the logical architecture of the system, while another could be used to represent the physical deployment of the system.

Figure : Relation between IT view and Enterprise

A "**Viewpoint**" is a set of views that have a common purpose or perspective. It can be related to specific domain, such as security or performance, and defines a specific set of views and model elements to be used to capture the relevant information from the system.

A "**View**" is a perspective or a specific part of a model that is relevant to a particular audience, such as stakeholders or developers. Each view provides a specific representation of the system, highlighting different aspects of the system and omitting the less important details.

## DAF Metamodel

Figure : DAF Metamodel

The DAF Metamodel supports Enterprise Architecture (Orange), Solution Architecture (blue) Design Architecture (Green). See the description of the 3 DAF layers below.

## Example Model Instance

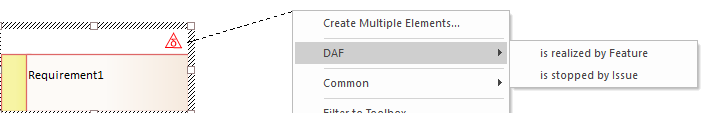
Each element of the Metamodel can be instantiated into a concrete model’s element.

The following diagram shows an example of a DAF Metamodel instance



Figure : Metamodel Instance

The Sparx EA quick linker helps the modeler create only the connections that are valid in the metamodel.



## Layers



Figure : DAF layers

The Digital Architecture Framework (DAF) is divided into three levels of abstraction: Enterprise, Solution, and Design Architecture. Each of these levels is further broken into specific domains.

### Enterprise Architecture

* This Layer covers logical considerations. Viewing the system in terms of roles participating in processes by using resources: Who, How, and What. It corresponds to the Computational Independent Model (CIM).
* Define enterprise frameworks, models, mechanisms, and structures so that capabilities of an enterprise evolve in a manner that effectively supports the business strategy
* Assisting the enterprise to plan and prioritize initiatives that optimize business & IT investments; designing initiatives that deliver an integrated landscape to reduces the fragmentation and variation that is the natural result of  delivering solutions in an autonomous manner.
* The Enterprise Architecture level focuses on the logical considerations of the system, such as the roles, processes, and resources involved. This abstraction level helps the enterprise plan and prioritize initiatives that optimize business and IT investments, and design initiatives that deliver an integrated landscape to reduce fragmentation and variation.

### Solution Architecture

* The Solution Architecture level translates the logical model into an operational one and describes the elements of the system that supports the business and assigns processes to them.
* The definition of solutions to business problems through the reasoned application of Information Technology.
* They may involve the application and integration of a broad variety of products, technologies and services, various IT systems and applications architectures, and diverse hardware and software components.

### Design Architecture

* This Layer designs the components of the solution, taking in account the different supporting technologies.
* The definition of the internal structure of the solution.
* Generation of Code
* This may be kept informal and is not being hosted in the Model

## DOMAINS

The abstraction layers are further organized into domains. A domain represents a vertical organization of concepts, incorporating related concepts at different levels of abstraction. These domains serve to organize and structure the various components and elements of the architecture in a logical and meaningful way.

The foundational domains in DAF are:

* Business
* Data
* Application
* Technology

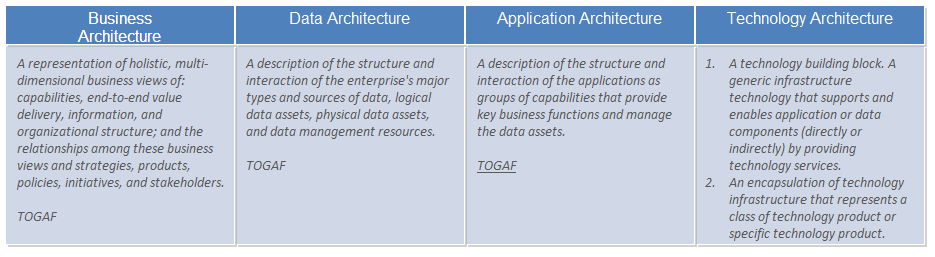


Figure : Foundational DAF domains, inspired by TOGAF

### Business

This domain covers the business-specific concepts, such as business processes, services, and functions that an organization offers to its customers. It focused on understanding the business objectives and strategies, and how the IT solutions can support those goals.

### Data

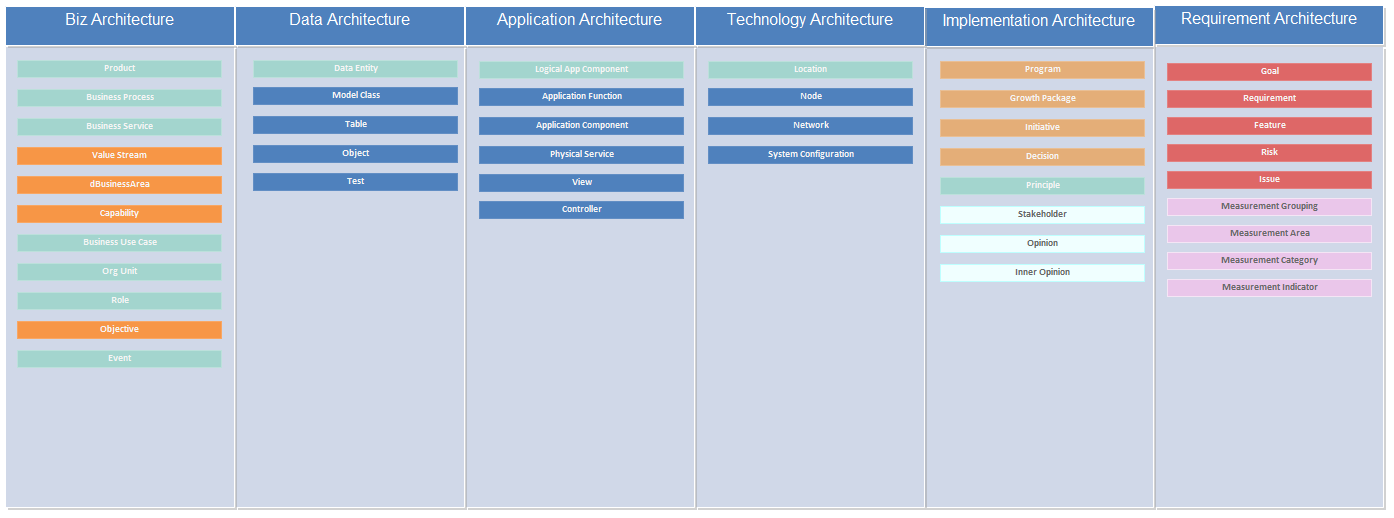
This domain covers data-related concepts, such as data models, data architecture, data governance, data security, and data management. This domain is focused on understanding the organization's data requirements, and how the IT solutions can support data management and access.

### Application:

This domain covers application-related concepts, such as application architecture, application integration, and service-oriented architecture. This domain is focused on understanding the organization's application requirements, and how the IT solutions can support application deployment and management.

### Technology

This domain covers technology-related concepts, such as infrastructure, network, security, and cloud. This domain is focused on understanding the organization's technology requirements, and how the IT solutions can support infrastructure and technology management.

Figure : Domains and concepts of DAF

### Extension Domains

Additional Domains can be also used with DAF when required.

Those include Strategy, Implementation and Requirements.

## DAF Viewpoints and Views

The intersection of a Layer and Domain creates a Viewpoint, addressing specific concerns about a system of interest. A Viewpoint is a way of looking at a system from a specific perspective and addressing specific concerns related to that system. In the context of digital engineering and DAF, a Viewpoint is the intersection of a Layer and a Domain.

Each Layer, such as Enterprise, Solution, and Design Architecture, provides a different level of abstraction and focus, while each Domain, such as Business, Data, Application, and Technology, represents a specific area of concern. By intersecting these two concepts, a Viewpoint is created that addresses specific concerns of the system-of-interest from the perspective of that Layer and Domain. For example, a Business Viewpoint within the Enterprise Layer would focus on the logical considerations of how the system supports the business strategy, while a Data Viewpoint within the Design Layer would focus on the design of the data components of the solution.

The use of Viewpoints allows for a more targeted and granular approach to analyzing and designing a system, as different stakeholders can look at the same system from different perspectives.

****

Figure : example of viewpoint and their stakeholders

Figure : Digital Viewpoint Model

**Views** are a representation of a system from that perspective as a related set of concerns. DAF features 12 viewpoints



Figure : DAF View Points

### Views and work products

DAF supports the SPEM concept of Deliverable.

A **View** resulting from an **Activity** performed by a **Role** can become **a work product** and form part of a **Deliverable**.

For example, in the diagram below , the Role of an Enterprise Architect is responsible for delivering the High-Level Design as a deliverable, which is the output of the "Create High-Level Design Document" activity.

This deliverable includes mandatory work products, such as the Component Diagram (As-Is/To-Be) and the Capability Business Model. When required, may also include proposed work products, such as the Platform Decomposition Diagram or the Process Diagram, depending on the project's scope.



Figure : SPEM View with the work products that compose the High Level Design deliverable

### DAF concepts Organized in Viewpoints

Each of the concepts of the Metamodel belong to a specific Viewpoint. As such it’s easier to assign responsibility to certain roles.



Figure : DAF concepts Organized in the Viewpoints

## Relationships

## A table with words and symbols AI-generated content may be incorrect.

Figure : Basic UML relationships

DAF relationships are extending the fundamental UML relationships:

* **Association**: A relationship that models a bi-directional semantic connection among instances
* **Aggregation**: A special kind of association that models a “whole-part” as shared relationship
* **Composition**: non-shareable, non-transferable ownership
* **Dependency**: A relationship that signifies that a single or a set of model elements requires other model elements
* **Generalization**: A taxonomic relationship between a more general element and a more specific element
* **Realization**: A specialized abstraction relationship between two sets of model elements, one representing a specification (the supplier) and the other represents an implementation of the latter (the client)

In the DAF metamodel, inheritance uses regular Generalizations, transitions of abstraction layers are represented with stereotyped Realization, strong connections between elements of the same abstraction level with stereotyped **Associations**, weak relationships with Dependencies; strong hierarchical relationships with **Compositions**, weak ones with **Aggregations.**

* A white and orange sheet with black text

  AI-generated content may be incorrect.

Figure : Additional Archimate relationships

### Relationship Strategy

The DAF Metamodel is designed to support the concept of **encapsulation**.

Encapsulation is a key concept in modeling that refers to the practice of enclosing concept and behaviors within a single entity (in this case a viewpoint), so as to protect its internal state from external interference. This helps to maintain the integrity of the model by controlling the flow of information and avoiding circular references. In object-oriented modeling, encapsulation allows objects to interact with one another without being dependent on their implementation, creating a level of abstraction that makes the system more flexible and maintainable.

**Circular references** in modeling refer to the situation where two or more models or elements within a model refer to each other in a circular manner. This creates a situation where one element depends on another, which in turn depends on the first, leading to a never-ending chain of inter-dependencies. Circular references are often considered problematic in modeling because they can lead to ambiguities and inconsistencies in the models, making it difficult to analyze and understand the relationships between elements. They can also cause problems with automated analysis and interpretation tools that rely on the models.

Instead of circular references, **derived relationships** are used.

A derived relationship refers to a relationship that is obtained or computed from other relationships or information in the model. It is not directly specified in the model but is instead derived from existing elements and their properties and relationships. Derived relationships are often used to simplify a model and make it more manageable by abstracting complex relationships and showing only the essential relationships between elements. These relationships may also be used to generate additional information about the system being modeled or to support certain modeling activities such as impact analysis, traceability, and consistency checks.

# Detailed Documentation

The following is the complete documentation of the DAF metamodel

## DAF Concepts Overview



Figure : DAF concepts and Properties

DAF version 6 supports ~71 concepts (implemented as UML Stereotypes), each of those has several properties (expressed as UML Tagged Values)

| Domain | Stereotype | Name | Documentation |
| --- | --- | --- | --- |
| Application | dTest | Test | Represents a mechanism for verifying or validating the functionality, behavior, or compliance of am application component . Tests can include automated or manual procedures designed to ensure alignment with requirements, detect defects, and confirm expected outcomes. |
| Application | dModelClass | Model Class | Represent a class of the domain model, that can be converted into a Persistent domain class, must inherit from Node. It is the application's dynamic data structure |
| Application | dLogicalAppComponent | Logical App Component | Encapsulates logical software components independent of implementation or technology. |
| Application | dApplicationFunction | Application Function | Encapsulates functions supported by applications. |
| Application | dController | Controller | Encapsulates application behavior and flow logic. |
| Application | dValue | Value | An attribute of a Node value type used in DAF. These values are persistent |
| Application | dApplicationComponent | Application Component | An application component is a self-contained unit. As such, it is independently deployable, re-usable, and replaceable. An application component performs one or more application functions. It encapsulates its behavior and data, exposes services, and makes them available through interfaces. Cooperating application components are connected via application collaborations.  An application component may be assigned to one or more application functions. An application component has one or more application interfaces, which expose its functionality. Application interfaces of other application components may serve an application component. The name of an application component should preferably be a noun. |
| Application | dView | View | A View used in DAF. Controller, Views and Associations define the application flow |
| Application | dPhysicalService | Physical Service | a technical implementation of a service that runs on physical or virtual infrastructure, providing specific functionalities, usually in the form of software applications, middleware, databases, or platforms. It is concerned with the deployment, operation, and management of the technology stack, ensuring that the service is available, performant, and reliable. Examples include web servers, APIs, databases, and network services. |
| Application | dValueRef | Value Ref. | A Node value type used in DAF that reference another Node Value, similar to a foreign Key. These values are persistent |
| Business | dRole | Role | The usual or expected function of an actor, or the part somebody or something plays in a particular action or event. An actor may have a number of roles. |
| Business | dActivity | Activity | Represents business or system activities. A function or work carried out by the business user. |
| Business | dActor | Actor | Represents an entity (such as a person, system, or organization) that interacts with the system to achieve a specific goal. An actor typically performs actions within business Use cases and can be internal (e.g., employees) or external (e.g., customers or partner organizations), active and passive. |
| Business | dBusinessUseCase | Business Use Case | Encapsulates business processes producing value for an enterprise. |
| Business | dBusinessService | Business Service | A system configuration. Settings are modeled as attributes with default values |
| Business | dAction | Action | A specialized activity. |
| Business | dEvent | Event | A behavioral situation that causes a change in the state of an element (typically a Process) |
| Business | dBusinessProcess | Business Process | A sequence of activities that produce a specific service or product supporting a particular business capability. |
| Data | dTable | Table | Represents physical tables in a database schema. |
| Data | dObject | Object | an Object is an instance of a class in a particular moment in runtime that can have its own state and data values. |
| Implementation | dDecision | Decision | Represents decisions and their impact on architecture or processes. |
| Implementation | dUserStory | User Story | Captures user-centric requirements expressed as stories. |
| Implementation | dInitiative | Initiative | An activity carried out by a Business Unit or organization to achieve a particular goal or improve the Maturity of a certain capability. |
| Implementation | dGrowthPackage | Growth Package | A set of actions identified to achieve one or more objectives for the business. A work package can be a part of an Initiative, a complete project, or a part of a Program. A Growth pack is a collection of Initiatives architected for consistency and aimed to be executed in the same period. |
| Implementation | dProgram | Program | A group of Growth Packages that achieve a complex outcome for the organization |
| Requirement | dRisk | Risk | Identifies and assesses potential issue impacting other elements. A risk is a potential future Issue. |
| Requirement | dFeature | Feature | A Feature is a fact that set his related Requirement to be true |
| Requirement | dIssue | Issue | A concrete problem that is affecting a business. Also, a fact that sets a Requirement to be false. (see also Risk) |
| Requirement | dRequirement | Requirement | A Business guide line about the Enterprise or the project. A requirement is formulated in a SMART fashion and uses Moscow verbs (Must Have, Should Have, Could Have, and Would Like to Have.). Requirements can be decomposed.   * Must Have: These are the requirements that are considered essential for the success of the project and must be met for the system to function. They are often referred to as "hard" or "mandatory" requirements and are typically expressed in Boolean format as "true" or "1". * **Should Have**: These are the requirements that are considered important, but not essential. They are typically expressed in Boolean format as "maybe" or "0/1" and are often used to prioritize the development of the system. * **Could Have**: These are the requirements that are considered desirable but not essential. They are typically expressed in Boolean format as "false" or "0". They are used to identify potential enhancements to the system that can be added at a later date. * **Would Like to Have**: These are the requirements that are considered less important and are often used to identify potential future enhancements to the system.   In addition, the requirement should be SMART, which stands for Specific, Measurable, Achievable, Relevant, and Time-bound. SMART requirements are clear, concise, and easy to understand, and they provide a clear way to measure the success of the project. |
| Requirement | dGoal | Goal | A **measurable** scope that the organization wants to achieve. Can be hierarchically decomposed. |
| Strategy | dMeasurementIndicator | Measurement Indicator | The specific measure captured for a Organization or Business Unit which can be measured and quantifiable. A measurement is a Key Performance indicator |
| Strategy | dMeasurementArea | Measurement Area | Represents areas of organizational or business performance measurement. The organization or Business units’ performance objective is captured as measurement area. |
| Strategy | dStakeholder | Stakeholder | Represents any party or force influencing the domain of interest. Any person or force that have an interest in the considered domain. This is typically indicated by name, can be connected with a Role in the organization. A stakeholder has open and Inner opinions that are relevant for the subject matter. |
| Strategy | dCapability | Capability | Represents an organizational capability to achieve business outcomes. Can be current available in an Organization Unit (or) A business-focused outcome that is delivered by the completion of one or more work packages (by increasing the capability Maturity). |
| Strategy | dProduct | Product | Encapsulates outputs produced by business processes. |
| Strategy | dPrinciple | Principle | Defines qualitative statements guiding architecture intent. |
| Strategy | dOrganizationUnit | Org Unit | Defines organizational units responsible for roles, can be decomposed |
| Strategy | dMeasurementCategory | Measurement Category | Categorizes measurements within a measurement area. |
| Strategy | dOpinionInner | Inner Opinion | Captures implicit stakeholder opinions related to a topic. A Stakeholder’s opinion that is relevant for the topic but is not formulated explicitly. |
| Strategy | dSkill | Skill | an ability that some role possesses with various degree |
| Strategy | dOpinion | Opinion | A view or judgment formed about a Topic. That is formulated in an explicit fashion. Captures explicit stakeholder judgments on a topic. |
| Strategy | dResource | Resource | a type of Asset supporting a Business Capability |
| Strategy | dObjective | Objective | A time-bounded **milestone** for an organization used to demonstrate progress towards a goal. The Objective is a non material achievement, other than a Goal, the objective has no specific value to be measured |
| Strategy | dValueStream | Value Stream | A representation of an end-to end collection of value-adding Business Processes that create an overall result for a customer, stakeholder or end user |
| Strategy | dLocation | Location | A place where business activity takes place and can be hierarchically decomposed. |
| Strategy | dMeasurementGrouping | Measurement Grouping | a group of Measurement Indicator that can be aggregated into Categories |
| Technology | dJSON\_Schema | JSON Schema | The Schema Object allows the definition of input and output data types. These types can be objects, but also primitives and arrays. This object is based on the JSON Schema Specification Draft 4 and uses a predefined subset of it. On top of this subset, there are extensions provided by this specification to allow for more complete documentation |
| Technology | dDeploymentModel | Deployment Model | The root class for the description of a Deployments to roll out an application, service, or container to a set of resources. akin to ArchiMate's Technology Layer deployment views. |
| Technology | dNetwork | Network | The main communication medium between services and deployments, mapped to ArchiMate Communication Networks, facilitating inter-service communication. |
| Technology | dJSON\_SchemaSubSet | JSON Schema Sub Set | Represents subsets of JSON schema structures. |
| Technology | dAPIResponse | API Response | Represents responses returned by API operations. |
| Technology | dJSON\_Element | JSON Element | Represents elements or nodes in JSON schemas. |
| Technology | dJSON\_Type | JSON Type | Represents data type definitions for structured information. |
| Technology | dCluster | Cluster | Represents a set of nodes working in the same Region, akin to ArchiMate’s Node concept within a server infrastructure. |
| Technology | dDeploymentNode | Node | Represents a physical machine or virtual server that hosts services, applications, or containers, akin to ArchiMate’s Node concept in the Technology Layer. |
| Technology | dJSON\_Datatype | JSON Data Type | Represents a data type used within JSON structures, defining the format and constraints of data elements in a schema. JSON Data Types include standard types such as string, number, object, array, boolean, and null, ensuring consistency and validation in data exchange between systems. |
| Technology | dVolume | Volume | Refers to the storage assigned to containers, ensuring persistence, aligning with the Data Object in ArchiMate’s Application Layer. |
| Technology | dAPI | Open API | OpenAPI Specification is an open-source format and initiative for designing and creating machine-readable interface files that are utilized in producing, describing, consuming, and visualizing RESTful APIs and web services. |
| Technology | dContainer | Container | Encapsulates application components and services, mirroring ArchiMate's concept of a Node, where an application runs. |
| Technology | dAPIParameter | API Parameter | Defines parameters in API operations. |
| Technology | dAPIOperation | API Operation | Defines operations exposed by an API for interaction with external systems. Also called endpoints. |
| Technology | dZone | Zone | Defines macro divisions containing regions mapped to ArchiMate’s Infrastructure Service. |
| Technology | dJSON\_Attribute | JSON Attribute | Defines attributes in JSON data structures. |
| Technology | dIPRange | IP Range | Defines a range of IP addresses (CIDR) used within a network. This aligns with ArchiMate's concept of Infrastructure Service, providing connectivity. |
| Technology | dSecurityGroup | Security Group | Controls and manages access and permissions between services, in line with TOGAF's Security Architecture for protecting digital assets. |
| Technology | dOnPremise | On Premise Node | an organization of Business interests |
| Technology | dDataEntity | Data Entity | An encapsulation of data that is recognized by a business domain expert as a thing. Logical data entities can be tied to applications, repositories, and services and may be structured according to implementation considerations. |
| Technology | dSystem | System Configuration | A system configuration. Settings are modeled as attributes with default values; each attribute of the System Config becomes a line in the configuration file. e.g. SystemConfig .path = c:/users/app |
| Technology | dSubNetwork | Sub Network | A subset of a network, used to organize and isolate portions of the network. Equivalent to an ArchiMate Node, responsible for networking resources. |
| Technology | dPublicCluster | Public Cluster | A public cloud-based cluster, following the ArchiMate concept of Infrastructure Service provided by external cloud providers (Google, AWS, etc.). |
| Technology | dRegion | Region | A geographic area containing clusters and zones, aligned with ArchiMate's Location concept for representing deployment geography. |

## Relationship Types

|  |  |  |  |
| --- | --- | --- | --- |
| Technical name | Human Name | Destination | Source |
| dAccord | According Opinion | is in accord with other Opinion | is in accord with original Opinion |
| dActionKey | Action Key | target Controller | Source Controller |
| dActivityContainsAction | Activity contains Action | is contained in Activity | contains Action |
| dActorParticipatesInBizUseCase | Actor Participates In Use Case | Partecipates in Use Case | is executed by Actor |
| dApplicationAgregatesFunction | aggregates Function | aggregates Function | part of Application |
| dApplicationComponentRealizesogicalComponent | Realizes Component | Realizes Logical component | is Realized by Component |
| dApplicationconfiguredbySystemConfiguration | Application configured by System Configuration | has configuration | configures Application |
| dApplicationConnectedWithApplication | Connects Application | connected From Application | Connected to Application |
| dApplicationExposesService | Application Exposes Ph. Service | is exposed by Application | Exposes Phy Service |
| dApplicationHasOwner | Application has Owner | owns Application | has Owner |
| dApplicationIRunsInNode | Application is run By Node | Runs Application | is run By Node |
| dApplicationRisk | Application Risk | is related to Application | has Risk |
| dAreaAggregatesCategories | Area aggregates Categories | aggregates categories | has Measurement Area |
| dBizServIsDescribedByUseCase | Biz Serv Is Described By Use Case | Has service | described by Use Case |
| dBizServUsesDataEntity | Service Uses Data | uses Data | is used by service |
| dBusinessAreadHasCapability | has Capability | has Capability | is organized in Biz Area |
| dCapabilityisMeasuredByKPI | is Measured By KPI | is Measured By Indicator | is measuring Capability |
| dCategoryaggregatesGroups | aggregates Groups | aggregates Groups | has Category |
| dContainerAggregatesVolume | Container Aggregates Volume | Aggregates Volume | is Aggregated in Container |
| dControllerControllsModel | Controlls Model | Controlls Model | is controlled by Controller |
| dControllerGovernsView | Governs View | Governs view | is Governed by Controller |
| dControllerImplementsFunction | controller Implements function | Implements function | has Controller |
| dDataEntityInformsCapability | informs Capability | is informed by Data Entity | Informs Capability |
| dDecisionRefersToPrinciple | Decision refers To Principle | refers To Principle | is referenced by Decision |
| dDiscord | Discording opinion | is in discord with other Opinion | is in discord with Original Opinion |
| dEntityAssociatesEntity | Entity Associates Entity | Is connected to Source Entity | Is connected to Target Entity |
| dEntityisFunctionallyImplementedByTable | Entity is Functionally Implemented By Table | Implements Entity | is Functionally Implemented By Table |
| dEventTriggersProcess | Event Triggers Process | triggers Process | is triggered by Event |
| dFeatureIsRealizedByUseCase | Feature is realized by Use Case | realizes Feature | is realized by Use Case |
| dGoalAggregatesGoal | Goal Aggregates Goal | parent Goal | Child Goal |
| dGoalhasObjective | Goal has Milestone | Has Objective | is related to Goal |
| dGoalisOperationalizedByCapability | goal is Operationalized By Capability | support Goal | is Operationalized By Capability |
| dGroupAggregatesKPI | Group aggregates KPI | aggregates KPI | part of a Grouping |
| dInitiativeIncreasesMaturityOf | Initiative Increases Maturity Of Capability | Increases Maturity Of Capability | is matured by Initiative |
| dInitiativeisMakingUseOfDecision | Initiative is Making Use Of Decision | is used in Initiative | Makes Use Of Decision |
| dIssueStopsRealizationOfRequirement | Issue Stops Realization Of Req | stop Requirement | is stopped by Issue |
| dLocationHostsNode | Location Hosts Node | Hosts Node | Is hosted in Location |
| dLogAppCompExposesBizService | exposes Biz Service | application runs service | service run by application |
| dLogicalApplicationComponentSupportsCapability | App Supports Capability | supported by Application | supports Capability |
| dModelClassRealizesEntity | Model Class Realizes Entity | Realizes Entity | is realized by Class |
| dNetworkContainsNode | Contains Node | Contains Node | is contained in a Network |
| dNetworkHasSecurityGroup | Network Has Security Group | Has Security Group | is associated to Network |
| dObjectInstanceofController | Instance of Controller | Has an Instance | Istance of Controller |
| dObjectInstanceofModel | Instance of Model | Has an Instance | Istance of Model |
| dObjectInstanceofView | Instance of View | Has an Instance | Istance of View |
| dOrgUnitHasStakeholders | Org Unit has Stakeholders | has Stakeholder | is concerned with Organization |
| dPackageIsDeliveredByInitiative | Package is delivered by Initiative | is delivered by Initiative | is part of a Package |
| dPhysicalServiceImplementsBusinessService | Physical Serv. Implements Biz Serv. | Implement Biz Service | is Implemented by physical service |
| dProcessEnsureCorrectOperationOfCapability | Ensure the correct Operation of | is operated by Process | Ensure the correct Operation of Capability |
| dProcessFlowsToProcess | Process Flows to Process | Sequence Flow To | Sequence Flow From |
| dProcessOrchestratesService | Process Orchestrates Service | Has Service | part of Process |
| dProductIncludesProcess | Product Includes Process | Includes Process | Is part of Product |
| dProgramIsOrganizedInPackages | Program is organized In Packages | is organized In Packages | part of a Program |
| dRelatedService | Child Process | Target Service | Source Service |
| dRequirementContainsRequirement | Requirement Contains Req | is contained by Requirement | Contains Requirement |
| dRequirementIsRelatedToGoal | req related to Goal | specifies Goal | is specified by Requirement |
| dRequirementIsSatisfiedByFeature | is Satisfied By | realizes Requirement | is realized by Feature |
| dResourceimpactedByRisk | Resource impacted by Risk | impact Resouce | is impacted by Risk |
| dResourceSupportsCapability | Resource supports Capability | Supports Capability | is supported By Resource |
| dRiskIRelatedToRisk | Risk is related to Risk | target Risk | Source Risk |
| dRiskRealizedByIssue | realized by Issue | is realizing Risk | is realized by Issue |
| dRoleConsumesService | Role Consumes Service | Consumes Service | is consumed by Role |
| dRoleExecutesCapability | role Executes Capability | is executed by Role | executes Capability |
| dRoleHasSkill | Role has skill | has SKill | known by Role |
| dRoleisAssociatedToOrganization | Role is Associated To Organization | works in Organization | has Role |
| dRoleIsPerformedByActor | Role is Performed By Actor | has Role | is executed by Actor |
| dStakeholderHasInnerOpinion | Stakeholde Has Inner Opinion | Has Inner Opinion | is thinked by Stakeholder |
| dStakeholderHasOpinion | Stakeholder Has Opinion | Has Opinion | is expressed by Stakeholder |
| dTableProvidesPersistenceForModelClass | Table Provides Persistence to Model | is Saved in a Table | Provides Persistence For Model |
| dTest\_ApplicationComponent | Test validates Component | Test validates Component | is validated by test |
| dUseCaseContainsActivities | Use Case Contains Activities | has Activitiy | part of UseCase |
| dUseCaseHasStory | Use Case Has Story | Is associated to Use Case | Can be told as User Story |
| dUseCaseisImplementedByController | Use Case is Implemented B y Controller | implements Use Case | Is implemented by Controller |
| dValueStreamIncludesProcesses | Value Stream Includes Processes | Includes Process | is part of Value Stream |
| TMF\_isComposedByProcess | is Composed By Process | is Composed By Process | is Composed within Process |
| TMF\_ProcessProducesEntity | Produces Entity | Process produces Entity | Entity is produced by process |
| TMF\_ProcessUsesEntity | Process Uses Entity | Uses Entity | is used by Process |
| TMF\_RelatedUseCase | Related Use Case | target related Use Case | related Use Case |
| TMF\_ThemeComposesUserStory | Composes User-Story | Composes User-Story | is composed In Theme |
| TMF\_UseCaseAssociatedWithTestCase | Associated With Test Case | Associated With Test Case | Associated With Use-Case |
| TMF\_Use-CaseDependOnTool | Depends On Tool | Depends On Tool | Is depended of Use-Case |
| TMF\_Use-CaseDependsOnPolicy | Depends On Policy | Depends On Policy | is dependent of Use-Case |
| TMF\_Use-CaseDependsOnUser-Story | Depends On User-Story | Depends On Use-Case | Depends On User-Story |
| TMF\_UseCaseRelatedProcess | Related Process | related to Process | related to Use Case |
| TMF\_ValueStreamAggregatesValueStage | Aggregates Value Stage | Aggregates Value Stage | is aggregated by Value Stream |
| TMF\_VerticalAggregatesProcess | Vertical Aggregates Process | Aggregates Process | Is aggregated in Verticals |
| TMF\_VerticalComposedVertical | Vertical Composes Vertical | Child Vertical | Parent Vertical |

### Relationships Quick link

|  |
| --- |
| dActivity -> dActivityContainsAction -> dAction |
| is contained in Activity |
| contains Action |
| dActor -> dActorParticipatesInBizUseCase -> dBusinessUseCase |
| Partecipates in Use Case |
| is executed by Actor |
| dActor -> dRoleIsPerformedByActor -> dRole |
| has Role |
| is executed by Actor |
| dAPI -> dAPIAggregatesSchema -> dJSON\_Schema |
| Aggregates Schema |
| is aggregates in API |
| dAPI -> -> dPhysicalService |
| dApplicationComponent -> dApplicationComponentRealizesogicalComponent -> dLogicalAppComponent |
| Realizes Logical component |
| is Realized by Component |
| dApplicationComponent -> dApplicationconfiguredbySystemConfiguration -> dSystem |
| has configuration |
| configures Application |
| dApplicationComponent -> dApplicationExposesService -> dPhysicalService |
| is exposed by Application |
| Exposes Phy Service |
| dApplicationComponent -> dApplicationHasOwner -> dStakeholder |
| owns Application |
| has Owner |
| dApplicationComponent -> dApplicationConnectedWithApplication -> dApplicationComponent |
| connected From Application |
| Connected to Application |
| dApplicationComponent -> dApplicationAgregatesFunction -> dApplicationFunction |
| aggregates Function |
| part of Application |
| dApplicationComponent -> dApplicationComponentHasContainer -> dContainer |
| Has Container |
| contains Application |
| dBusinessProcess -> dProcessOrchestratesService -> dBusinessService |
| Has Service |
| part of Process |
| dBusinessProcess -> dProcessFlowsToProcess -> dBusinessProcess |
| Sequence Flow To |
| Sequence Flow From |
| dBusinessService -> dBizServUsesDataEntity -> dDataEntity |
| uses Data |
| is used by service |
| dBusinessService -> dRelatedService -> dBusinessService |
| Target Service |
| Source Service |
| dBusinessUseCase -> dBizServIsDescribedByUseCase -> dBusinessService |
| Has service |
| described by Use Case |
| dBusinessUseCase -> dUseCaseContainsActivities -> dActivity |
| has Activitiy |
| part of UseCase |
| dBusinessUseCase -> dFeatureIsRealizedByUseCase -> dFeature |
| realizes Feature |
| is realized by Use Case |
| dCapability -> dProcessEnsureCorrectOperationOfCapability -> dBusinessProcess |
| is operated by Process |
| Ensure the correct Operation of Capability |
| dCapability -> dCapabilityisMeasuredByKPI -> dMeasurementIndicator |
| is Measured By Indicator |
| is measuring Capability |
| dCapability -> dRoleExecutesCapability -> dRole |
| is executed by Role |
| executes Capability |
| dCapability -> dGoalisOperationalizedByCapability -> dGoal |
| support Goal |
| is Operationalized By Capability |
| dCapability -> dLogicalApplicationComponentSupportsCapability -> dLogicalAppComponent |
| supported by Application |
| supports Capability |
| dCapability -> dDataEntityInformsCapability -> dDataEntity |
| is informed by Data Entity |
| Informs Capability |
| dCluster -> dClusterContainsDeploymentNode -> dDeploymentNode |
| Contains DeploymentNode |
| is Contained in Cluster |
| dContainer -> dContainerAggregatesVolume -> dVolume |
| Aggregates Volume |
| is Aggregated in Container |
| dController -> dControllerImplementsFunction -> dApplicationFunction |
| Implements function |
| has Controller |
| dController -> dControllerGovernsView -> dView |
| Governs view |
| is Governed by Controller |
| dController -> dControllerControllsModel -> dModelClass |
| Controls Model |
| is controlled by Controller |
| dController -> dUseCaseisImplementedByController -> dBusinessUseCase |
| implements Use Case |
| Is implemented by Controller |
| dController -> dActionKey -> dController |
| target Controller |
| Source Controller |
| dDataEntity -> dEntityAssociatesEntity -> dDataEntity |
| Is connected to Source Entity |
| Is connected to Target Entity |
| dDecision -> dInitiativeisMakingUseOfDecision -> dInitiative |
| is used in Initiative |
| Makes Use Of Decision |
| dDecision -> dDecisionRefersToPrinciple -> dPrinciple |
| refers To Principle |
| is referenced by Decision |
| dDeploymentModel -> dDeploymentModelAggregatesCluster -> dCluster |
| Aggregates Cluster |
| is Aggregated in Deployment Model |
| dDeploymentNode -> dApplicationIRunsInNode -> dApplicationComponent |
| Runs Application |
| is run By Node |
| dEvent -> dEventTriggersProcess -> dBusinessProcess |
| triggers Process |
| is triggered by Event |
| dFeature -> dRequirementIsSatisfiedByFeature -> dRequirement |
| realizes Requirement |
| is realized by Feature |
| dGoal -> dGoalhasObjective -> dObjective |
| Has Objective |
| is related to Goal |
| dGoal -> dGoalAggregatesGoal -> dGoal |
| parent Goal |
| Child Goal |
| dGrowthPackage -> dPackageIsDeliveredByInitiative -> dInitiative |
| is delivered by Initiative |
| is part of a Package |
| dInitiative -> dInitiativeIncreasesMaturityOf -> dCapability |
| Increases Maturity Of Capability |
| is matured by Initiative |
| dIssue -> dIssueStopsRealizationOfRequirement -> dRequirement |
| stop Requirement |
| is stopped by Issue |
| dIssue -> dRiskRealizedByIssue -> dRisk |
| is realizing Risk |
| is realized by Issue |
| dJSON\_Element -> dJSONElement\_associates\_JSONElement -> dJSON\_Element |
| target JsonElement |
| Source JsonElement |
| dJSON\_Schema -> dPolymorphicCollection -> dJSON\_Schema |
| @baseType |
| @Type |
| dJSON\_Schema -> dJSON\_SchemaSubSetGeneralizes\_JSON\_Schema -> dJSON\_SchemaSubSet |
| Parent Schema |
| Child Schema |
| dJSON\_Schema -> dSchema\_Associates\_Schema -> dJSON\_Schema |
| Target Schema |
| Source Schema |
| dJSON\_Schema -> dSchemaGeneralizesSchema -> dJSON\_Schema |
| Child Schema |
| Parent Schema |
| dJSON\_Schema -> dSchemaAssociatesElement -> dJSON\_Element |
| Associates Element |
| Associates Schema |
| dJSON\_Schema -> dJSONSchemaRealizesModel -> dModelClass |
| realizes Model Class |
| is realized by Schema |
| dLocation -> dLocationHostsNode -> dDeploymentNode |
| Hosts Node |
| Is hosted in Location |
| dLogicalAppComponent -> dLogAppCompExposesBizService -> dBusinessService |
| application runs service |
| service run by application |
| dMeasurementArea -> dAreaAggregatesCategories -> dMeasurementCategory |
| aggregates categories |
| has Measurement Area |
| dMeasurementCategory -> dCategoryaggregatesGroups -> dMeasurementGrouping |
| aggregates Groups |
| has Category |
| dMeasurementGrouping -> dGroupAggregatesKPI -> dMeasurementIndicator |
| aggregates KPI |
| part of a Grouping |
| dModelClass -> dModelClassRealizesEntity -> dDataEntity |
| Realizes Entity |
| is realized by Class |
| dModelClass -> dTableProvidesPersistenceForModelClass -> dTable |
| is Saved in a Table |
| Provides Persistence For Model |
| dNetwork -> dNetworkContainsNode -> dDeploymentNode |
| Contains Node |
| is contained in a Network |
| dNetwork -> dNetworkHasSecurityGroup -> dSecurityGroup |
| Has Security Group |
| is associated to Network |
| dNetwork -> dNetworkAggregatesSubNetwork -> dSubNetwork |
| Aggregates Sub-Network |
| is Aggregated in Network |
| dObject -> dObjectInstanceofModel -> dModelClass |
| Has an Instance |
| Istance of Model |
| dObject -> dObjectInstanceofController -> dController |
| Has an Instance |
| Istance of Controller |
| dObject -> dObjectInstanceofView -> dView |
| Has an Instance |
| Istance of View |
| dOpinion -> dDiscord -> dOpinion |
| is in discord with other Opinion |
| is in discord with Original Opinion |
| dOpinion -> dAccord -> dOpinion |
| is in accord with other Opinion |
| is in accord with original Opinion |
| dOrganizationUnit -> dOrgUnitHasStakeholders -> dStakeholder |
| has Stakeholder |
| is concerned with Organization |
| dPhysicalService -> dPhysicalServiceImplementsBusinessService -> dBusinessService |
| Implement Biz Service |
| is Implemented by physical service |
| dProduct -> dProductIncludesProcess -> dBusinessProcess |
| Includes Process |
| Is part of Product |
| dProgram -> dProgramIsOrganizedInPackages -> dGrowthPackage |
| is organized In Packages |
| part of a Program |
| dRegion -> dRegionAggregatesCluster -> dCluster |
| Aggregates Cluster |
| is Aggregated in Region |
| dRequirement -> dRequirementIsRelatedToGoal -> dGoal |
| specifies Goal |
| is specified by Requirement |
| dRequirement -> dRequirementContainsRequirement -> dRequirement |
| is contained by Requirement |
| Contains Requirement |
| dResource -> dResourceSupportsCapability -> dCapability |
| Supports Capability |
| is supported By Resource |
| dRisk -> dRiskIRelatedToRisk -> dRisk |
| target Risk |
| Source Risk |
| dRisk -> dResourceimpactedByRisk -> dResource |
| impact Resouce |
| is impacted by Risk |
| dRisk -> dApplicationRisk -> dApplicationComponent |
| is related to Application |
| has Risk |
| dRole -> dRoleisAssociatedToOrganization -> dOrganizationUnit |
| works in Organization |
| has Role |
| dRole -> dRoleConsumesService -> dBusinessService |
| Consumes Service |
| is consumed by Role |
| dRole -> dRoleHasSkill -> dSkill |
| has SKill |
| known by Role |
| dSecurityGroup -> dSecurityGroupHasPhysicalService -> dPhysicalService |
| has Physical Service |
| Asociated to Sec. group |
| dStakeholder -> dStakeholderHasInnerOpinion -> dOpinionInner |
| Has Inner Opinion |
| is thinked by Stakeholder |
| dStakeholder -> dStakeholderHasOpinion -> dOpinion |
| Has Opinion |
| is expressed by Stakeholder |
| dSubNetwork -> dSubNetworkAggregatesIPRange -> dIPRange |
| Aggregates IP Range |
| is Aggregated in sub Net. |
| dTable -> dEntityisFunctionallyImplementedByTable -> dDataEntity |
| Implements Entity |
| is Functionally Implemented By Table |
| dTest -> dTest\_ApplicationComponent -> dApplicationComponent |
| Test validates Component |
| is validated by test |
| dUserStory -> dUseCaseHasStory -> dBusinessUseCase |
| Is associated to Use Case |
| Can be told as User Story |
| dValueStream -> dValueStreamIncludesProcesses -> dBusinessProcess |
| Includes Process |
| is part of Value Stream |
| dValueStream -> dValueStreamEnablesCapability -> dCapability |
| Enables Capability |
| is Enabled by Value Stream |
| dZone -> dZoneAggregatesRegion -> dRegion |
| Aggregates Region |
| is Aggregated in Region |

## Visual Characteristics of Special stereotypes

Some DAF stereotypes have special visual characteristics. They can react to Tagged Values or other special properties, like diagram properties as shown below.

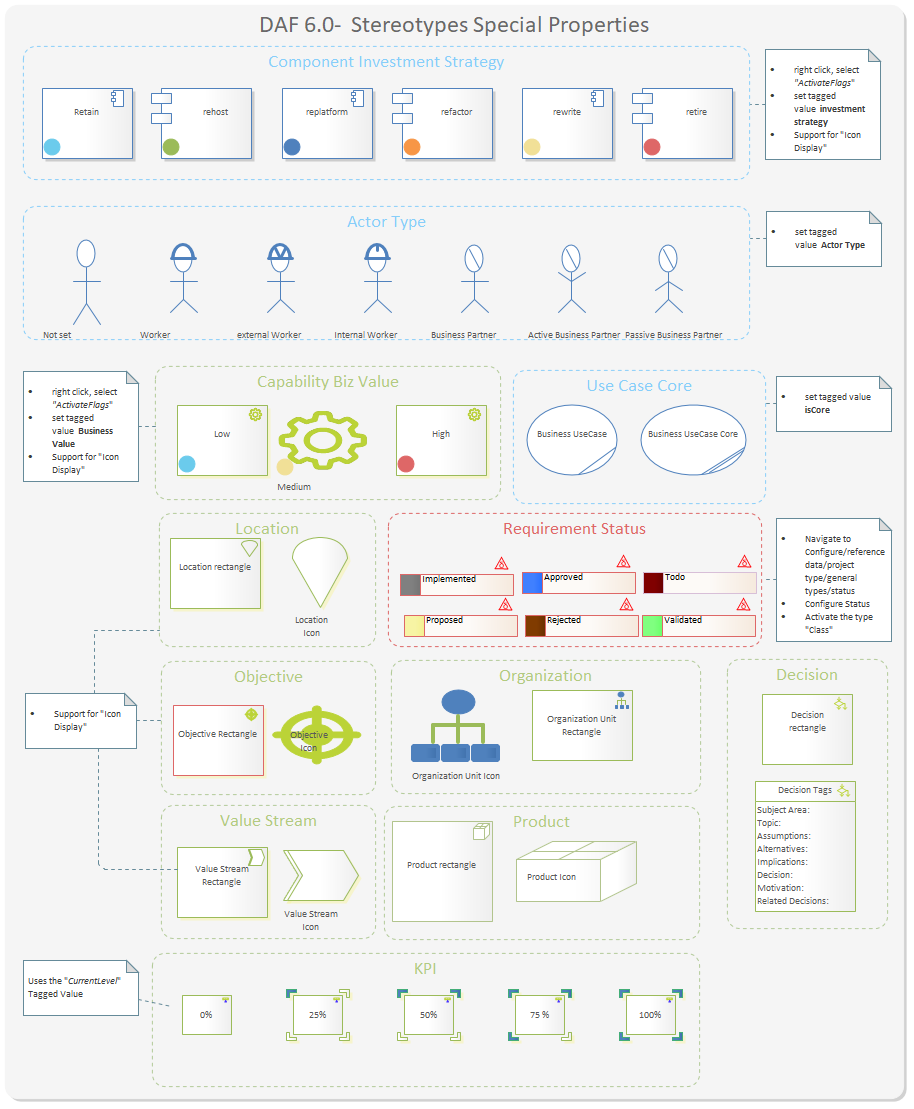


Figure : DAF visual properties

The metamodel uses certain colors to indicate domain of architecture

//blue : 107**,**203**,**236

**Enterprise ARCHITECTURE**

//Green: 155**,**187**,**89

**SOLUTION ARCHITECTURE**

//dark blue: **(**79**,**129**,**189

**DESIGN ARCHITECTURE**

//Orange**:** 247**,**150**,**70

**Strategy**

**//Dark Orange: 233,191,149**

**Performance**

**//violet: 234,198,234**

**Requirements**

**//red: 222,103,103**

**Requirements**

**//red: 222,103,103**

**Stakeholder**

**//light blue: 240,255,255**

**Technology**

**//Green: 156,204,84**

**COMMON**

//yellow**:** 241**,**224**,**151**);**

**Color in HEX**

*Blue: 107,203,236 → #6BCBEC*

*- Green: 155,187,89 → #9BBB59*

*- Dark Blue: 79,129,189 → #4F81BD*

*- Orange: 247,150,70 → #F79646*

*- Dark Orange: 233,191,149 → #E9BF95*

*- Violet: 234,198,234 → #EAC6EA*

*- Red: 222,103,103 → #DE6767*

*- Light Blue: 240,255,255 → #F0FFFF*

*- Green (another shade): 156,204,84 → #9CCC54*

*- Yellow: 241,224,151 → #F1E097*

# Appendices

## Stereotypes and Tagged Values

|  |  |  |
| --- | --- | --- |
| Stereotpe | Tagged Values | Notes |
| **dApplicationFunction** |  | a logical Function that the application is supporting |
| **dRisk** |  | the potential for an Issue |
|  | Likehood | the probability that this risk becomes an Issue. |
|  | Severity | the expected impact if the risk materializes. |
|  | RiskStrategy | the strategy to be adopted to manage the risk |
| **dAction** |  | A specialized activity. |
| **dActivity** |  | A function or work carried out by the business user. |
| **dActor** |  | The actor entity used for representing a User, System, a Worker or Partner. |
|  | #FTEs | Estimated number of FTEs that operate as this Actor. |
|  | ActorGoal | Objectives that this actor has, in general terms. |
|  | ActorType | the type of Actor |
|  | ActorTasks | Tasks that this actor performs, in general terms. |
| **dApplicationComponent** |  | An encapsulation of application functionality aligned to implementation structure which is modular and replaceable. It encapsulates its behavior and data, provides services, and makes them available through interfaces. |
|  | ID | Unique identifier for the element, useful for referring to third party systems |
|  | Category | User-definable categorization taxonomy for each element. |
|  | BizSatisfaction | How much is the business happy with the application, expressed in 1-100 |
|  | ITSatisfaction | how is IT happy with this application, often related to technical debt. |
|  | BizCriticality | how is Business happy with this application, often related to business debt. |
|  | Cost | the cost of this application in 1000 of $ |
|  | InvestmentStrategy |  |
|  | ApplicationType | the type of Application, e.g. if it's a COTS, Custom and so on |
|  | LastStandardReviewDate | Last review date for the application component. |
|  | NextStandardReviewDate | Next review date for the application component. |
|  | RetireDate | The retire date for the Application Component. |
|  | Owner | The owner for the application Component. |
|  | Port |  |
|  | Source | The original source of the element's information |
| **dBusinessProcess** |  | A business Processes represent a sequence of activities that together achieve a specified outcome, can be decomposed into Business Services. It ensures the correct operation of business capabilities. |
|  | Category | User-definable categorization. |
|  | ID | Unique identifier for the architecture entity. |
|  | LastStandardReviewDate | Last date that the standard was reviewed. |
|  | NextStandardReviewDate | Next date for the standard to be reviewed. |
|  | Owner | Owner of the architecture entity. |
|  | ProcessCriticality | Criticality of this process to business operations. |
|  | ProcessType | Whether this process is supported by IT or is a manual process. |
|  | ProcessVolumetrics | Data on frequency of process execution. |
|  | RetireDate | Date when the standard was/will be retired. |
|  | Source | Location from where the information was collected. |
|  | StandardCreationDate | If the product is a standard, when the standard was created. |
|  | \_metatype |  |
| **dBusinessService** |  | 1. A repeatable activity; a discrete behavior that a building block may be requested or otherwise triggered to perform. 2. . A service provided by Business that achieve a business outcome in response to a request. |
|  | Category | User-definable categorization for Business Service. |
|  | ID | Unique identifier for Business Service. |
|  | LastStandardReviewDate | Last date that the standard was reviewed. |
|  | NextStandardReviewDate | Next date for the standard to be reviewed. |
|  | Owner | Owner of the architecture entity. |
|  | RetireDate | Date when the standard was/will be retired. |
|  | Source | Location from where the information was collected. |
| **dBusinessUseCase** |  | A Business Use Case is part of a business process that produces an advantage to the enterprise. |
|  | GoalInContext | The goal should implicitly express the actor's intent or purpose of the use case, such as "Enrol Student in Seminar." |
|  | Precondition | A list of the conditions, if any, that must be met before a use case may be invoked. Can be a previous Use case or self the presence of the system in Scope. |
|  | Trigger | Event that is responsible for invocation of the use case. |
|  | Scope | Boundaries in which the use case is operated when invoked (E.g. CMS) |
|  | Level | Authorizations for operations/actions to be performed against the Chi business objects in scope. Against every object/process 4 CRUD basic operations are possible:  Create (Write)  Read (Open)  Update (Change)  Delete (Destroy) |
|  | OtherActors | The list of actors associated with the use case. Although this information is contained in the use case itself, it helps to increase the understandability of the use case when the diagram is unavailable. |
|  | MainSuccessScenario | The main path of logic an actor follows through a use case. Often referred to as the "happy path" or the "main path" because it describes how the use case works when everything works as it normally should. |
|  | Extensions |  |
|  | isCore |  |
| **dCapability** |  | A business outcome that is current available in an Organization Unit (or) A business-focused outcome that is delivered by the completion of one or more work packages. |
|  | BusinessValue | assess how this capability provides value to the enterprise. |
|  | Category | User-definable categorization for Capability |
|  | ID | Unique identifier for the Capability |
|  | Owner | Owner of the architecture entity. |
|  | Source | Source from where the information was collected. |
|  | Increments | Current (AS-IS) maturity level for the entity.  0- none  1- Initial  2- Under Development  3 - Defined  4- Managed  5 - Measured |
|  | IncrementsToBe | Future (To-Be) maturity level for the entity. |
|  | IncrementVertical | The maturity level of your vertical (the industry) |
|  | IncrementSupplyChain | The maturity level of your supply chain (Partners) |
|  | Cost | the aggregated cost of this Capability. Includes all the aggregated cost's elements |
|  | Criticality | What is the criticality of this capability, what would be the cost of failure |
|  | Risk | How much Risk connected to this business Capability. 1 low risk, 100 max risk. |
|  | CapabilityLevel | the level of this category in the hierarchy |
| **dController** |  | The Controller defines application behavior and the Business Logic involving more than one ModelClass. It dispatches requests and selects views for presentation. It interprets user inputs and maps them into actions.  Controller, with Views and Associations define the application flow. A controller represents the Business logic where a certain flow is physically implemented.  In a Service Oriented Architecture (SOA) a Controller can be exposed by a Physical Service. |
| **dDataEntity** |  | An encapsulation of data that is recognized by a business domain expert as a thing. Logical data entities can be tied to applications, repositories, and services and may be structured according to implementation considerations. |
|  | PrivacyClassification | Level of restriction placed on access to the data. |
|  | RetentionClassification | Level of retention to be placed on the data. |
|  | Category | User-definable categorization. |
|  | ID | Unique identifier for the Data Entity. |
|  | Owner | Owner of the Data Entity. |
|  | Source | Location from where the information was collected. |
| **dDecision** |  | A decision-making step with accompanying decision logic used to determine execution approach for a process or to ensure that a process complies with governance criteria.  *documents important decisions about any aspect of the initiative including the structure of the system, the provision and allocation of function, the contextual fitness of the system and adherence to standards.* |
|  | Subject Area | Area of Concern |
|  | Topic | Topic of Interest |
|  | Assumptions | What is believed to be true about the context of the problem, constraints on the solution. |
|  | Motivation | Why this decision is important. |
|  | Alternatives | A list of alternatives and explanations |
|  | Decision | The decision taken, possibly with references to related work products |
|  | Justification | Why the decision was made and a list of compliance to Architecture Principles and explanations of deviations from compliance. |
|  | Implications | What impact the decision will have |
|  | RelatedDecisions | A list of related decisions |
| **dDeploymentNode** |  | a deployment node represents a physical or a virtual machine. |
|  | IP | The Ip of this machine |
|  | PublicIP |  |
|  | CPU | the number and type of CPU |
|  | RAM | the Memory dimension |
|  | Owner | The owner / responsible for this node |
|  | Disk | the dimension of the disk |
|  | ArchitectureType | the type of CPU Architecture (e.g. AMD 64) |
| **dFeature** |  | A Feature is a fact that set his related Requirement to be true |
|  | Author | This feature's author's name and role in the project |
|  | Proofreader | This feature's revisor's name and role in the project |
|  | Status | This feature's status |
| **dGoal** |  | A **measurable** scope that the enterprise wants to achieve. Can be hierarchically decomposed. |
|  | ID | Unique identifier for the Goal. |
|  | Value\_amount | The actual amount of the value this goal intends to alter |
|  | Value\_Goal | The amount by which the value is to be altered |
|  | Priority | A priority in % |
|  | Value\_Name | The name of the value this goal intends to alter |
|  | dAssumption | The assumptions made to achieve the goal. |
| **dGrowthPackage** |  | A set of actions identified to achieve one or more objectives for the business. A work package can be a part of an Initiative, a complete project, or a part of a Program. A Growth pack is a collection of Initiatives architected for consistency and aimed to be executed in the same period. |
| **dInitiative** |  | An activity carried out by a Business Unit or organization to achieve a particular goal or improve the maturity of a certain Capability. |
|  | DetailedDescription | Detailed Description of the initiative. |
|  | Impacted Capability | Describes the contribution this work package makes to capability delivery. |
|  | RelatedProgram | The Growth Package or Program associated to the Initiative. |
|  | startDate | The start date for the initiative. |
|  | finishDate | The finish date of the initiative. |
|  | InitiativeDuration | The expected duration of the initiative in Months/ Weeks or Years. |
|  | kind | To capture the type of initiative. |
|  | purpose | Purpose of the initiative. |
|  | Rank | The rank of the initiative. |
|  | Description | Description for the initiative. |
|  | initiativeStatus | a set of colors to Visually describe the health of the initiative. |
| **dIssue** |  | A concrete problem that is affecting a business. Also, a fact that sets a Requirement to be false. |
|  | Author | This issue's author's name and role in the project |
|  | Responsible | The responsible to close the present issue |
| **dLocation** |  | A place where business activity takes place. Can be hierarchically decomposed. |
|  | AreaCode | The area code of the Location. |
|  | City | The City of the Location. |
|  | Country | The City of the Location. |
|  | EmailID | The contact e-mail Id for the Location. |
|  | PhoneNumber | The contact phone number for the Location. |
|  | Province | The Province of the Location. |
|  | Street | The street part of the Location. |
|  | ID | Unique identifier for the Location |
| **dEvent** |  | An organizational state change that triggers a business process; may originate from inside or outside the organization and may be resolved inside or outside the organization. |
|  | Category | User-definable categorization. |
|  | ID | Unique identifier for the Event. |
|  | Owner | Owner of the Event. |
|  | Source | Location from where the information was collected. |
| **dLogicalAppComponent** |  | An encapsulation of application functionality that is independent of a particular implementation. |
| **dMeasurementArea** |  | The organization or business units performance objective is captured as measurement area. |
|  | ID | Unique identifier for the Measurement Area. |
|  | Definition | Definition of the Measurement Area. |
| **dMeasurementCategory** |  | The category to be measured under a measurement area. |
|  | MeasurementArea | The reference Measurement Area to which this category belongs to. |
|  | Definition | Definition of the measurement category. |
|  | ID | Unique identifier for the Measurement Category. |
| **dMeasurementGrouping** |  | Categorization of measurement indicators into various groups under a measurement category. |
|  | MeasurementCategory | Related Measurement Category to which the group belongs to. |
|  | Definition | Definition of the Measurement Grouping. |
|  | ID | Unique identifier for the Measurement Grouping. |
| **dMeasurementIndicator** |  | The specific measure captured for a Organization or Business Unit which can be measured and quantifiable. |
|  | ID | Unique identifier for the Measurement Indicator. |
|  | Definition | Definition of the Measurement Indicator. |
|  | UnitOfMeasure | The unit of measure (if available) for the Measurement Indicator. |
|  | CurrentLevel | the level of the Measurement Indicator the last time that was measured. |
|  | SatisfactionLevel | Satisfaction level for the Measurement Indicator, desidered amount to be reached and/or Max value |
|  | LastStandardReviewDate | last time this KPI was reviewed |
| **dModelClass** |  | Represent a class of the domain model, that can be converted into a Persistent domain class, must inherit from Node. It is the application's dynamic data structure |
|  | child\_order | The order of the associated children e.g. for Recipe: Image|Info|AdminInfo |
|  | display\_value | The display value of a node e.g. for Recipe: name|AdminInfo/status|Info/status |
|  | initparams | The configuration section that is used on initialization of the corresponding mapper. |
|  | is\_searchable | Indicates that this type should be included in the default search. |
|  | is\_soap | Indicates wennever an interface (e.g SOAP or REST) should be generated for this type. |
|  | orderby | Definition of default sorting. Possible values: 'none' (no order), 'sortkey' (generates a 'sortkey' column, that is used for explicit sorting) or any the name of any dValue defined in the node optionally followed by [ASC|DESC] e.g. 'name ASC' |
|  | parent\_order | The order of the associated parents. |
|  | pk\_name | The name of the primary key column. The generator will add this if there is no appropriate attribute. |
|  | table\_name | The name of the database table. If not given the name will be taken. |
| **dObject** |  | A dObject is a candidate for a dModelClass represented in an activity diagram. In alternative can be an instance of a Model Class |
| **dObjective** |  | A time-bounded **milestone** for an organization used to demonstrate progress towards a goal. The Objective is a non-material achievement, other than a Goal, the objective has no specific value to be measured |
|  | Category | User-definable categorization. |
|  | finishDate | the date in which the objective is achieved. |
|  | ID | Unique identifier for the Objective. |
|  | Owner | Owner for the Objective. |
|  | Source | Location from where the information was collected. |
| **dOpinion** |  | A stakeholder view or judgment formed about a Topic. That is formulated in an explicit fashion |
| **dOpinionInner** |  | A Stakeholder’s opinion that is relevant for the topic but is not formulated explicitly. |
| **dOrganizationUnit** |  | A self-contained unit of resources with line management responsibility, goals, objectives, and measures. Organizations may include external parties and business partner organizations. |
|  | HeadCount | Number of FTEs working within the organization. |
|  | ID | Unique identifier for the Organization Unit. |
| **dPhysicalService** |  | the physical realization of a Business Service |
|  | Id | Unique identifier for Service. |
|  | LastStandardReviewDate | Last date that the service was reviewed. |
|  | NextStandardReviewDate | Next date for the service to be reviewed. |
|  | Owner | Owner of the service. |
|  | RetireDate | Date when the service was/will be retired. |
|  | Source | information |
| **dPr****inciple** |  | A qualitative statement of intent that should be met by the architecture. Has at least a supporting rationale and a measure of importance. |
|  | PrincipleID | Unique identifier for the Principle expressed as text. |
|  | Implications | Statement of what the principle means in practical terms. |
|  | PrincipleMeasurement | Identifies mechanisms that will be used to measure whether the principle has been met or not. |
|  | Priority | Priority of this principle relative to other principles. |
|  | Rationale | Statement of why the principle is required and the outcome to be reached. |
|  | Statement | Statement of what the principle is. |
|  | Type | The category of the principle. |
|  | Owner | Owner for the Principle |
|  | Source | Location from where the information was collected. |
| **dProduct** |  | Output generated by the business. The business product of the execution of a process. |
|  | ID | Unique identifier for the Product. |
|  | Owner | Owner of the Product. |
|  | Source | Location from where the information was collected. |
|  | RetireDate |  |
|  | Price |  |
| **dProgram** |  | a set of Initiatives that can be organized in Growth Packages |
| **dRequirement** |  | A Business guide line about the Enterprise or the project. A requirement is formulated in a SMART fashion and uses Moscow verbs. Requirements can be decomposed. |
|  | ID | Unique identifier for the requirement. |
|  | Author | This requirement's author's name and role in the project |
|  | Status | the status of the Requirement in the workflow. |
|  | Priority | A priority in %. Requirements are ordered by priority. |
|  | Proofreader | Each requirement needs to be confirmed.  This requirement's proofreader 's name and role in the project |
|  | Type | The type of requirement. |
| **dResource** |  | a type of Asset supporting a business capability |
|  | # of items |  |
| **dRole** |  | The usual or expected function of an actor, or the part somebody or something plays in a particular action or event. An actor may have a number of roles. |
|  | #FTEs | Number of Full time Equivalent employees working in the Role. |
|  | User | User-definable categorization. |
|  | ID | Unique identifier for the Role . |
|  | Owner | Owner of the Role. |
|  | Source | Location from where the information was collected. |
| **dStakeHolder** |  | Any person or force that have an interest in the considered domain. This is typically indicated by name, can be connected with a Role in the organization. A stakeholder has open and Inner opinions that are relevant for the subject matter. |
|  | Legitimacy | Measure the degree of general acceptance of this stakeholder influence |
|  | Power | the measure of the possibilities in context that this person has to realize his objectives. |
|  | Urgency | the timely importance of the Stakeholder opinions |
| **dSystem** |  | A system configuration. Settings are modeled as attributes with default values |
|  | config | The configuration file where the settings will be placed. |
|  | plattform | The platform to which the configuration settings apply. |
| **dTable** |  | a physical Table in a Database |
|  | Database | The DB schema hosting the table |
| **dUserStory** |  | a requirement expressed from the user point of view that achieve a business value |
|  | TestScenario | The situation that verifies the user story is realized |
|  | EstimatedEffort | Effort expressed in hours |
|  | StoryDetails | intention of the story expressed as "As a **Actor** I want to do **something** So that I achieve **Value"** |
| **dValue** |  | A Node value type used in DAF. These values are persistent |
|  | app\_data\_type | The datatype used in the application e.g. DATATYPE\_DONTCARE, DATATYPE\_ATTRIBUTE, DATATYPE\_ELEMENT or DATATYPE\_IGNORE |
|  | db\_data\_type | The datatype used in the database |
|  | is\_editable | Whether the attribute is editable or not |
|  | input\_type | The HTML input type for the attribute e.g. select#fix:key1[val1]|key2[val2] |
|  | display\_type | The HTML display type for the attribute e.g. image |
|  | restrictions\_match | A regular expression that the value must match (e.g. '[0-3][0-9]\.[0-1][0-9]\.[0-9][0-9][0-9][0-9]' for date values) |
|  | restrictions\_not\_match | A regular expression that the value must NOT match |
|  | restrictions\_description | A text describing the restrictions |
|  | column\_name | The name of the database column. If not given the name will be taken |
| **dValueRef** |  | A Node value type used in DAF that reference another Node Value, similar to a foreign Key. These values are persistent |
| **dValueStream** |  | A representation of an end-to end collection of value-adding Business Processes that create an overall result for a customer, stakeholder or end user |
|  | Criticality |  |
|  | is Decomposed |  |
|  | entrance criteria |  |
| **dView** |  | A View used in DAF. Controller, Views and Associations. It define the application flow from the user point of view. |

## Model Checks

Model checks are implemented using Bellekens EA Validator.

See <https://github.com/GeertBellekens/Enterprise-Architect-Toolpack/releases>

Following checks are implemented:

| Category / Type | Description of the validation rules | Rationale |
| --- | --- | --- |
| All |  |  |
|  | An Element with the status not set | A model element must have the status set |
|  | Elements with the EA status Proposed not managed | Proposed status indicates that the Element is probably not managed |
|  | The Main DAF diagram without objects | No diagram should be empty. |
|  | A Diagram must not contain more than 25 elements | Ideally a diagram should not contain more than seven elements |
|  | A Diagram should not contain less than 2 elements | Ideally a diagram should contain at least two elements |
| CIM |  |  |
| Application Architecture | A “LogicalAppComponent” should exposes a “BusinessService” | a 'dLogicalAppComponent ' element should be connected to a 'dBusinessService' element by a ' exposes Biz Service' Connector |
| Business Architecture | A Goal should be connected to a Business Capability | a 'dCapability' element should be connected to a ' dGoal' element by a 'isOperationalizedBy' Connector |
|  | A Business Capability should be linked to a Measurement Indicator | a 'dCapability' element should be connected to a 'dMeasurementIndicator' element by a 'isMeasuredByf' Connector |
|  | Logical Component is not linked to a Business Capability | a 'dCapability' element should be connected to a 'dLogicalAppComponent' element by a 'Supports Capability' Connector |
|  | Missing link between the Business Process and a Business Capability | a 'dCapability' element should be connected to a 'dBusinessProcess' element by a 'Ensure the correct Operation' Connector |
|  | A Business Capability is not supported by a role | a 'dCapability' element should be connected to a Role element |
|  | Data Entity is not linked to a Business Capability | a 'dCapability' element should be connected to a Data Entity element |
|  | An Initiative should support at least one Business Capability | a 'dCapability' element can be connected to a Initiative element |
|  | A Business Capability is not linked to a Business Process | A Business process ensures the operation of a business capability |
|  | A Business Capability is not supported by at least one Logical Application | A logical application should support at least one business capability |
| Requirement Management | A goal should be linked to a business requirement | a 'dRequirement' element should be connected to a 'dGoal' element by a 'is related to' Connector |
|  | A Business Requirement should be linked to a Feature | a 'dFeature' element should be connected to a ' dRequirement' element by a 'isSatisfiedBy' Connector |
| Technology Architecture | The Network element should be connected to a DeploymentNode | a 'dNetwork' element should be connected to a 'dDeploymentNode' element by a 'NetworkContainsNode' Connector |
|  | A Location element should be connected to a deployment Node | a 'dLocation' element should be connected to a 'dDeploymentNode' element by a 'Hosts Node' Connector |
| PIM |  |  |
| Application Architecture | A Function has no documentation | A function should have some minimal documentation |
|  | An ApplicationComponent documentation contains long dash '–' character | dApplicationComponent documentation should not have a long dash '–' character |
|  | A dApplicationComponent does not have documentation | A dApplicationComponent should have some minimal documentation. |
|  | An dApplicationComponent name should not contain 2 spaces between words in their name | An dApplicationComponent name should not have 2 spaces between words in their name |
|  | An dApplicationComponent name contains leading or trailing space characters in their name | An dApplicationComponent name should not contains leading or trailing space characters in their name |
|  | An dApplicationComponent documentation is in HTML format (rather than plain text) | An dApplicationComponent documentation should be in plain text not in the HTML format |
|  | Application component has no name | Application component requires a name |
|  | Function has no name | A function requires a name |
|  | A function is not associated with an application | a function should be associated with an application |
|  | An ApplicationComponent is not connected to an ApplicationFunction | a 'dApplicationComponent' element should be connected to a 'dApplicationFunction' element by a 'ApplicationAgregatesFunction' Connector |
|  | An application Component doesn't realize a Logical Component | a 'dApplicationComponent' element should be connected to a ' dLogicalAppComponent' element by a 'Realizes' Connector |
|  | A system is not connected to an ApplicationComponent | a 'dApplicationComponent' element should be connected to a 'dSystem' element by a 'configured by' Connector |
|  | An applicationComponent does not exposes a PhysicalService | a 'dApplicationComponent' element should be connected to a 'dPhysicalService' element by a 'Exposes Service' Connector |
|  | A PhysicalService should Implement a BusinessService | a 'dPhysicalService' element should be connected to a 'dBusinessService' element by a 'Implements' Connector |
|  | A Controller should implement a Function | a 'dController' element should be connected to a 'dPhysicalService' element by a 'Implements Service' Connector |
|  | A Controller should Control a ModelClass | a 'dController' element should be connected to a 'dModelClass' element by a 'Controlls' Connector |
|  | A Controller should Implement a BusinessUseCase | a 'dController' element should be connected to a 'dBusinessUseCase' element by a 'isImplementedBy' Connector |
|  | A Controller should govern a View | a 'dController' element should be connected to a 'dView' element by a 'Governs' Connector |
| Business Architecture | A Feature is not realized by Business Use Case | a BusinessUseCase' element should be connected to a 'dFeature' element by a 'is realized by Use Case' Connector |
|  | A BusinessService is not described by a BusinessUseCase | a 'dBusinessUseCase' element should be connected to a 'dBusinessService' element by a 'isDescribedBy' Connector |
|  | Use Cases that are not in exactly one Boundary | A UseCase must appear in exactly 1 Boundary (of all Use Case-diagrams) |
| Technology Architecture | An ApplicationComponent is not physically hosted by a DeploymentNode | a 'dDeploymentNode' element should be connected to a 'dApplicationComponent' element by a 'isPhysicallyHostedBy' Connector |
| PSM |  |  |
| Data Architecture | One or more Class diagrams has more than 25 elements | Update the invalid Class diagrams so they have maximum 25 elements |
|  | There are some unreferenced Enumerations | Search and fix the Enumerations |
|  | ere are some unreferenced Datatypes | All the data type must be described in a class |
|  | Some Enumerations are without values | An Enumeration must have at least 1 value. |
|  | A Model class has no name (or empty name) | Model class has no name (or empty name) |
|  | An Association class is not related to other class(es) | Association should be related or connected to other class(es) |
|  | The Attribute documentation is in HTML format rather than in plain text | The Attribute documentation should be in plain text and not any other format |
|  | An Association documentation should not contain a long dash '–' character | Association documentation should not contain a long dash '–' character |
|  | An Association documentation is in HTML format (rather than plain text) | Association documentation is in HTML format (rather than plain text) |
|  | An Association name should not contain a long dash '–' character | Association name should not contain a long dash '–' character |
|  | A Datatype/Enumeration documentation contain a long dash '–' character | Datatype/Enumeration documentation should not contain a long dash '–' character |
|  | Datatype/Enumeration documentation should not be in the HTML format | Datatype/Enumeration documentation should be in the plain text format |
|  | Datatype/Enumeration does not have any documentation | Datatype/Enumeration should have some documentation |
|  | Some Classes containing Attributes are without datatypes | Every Attribute of a Class must have a Datatype |
|  | Two classes with the same name in the model | Two classes with the same name cannot be in the same model |
|  | A Model Class is without any relations to other entities | Entitie(s) are without any relations to other entities. This validation skips entities which are marked as <<baseType>> |
|  | Model Class does not have Documentation | Model Class does not have Documentation |
|  | Entity is directly derived from more than one base class | Entity is directly derived from more than one base class |
|  | Entity is not directly derived from more than one base class | Entity should be directly derived from more than one base class |
|  | Entities (classes or association classes) do not appear in any diagram in the model | Entities (classes or association classes) should appear at least in one diagram in the model |
|  | An Attribute is not defined as public | Ensure that the attribute need to be != public |
|  | An Attribute has no name (or empty name) | An Attribute must have a name |
|  | Attribute of an entity does not have a type | Attribute of an entity should have a type |
|  | Unnamed association (between 2 model Classes) | all associations must have a name |
|  | Unnamed self-association | A self-association should have a name |
|  | The Association name contains space characters | The Association name should not contain space characters |
|  | Duplicate associations - associations with exactly the same name | associations must have unique names |
|  | Model Classes are without Description | All Model Classes must have a Description. |
|  | Association should specify navigability | unspecified navigability not recommended for information model |
| Implementation | An Initiative should be connected to at least one Business Capability | a 'dInitiative' element should be connected to a 'dCapability' element by a 'IncreasesMaturityOf' Connector |
|  |  |  |

## DAF COLORS