Digital Architectural Framework



DAF

DAF v. 5.5

nov 2024

aaaa

# 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 that aims 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. The DAF implements the OMG's concept of a Platform Independent Model (PIM), which represents a model of a software system that is abstracted from the details of any particular platform or technology, allowing it to be adapted to different platforms or technologies.



Figure 1: Value Streams covered by DAF

The DAF meta-model is based on industry standards such as UML, ArchiMate, TOGAF, RUP, 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  
This work is licensed under a [Creative Commons Attribution 4.0 International License](http://creativecommons.org/licenses/by/4.0/)

You are free to:

* **Share** — copy and redistribute the material in any medium or format
* **Adapt** — remix, transform, and build upon the material for any purpose, even commercially.

# Concepts

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 a formal language, 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 adhere 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."

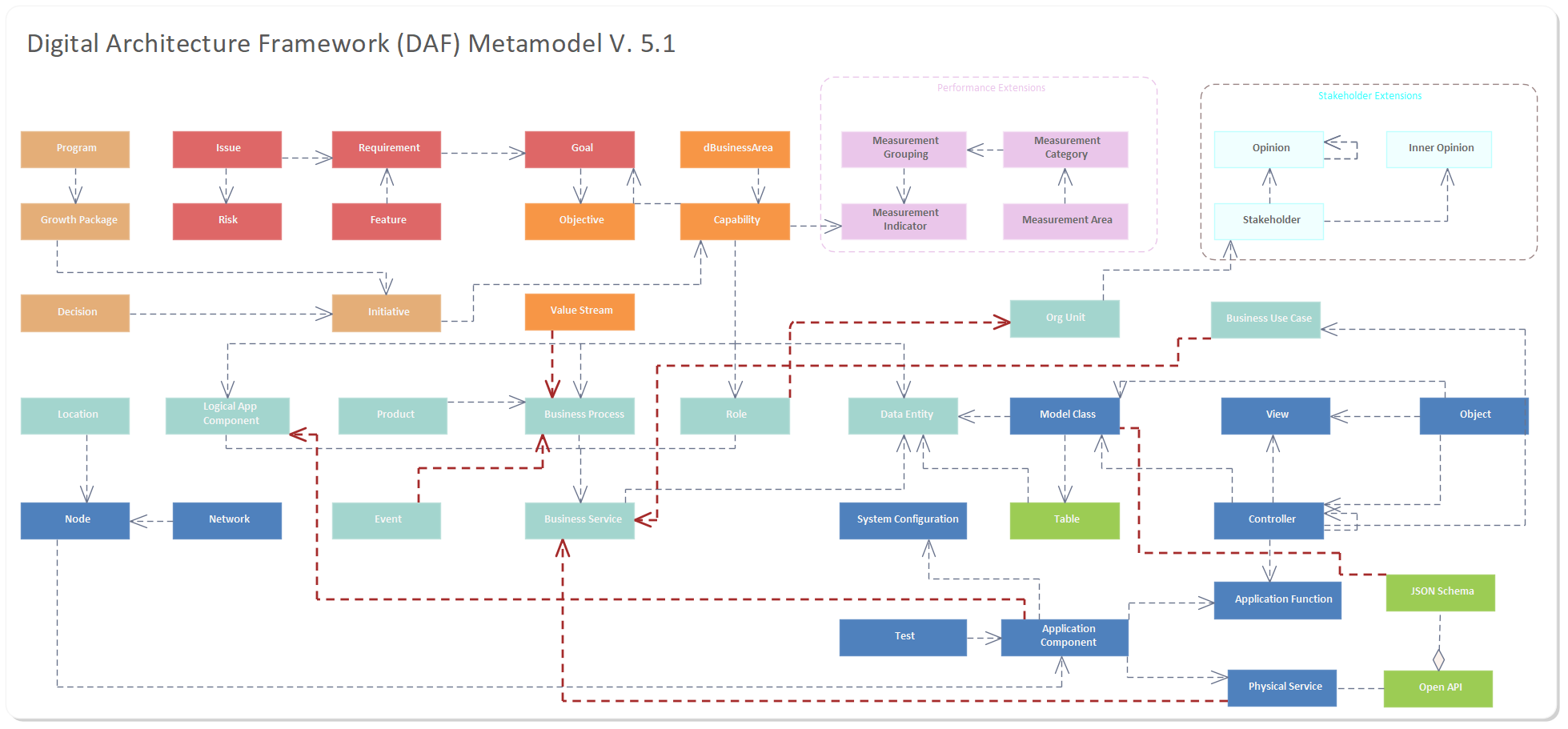
.

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.

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.

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 viewpoints and model elements to be used to capture the relevant information from the system.

## 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 domains below.

## Example Model Instance

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

The following diagram shows an example of a DAF Metamodel instance



Figure : Metamodel Instance

The sparx EA quick linker helps creating only the connections that are valid from the metamodel point of view.

## 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 down into specific domains.

### Enterprise Architecture

* This Layer covers logical considerations.  Viewing the system in terms of roles participating to processes by using resources: Who, How, and What.
* 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 in order 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 support 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, including related concepts at different levels of abstraction. These domains serve as a way 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



* **Business**: This domain covers the business-specific concepts, such as business processes, services, and functions that an organization offers to its customers. This domain is focused on understanding the business objectives and strategies, and how the IT solutions can support those goals.
* **Data**: This domain covers the 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 the 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 the 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.
* Additional Domains can be added (e.g Strategy, Implementation or Security) if required.

## 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 the 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.

****

**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

When is the result of an **Activity**, performed by a Role a **View** can become a **work product**, part of a **Deliverable**.

In the example below, the Role **Enterprise Architect** delivers the **High Level Design** (a deliverable) as an output of the **create** **High Level Design Document** activity.

The document contains mandatory work products such as the Component Diagram and the Capability Business Model. It may include proposed work products such as the platform decomposition diagram or the process diagram.



### DAF concepts Organized in the View Points

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



Figure : DAF concepts Organized in the View Points

## 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)

Figure : Basic UML relationships

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.**

Figure : Additional relationships from ArchiMate

### 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 view point), 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 away 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.

## DAF Concepts Overview



Figure : DAF concepts and Properties

DAF supports ~50 concepts (UML Stereotypes), each of those has several properties (expressed as UML Tagged Values)

# Detailed Documentation

The following is the complete documentation of the DAF metamodel

## Stereotypes

|  |  |  |
| --- | --- | --- |
| Stereotpe | Tagged Values | Notes |
| **dApplicationFunction** |  | a logical Function that the application is supporting |
| **dRisk** |  | the potential for an Issue |
|  | Likehood |  |
|  | Severity |  |
|  | RiskStrategy |  |
| **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. |
|  | \_Image |  |
|  | Icon |  |
|  | \_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- UnderDevelopment  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 Measurment 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 wether 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 wor king within the organization. |
|  | ID | Unique identifier for the Organization Unit. |
| **dPhysicalService** |  | the physical realization of a Business Service |
|  | Id |  |
|  | LastStandardReviewDate |  |
|  | NextStandardReviewDate |  |
|  | Owner |  |
|  | RetireDate |  |
|  | Source |  |
|  | \_Image |  |
| **dPrinciple** |  | 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 plattform 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. |

## Visual Characteristics of Special stereotypes

Some 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

***DAF COLORS***

//blue

SetPenColor**(**107**,**203**,**236**);**

**Enterprise ARCHITECTURE**

//Green

SetPenColor**(**155**,**187**,**89**);**

**SOLUTION ARCHITECTURE**

//dark blue

SetPenColor**(**79**,**129**,**189**);**

**DESIGN ARCHITECTURE**

//Orange

SetPenColor**(**247**,**150**,**70**);**

**COMMON**

//red

SetPenColor**(**222**,**103**,**103**);**

//yellow

SetPenColor**(**241**,**224**,**151**);**

## Relationships Types

| Stereotype | Tagged Values | Notes |
| --- | --- | --- |
| **aggregates categories** |  | Relationship between dMeasurementArea and dMeasurementCategory |
| **aggregates Groups** |  | Relationship between dMeasurementCategory and dMeasurementGrouping |
| **aggregates KPI** |  | Relationship between dMeasurementGrouping and dMeasurementIndicator |
| **Application Owner** |  | Relationship between dApplicationComponent and dStakeholder |
| **areOrganizedWithin** |  |  |
| **configured by** |  | Relationship between dApplicationComponent and dSystem |
| **Consumes Service** |  | Relationship between dRole and dBusinessService |
| **contains** |  | Relationship between dController and dModelClass |
| **Contains Activities** |  | Relationship between dBusinessUseCase and dActivity |
| **Controlls** |  | Relationship between dBusinessUseCase and dActivity |
| **dAccord** |  | Opinion that is in sync with another opinion.  Relationship between dOpinion and dOpinion |
| **dActionKey** |  | Associates two ChiController instances (to define a control flow) or a ChiView with a ChiController (to define a view attachment), must be one directional |
|  | action | The action, which is triggered by this Dependency. If empty, any action is valid |
|  | context | The context, in which this association is valid. If empty, any context is valid. e.g. when 2 controllers are linked twice |
|  | config | The configuration, in which this association is defined |
|  | Method | a.k.a “behaviour”. The name of the method executed by the action, can be empty if the method name is identical to the action name |
|  | decorator | Allows to dynamically attach special behaviors to the result operation without changing his implementation . e.g. a decorator REST, will expose this endpoint as a REST service. |
| **dApplicationAgregatesFunction** |  | dApplicationAgregatesFunction |
| **dAssociation** |  | A type of relation between two architecture entity. |
|  | fk\_name |  |
| **dDiscord** |  | A disagreement on some Opinion. Relationship between dOpinion and dOpinion |
| **dEnsureCorrectOperationOf** |  | A relationship entity used to represent the correct operator relationship.  Relationship between dCapability and dBusinessProcess |
| **dManyToMany** |  | A many to many node used in Chronos. It is used to realize a many to many relation between two Nodes. |
| **Executes** |  | Relationship between dCapability and dRole |
| **exposes Biz Service** |  | Relationship between dLogicalAppComponent and dBusinessService |
| **Exposes Service** |  | Relationship between dApplicationComponent and dPhysicalService |
| **Flows to** |  | Relationship between dBusinessProcess and dBusinessProcess |
| **Governs** |  | Relationship between dController and dView |
| **Has Inner Opinion** |  | Relationship between dStakeholder and dOpinionInner |
| **has Milestone** |  | Relationship between dGoal and dObjective |
| **Has Opinion** |  | Relationship between dStakeholder and dOpinion |
| **has Stakeholders** |  | Relationship between dOrganizationUnit and dStakeholder |
| **Hosts Node** |  | Relationship between dLocation and dDeploymentNode |
| **Impact Resource** |  |  |
| **Implement Biz Service** |  |  |
| **Implements** |  | Relationship between dPhysicalService and dBusinessService |
| **Implements Service** |  | Relationship between dController and dPhysicalService |
| **Includes Process** |  | Relationship between dProduct and dBusinessProcess |
| **Includes Processes** |  | Relationship between dValueStream and dBusinessProcess |
| **IncreasesMaturityOf** |  | Relationship between dInitiative and dCapability |
| **informs Capability** |  | Relationship between dCapability and dDataEntity |
| **is associated to organization** |  | Relationship between dRole and dOrganizationUnit |
| **is delivered by** |  | Relationship between dGrowthPackage and dInitiative |
| **is organized In** |  | Relationship between dProgram and dGrowthPackage |
| **is related to** |  | Relationship between dRequirement and dGoal |
| **is related to risk** |  | Relationship between dRisk and dRisk |
| **isDescribedBy** |  | Relationship between dBusinessUseCase and dBusinessService |
| **isFunctionallyImplementedBy** |  | Relationship between dTable and dDataEntity |
| **isHostedBy** |  |  |
| **isImplementedBy** |  | Relationship between dController and dBusinessUseCase |
| **isMakingUseOf** |  | Relationship between dDecision and dInitiative |
| **isMeasuredBy** |  | Relationship between dCapability and dMeasurementIndicator |
| **isOperationalizedBy** |  | Relationship between dCapability and dGoal |
| **isOrganizedIn** |  |  |
| **isPerformedBy** |  | Relationship between dActor and dRole |
| **isPhysicallyHostedBy** |  | Relationship between dDeploymentNode and dApplicationComponent |
| **Orchestrates service** |  | Relationship between dBusinessProcess and dBusinessService |
| **participatesIn** |  | Relationship between dActor and dBusinessUseCase |
| **ProvidesPersistenceFor** |  | Relationship between dModelClass and dTable |
| **realized by Issue** |  | Relationship between dIssue and dRisk |
| **Realizes** |  | Relationship between dApplicationComponent and dLogicalAppComponent |
| **RealizesEntity** |  | Relationship between dModelClass and dDataEntity |
| **refersTo** |  | Relationship between dDecision and dPrinciple |
| Resource Risk |  | Relationship between dRisk and dResource |
| **StopTheRealizationOf** |  | Relationship between dIssue and dRequirement |
| **Supports** |  | Relationship between dResource and dCapability |
| **Supports Capability** |  | Relationship between dCapability and dLogicalAppComponent |
| **Uses Data** |  | Relationship between dBusinessService and dDataEntity |

### Relationships Quick link

| Source | Relation | Target |
| --- | --- | --- |
| dActivity | contains Action | dAction |
| dActor | isPerformedBy | dRole |
| dActor | participatesIn | dBusinessUseCase |
| dApplicationComponent | Application Owner | dStakeholder |
| dApplicationComponent | configured by | dSystem |
| dApplicationComponent | Connected with Application | dApplicationComponent |
| dApplicationComponent | dApplicationAgregatesFunction | dApplicationFunction |
| dApplicationComponent | Exposes Service | dPhysicalService |
| dApplicationComponent | Realizes | dLogicalAppComponent |
| dBusinessProcess | Flows to | dBusinessProcess |
| dBusinessProcess | Orchestrates service | dBusinessService |
| dBusinessService | dRelatedService | dBusinessService |
| dBusinessService | Uses Data | dDataEntity |
| dBusinessUseCase | Contains Activities | dActivity |
| dBusinessUseCase | is realized by Use Case | dFeature |
| dBusinessUseCase | isDescribedBy | dBusinessService |
| dCapability | Ensure the correct Operation of | dBusinessProcess |
| dCapability | Executes | dRole |
| dCapability | informs Capability | dDataEntity |
| dCapability | isMeasuredBy | dMeasurementIndicator |
| dCapability | isOperationalizedBy | dGoal |
| dCapability | Supports Capability | dLogicalAppComponent |
| dController | Controlls | dModelClass |
| dController | dActionKey | dController |
| dController | Governs | dView |
| dController | Implements Service | dPhysicalService |
| dController | isImplementedBy | dBusinessUseCase |
| dDataEntity | AssociatedEntity | dDataEntity |
| dDecision | isMakingUseOf | dInitiative |
| dDecision | refers To | dPrinciple |
| dDeploymentNode | isPhysicallyHostedBy | dApplicationComponent |
| dEvent |  | dBusinessProcess |
| dFeature | isSatisfiedBy | dRequirement |
| dGoal | has Milestone | dObjective |
| dGrowthPackage | is delivered by | dInitiative |
| dInitiative | IncreasesMaturityOf | dCapability |
| dIssue | realized by Issue | dRisk |
| dIssue | StopTheRealizationOf | dRequirement |
| dLocation | Hosts Node | dDeploymentNode |
| dLogicalAppComponent | exposes Biz Service | dBusinessService |
| dMeasurementArea | aggregates categories | dMeasurementCategory |
| dMeasurementCategory | aggregates Groups | dMeasurementGrouping |
| dMeasurementGrouping | aggregates KPI | dMeasurementIndicator |
| dModelClass | ProvidesPersistenceFor | dTable |
| dModelClass | RealizesEntity | dDataEntity |
| dNetwork | NetworkContainsNode | dDeploymentNode |
| dObject | InstanceofController | dController |
| dObject | InstanceofModel | dModelClass |
| dObject | InstanceofView | dView |
| dOpinion | dAccord | dOpinion |
| dOpinion | dDiscord | dOpinion |
| dOrganizationUnit | has Stakeholders | dStakeholder |
| dPhysicalService | Implements | dBusinessService |
| dProduct | Includes Process | dBusinessProcess |
| dProgram | is organized In | dGrowthPackage |
| dRequirement | Contains Requirement | dRequirement |
| dRequirement | is related to | dGoal |
| dResource | supports | dCapability |
| dRisk | Application Risk | dApplicationComponent |
| dRisk | is related to risk | dRisk |
| dRisk | Resource Risk | dResource |
| dRole | Consumes Service | dBusinessService |
| dRole | is Associated To | dOrganizationUnit |
| dStakeholder | Has Inner Opinion | dOpinionInner |
| dStakeholder | Has Opinion | dOpinion |
| dTable | isFunctionallyImplementedBy | dDataEntity |
| dUserStory |  | dBusinessUseCase |
| dValueStream | Includes Processes | dBusinessProcess |

# Appendices

## Model Checks

Model checks are implemented using Bellekens Validator. Following checks are implemented:

|  |  |  |
| --- | --- | --- |
| Name | Category | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Shape Scripts

### Shape scripts RGB Colors

//blue

**(**107**,**203**,**236**);**

**Enterprise ARCHITECTURE**

//Green

**(**155**,**187**,**89**);**

**SOLUTION ARCHITECTURE**

//dark blue

**(**79**,**129**,**189**);**

**DESIGN ARCHITECTURE**

//Orange

**(**247**,**150**,**70**);**

**COMMON**

//red

**(**222**,**103**,**103**);**

//yellow

**(**241**,**224**,**151**);**

## Shape scripts

Most Stereotypes in DAF have a distrinct shape, that is coded using ShapeScript

### Generic script

shape main

**{**

layouttype**=**"border"**;**

defsize**(**90**,**70**);**

rectangle**(**0**,**0**,**100**,**100**);**

addsubshape**(**"padding"**,**"n"**);**

if**(**hasproperty**(**"rectanglenotation"**,**"0"**))**

**{**

addsubshape**(**"port"**,**"w"**);**

**}**

addsubshape**(**"name"**,**"center"**);**

shape port

**{**

preferredwidth**=**20**;**

scalable**=**false**;**

rectangle**(-**10**,-**10**,**10**,**0**);**

rectangle**(-**10**,**10**,**10**,**20**);**

**}**

shape padding

**{**

preferredheight**=**15**;**

**}**

shape name

**{**

h\_align**=**"center"**;**

if**(**hasproperty**(**"stereotypehidden"**,**"false"**))**

**{**

Println**(**"<<#stereotype#>>"**);**

**}**

print**(**"#name#"**);**

**}**

**}**

**decoration** component

**{**

orientation**=**"ne"**;**

if**(**hasproperty**(**"rectanglenotation"**,**"0"**))**

**{**

**}**

else

**{**

//decoration shape here

**}**

**}**

### Initiatives

startpath**();** // Start to trace out a path

moveto**(**0**,**0**);**

LineTo**(**80**,**0**);**

LineTo**(**100**,**50**);**

LineTo**(**80**,**100**);**

LineTo**(**0**,**100**);**

LineTo**(**20**,**50**);**

LineTo**(**0**,**0**);**

MoveTo**(**20**,**50**);**

LineTo**(**100**,**50**);**

endpath**();** // End tracing out a path

### Worker with circle

ellipse**(**5**,**0**,**95**,**100**);** // background

ellipse**(**35**,**10**,**65**,**35**);** // head

moveto**(**39**,**14**);**

lineto**(**61**,**31**);**

moveto**(**50**,**40**);** // torso

lineto**(**50**,**70**);**

lineto**(**30**,**85**);** //left leg

moveto**(**50**,**70**);**//right leg

lineto**(**70**,**85**);**

moveto**(**35**,**45**);** //arms

lineto**(**65**,**45**);**

### Worker

Ellipse**(**40**,**10**,**60**,**40**);**

SetPenWidth**(**3**);**

Arc**(**39**,**9**,**61**,**41**,**55**,**25**,**39**,**25**);**

MoveTo**(**37**,**25**);**

LineTo**(**63**,**25**);**

SetPenWidth**(**1**);**

MoveTo**(**50**,**40**);**

LineTo**(**50**,**70**);**

LineTo**(**35**,**85**);**

MoveTo**(**50**,**70**);**

LineTo**(**65**,**85**);**

MoveTo**(**35**,**50**);**

LineTo**(**65**,**50**);**

### Internal Worker

Ellipse**(**40**,**10**,**60**,**40**);**

SetPenWidth**(**3**);**

Arc**(**39**,**9**,**61**,**41**,**55**,**25**,**39**,**25**);**

MoveTo**(**37**,**25**);**

LineTo**(**63**,**25**);**

MoveTo**(**50**,**11**);**

LineTo**(**50**,**16**);**

SetPenWidth**(**1**);**

MoveTo**(**50**,**40**);**

LineTo**(**50**,**70**);**

LineTo**(**35**,**85**);**

MoveTo**(**50**,**70**);**

LineTo**(**65**,**85**);**

MoveTo**(**35**,**50**);**

LineTo**(**65**,**50**);**

### External Worker

Ellipse**(**40**,**10**,**60**,**40**);**

SetPenWidth**(**3**);**

Arc**(**39**,**9**,**61**,**41**,**55**,**25**,**39**,**25**);**

MoveTo**(**37**,**25**);**

LineTo**(**63**,**25**);**

MoveTo**(**44**,**13**);**

LineTo**(**50**,**25**);**

LineTo**(**56**,**13**);**

SetPenWidth**(**1**);**

MoveTo**(**50**,**40**);**

LineTo**(**50**,**70**);**

LineTo**(**35**,**85**);**

MoveTo**(**50**,**70**);**

LineTo**(**65**,**85**);**

MoveTo**(**35**,**50**);**

LineTo**(**65**,**50**);**

### Business Partner

Ellipse**(**40**,**10**,**60**,**40**);**

moveto**(**44**,**14**);**

lineto**(**56**,**36**);**

MoveTo**(**50**,**40**);**

LineTo**(**50**,**70**);**

LineTo**(**35**,**85**);**

MoveTo**(**50**,**70**);**

LineTo**(**65**,**85**);**

MoveTo**(**35**,**50**);**

LineTo**(**65**,**50**);**

### Passive business partners

Ellipse**(**40**,**10**,**60**,**40**);**

moveto**(**44**,**14**);**

lineto**(**56**,**36**);**

MoveTo**(**50**,**40**);**

LineTo**(**50**,**70**);**

LineTo**(**35**,**85**);**

MoveTo**(**50**,**70**);**

LineTo**(**65**,**85**);**

MoveTo**(**35**,**60**);**

LineTo**(**50**,**50**);**

Lineto**(**65**,**60**);**

### Active Business Partner

Ellipse**(**40**,**10**,**60**,**40**);**

moveto**(**44**,**14**);**

lineto**(**56**,**36**);**

MoveTo**(**50**,**40**);**

LineTo**(**50**,**70**);**

LineTo**(**35**,**85**);**

MoveTo**(**50**,**70**);**

LineTo**(**65**,**85**);**

MoveTo**(**35**,**40**);**

LineTo**(**50**,**50**);**

Lineto**(**65**,**40**);**