INGENIERIA WEB Y TESTING TEMPRANO

User guide and best practices for NDT-Profile 2.X
Proyect NDT-Suite 2.X

Author: IWT2 Group

Version: 02.03
Date: 07/04/2011

Version Control Sheet

Title	User guide and best practices for NDT-Profile 2.X
Version	02.03
Author	IWT2 Group
Date	07/04/2011

CONTROL DE VERSIONES		
Version	Description / Reason version	Filing Date
01.00	Initial document	12/09/2008
01.01	Added appliances maintenance management system	02/03/2009
01.02	Fixed numbering of tables and figures	07/03/2009
01.03	Incorporated the changes requested by the technicians of the Technical Bureau of Quality and Project Management	17/03/2008
01.04	Technical Reviews Technical Bureau of Quality and Project Management	31/03/2009
02.00	Review the document for its adaptation to NDT-2.X Profile	26/11/2009
02.01	Correcting errors after changes in NDT-2.X Profile	13/04/2010
02.02	Correcting mistakes by Julian Garcia	05/05/2010
02.03	The information from the definition of functional requirements is completed	07/04/2011

Index

List of Figures

List of Tables

1. Objectives of the paper

This document comes as a result of the work of 10 years of the band's Web Artifactering Research and Early Testing at the University of Seville. One of the main products obtained in these years is the development methodology NDT oriented web environments and hypermedia. On the web http://www.iwt2.org/ can find a compilation of the most important works produced in these 10 years, as well as support NDT tools mentioned in the following sections.

2. Introduction to NDT-Suite

Navigational Development Techniques (NDT) is a methodology for developing Web and hypermedia systems. NDT-Suite is a set of tools to implement the methodology NDT.

NDT-Suite supports all phases of requirements analysis, design, construction and implantation, testing and maintenance, all based on the methodology Metrics V3. A merger carried out is based on defining a process similar to V3 Metrics but using UML models and extensions of them perform NDT as well as their artifactering processes guided by models.

NDT-Suite consists of three tools, which are: NDT-Profile, NDT-Quality and NDT-Driver.

2.1. NDT-Profile

NDT-Profile is a tool on a defined profile, based on the NDT metamodels, on Enterprise Architect. After a comparative study of 10 tools, this was the one that offered greater support and offered better benefits. The profile on Enterprise offers a range of tools and artifacts definition for working with NDT methodology allowing for easy document management. This profile is distributed as an empty project in Enterprise Architect.

The use of NDT-Profile offers the possibility to have all the NDT artifacts in a simple way, since they are integrated into the tool itself, but in addition, also supports all UML models and integrate them easily in the methodology NDT.

As a new, NDT-Profile 2.X provides a set of templates that automatically generate a document for some of the life cycle phases. Specifically: the phases of requirements, the phase of analysis, the phase of design and the phase testing of the system. For more information view "16. Generate documentation".

2.2. NDT-Quality

NDT-Quality automates some of the methodological review of a project developed with NDT-Profile, creating a report with a description of the inconsistencies that were found during the review, further verifying that the transition from requirements analysis and design analysis done correctly.

NDT-Quality generates a report consisting of a series of mistakes. These errors are classified as "Mild" and "Graves" and review specific aspects of the NDT methods such as fault complete definitions, error in definitions of constraints, etc. Errors also are shown grouped according to the phase where they are committed: Viability Study, Requirements, Analysis, Design and Testing. In addition, it also validates the traceability between phases and controls the consistency between each phase. It also checks that the resulting prototype is navigable.

2.3. NDT-Driver

NDT-Driver allows automatically, taking as input a file developed by NDT-Profile 2.X, execute the transformations defined in NDT methods. For this it is essential that the NDT-Quality tool does not report any grave bugs.

This tool only generates the basic models of the phase in question, providing a starting point to you for you complete these basic models.

2.4. Using this manual

This manual describes in detail the use of NDT-Profile, and the best practices that are mandatory. The support tools may be used only if we follow the good practices identified in this manual.

Table 1. Manual usage rules

Best practices

Good practice in this manual are mandatory in all projects developed on NDT-Profile.
There shall stand any deliverable that has changed or omitted the structure of information and best practices defined in this document

3. Description of NDT-Profile

3.1. Packages and folder structure

NDT-Profile is distributed as an empty project for the Enterprise Architect tool, which functions as a template for creating new projects. The developmental phases included in the project are the phases defined in Metric. These phases will be identified by the following abbreviations.

- EVS: System viability study document.
- DRS: System requirements document.
- DAS: System analysis document.
- DDS: System design document.
- DPS: System test document.
- DMS: System maintenance document.

Additionally, it introduces a series of folders with information about the project:

- PARTICIPANTS: outline the companies and individuals who participate in the project.
- VERSION CONTROL: describing the different baselines (baselines).
- OBJECTIVES OF THE PROJECT: describing the goals to be achieved in the project.

Once the template is opened, the computer is the main screen shown in Figure 1.

NOTE: The distribution shown in Figure 1 could be modified if the user of the tool Enterprise Architect would have altered according to your preferences.

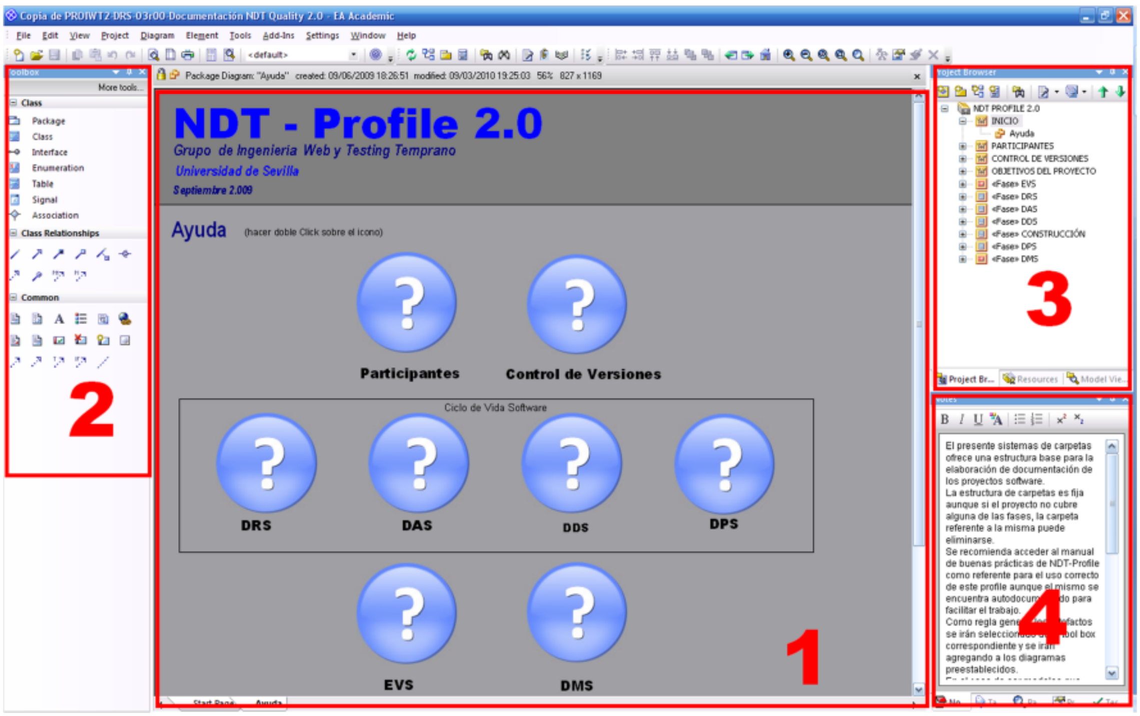

Figure 1. Main screen of Enterprise Architect

In the center (marked with number 1 in Figure 1) is the working area on which the plot develops open. This is where you see all the elements that are drawn on the diagram that is currently active. Just below is a set of tabs that correspond to the diagrams that are open.

A work area must be adding the artifacts, links or other items deemed necessary from the toolbox (marked with number 2 in Figure 1). Each diagram type has an associated toolbox, which will activate when the diagram is displayed. If the panel does not display the toolbox, it is displayed by clicking on the menu **View > Toolbox,** or by pressing Alt +5. You can also select the set of tools with which you want to work in the tool box (as shown in Figure 2). For example, to select the tools of requirements in the toolbox, select the "More tools ...", is positioned on "NDT System Requirements" and select the corresponding toolbar.

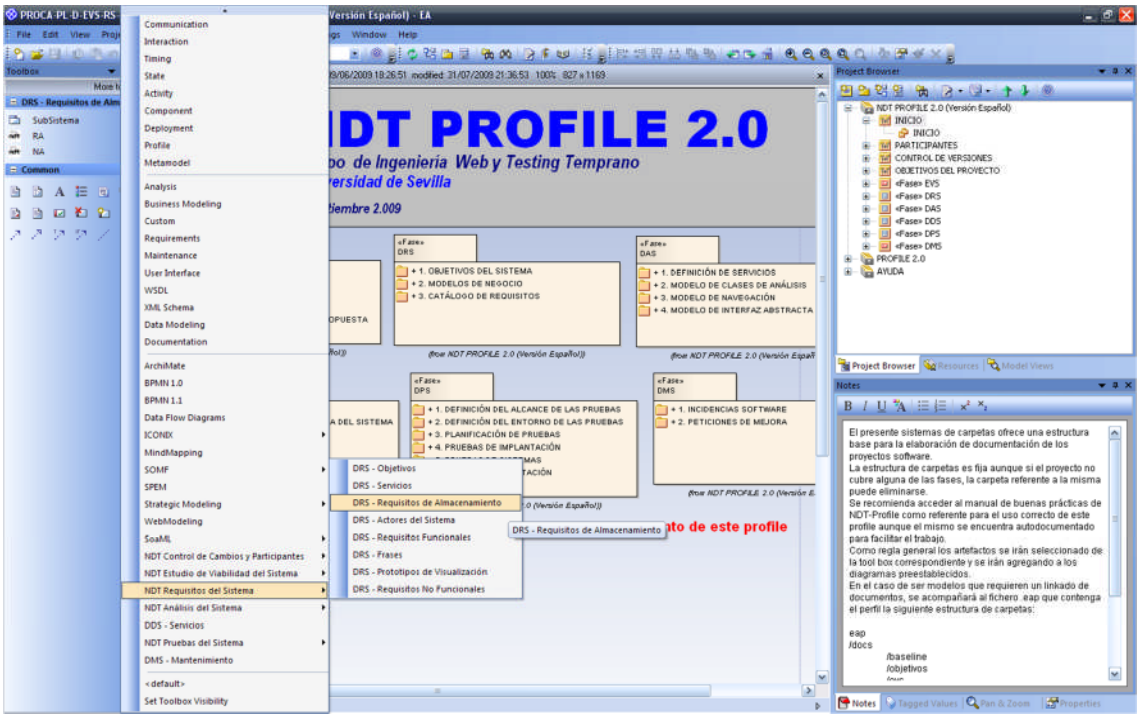

Figure 2. Choosing items in the toolbox

In the profile of NDT has seven sets of tools, one for each of the phases of the life cycle which covers:

- NDT Control Changes and participants, to cover those sections
- NDT Viability study to cover the Viability study phase
- NDT Requirements to cover the requirements artifactering phase
- NDT Analysis to cover the phase of analysis
- NDT Design to cover the design phase
- NDT Tests to cover the testing phase
- NDT Maintenance to cover the maintenance phase

At each time, the development team will select the set of tools for the phase at which they are working.

Above right is the Project Browser (marked with the number 3 in Figure 1), which allows you to navigate the project. Here are packages, diagrams, elements and properties of elements. From here you can drag and drop items between folders, or drag to the active diagram. In addition, right clicking on the elements can perform different actions on them. If the panel does not display the Project Browser, it displays the menu by clicking on **View> Project Browser**, or by pressing Alt +0.

Bottom right panels are different (marked with the number 4 in Figure 1) of various utilities. These are Notes pane, which displays the item description that is currently active or Tagged Values panel, which shows the tagged values of the artifacts. This last is necessary since there are mandatory tagged values, and although they are optional, the tagged values give us important information about the artifacts. To display the Tagged Values panel to be activated in the menu **View> Tagged Values**, or press Ctrl + Shift +6.

Within the project browser is a series of options (Figure 3) to create document, diagram, classes, folders, search and advance and delay in the browser:

Figure 3. Toolbox Project Browser

With the first button, create a folder (which hangs directly from the project). In NDT-Profile this button is not necessary to use it. The second button lets you create a package. The third button opens the dialog box to create a new diagram. The fourth button opens the dialog box to create a new document.

Table 2. Rules and structures packages Packages

Best practices

The folder structure of NDT-Profile can not be modified. You can only create new folders within existing folders.

It can import elements from one folder to another by dragging with the mouse or the "move" of Enterprise Architect.

The folder PROFILE, should never be changed

3.2. Traceability Matrixs

Maintain traceability between the artifacts of a project is one of the most important tasks in software projects. NDT-Profile manages the traceability of artifacts through the use of traceability matrixs. The traceability matrix is a visualization tool that allows you to see comfortably and globaly, all relationships between two groups of artifacts (eg between requirements and use cases or between use cases and classes). This makes it the perfect tool to add a large number of relationships quickly and therefore the perfect tool to define traceability between two sets of artifacts.

You can to use the Resources panel (Figure 4) to access the list of traceability matrixs defined in NDT-Profile. If the panel is not displayed, it is displayed by clicking on the menu **View > Resources**, or by pressing Alt +6.

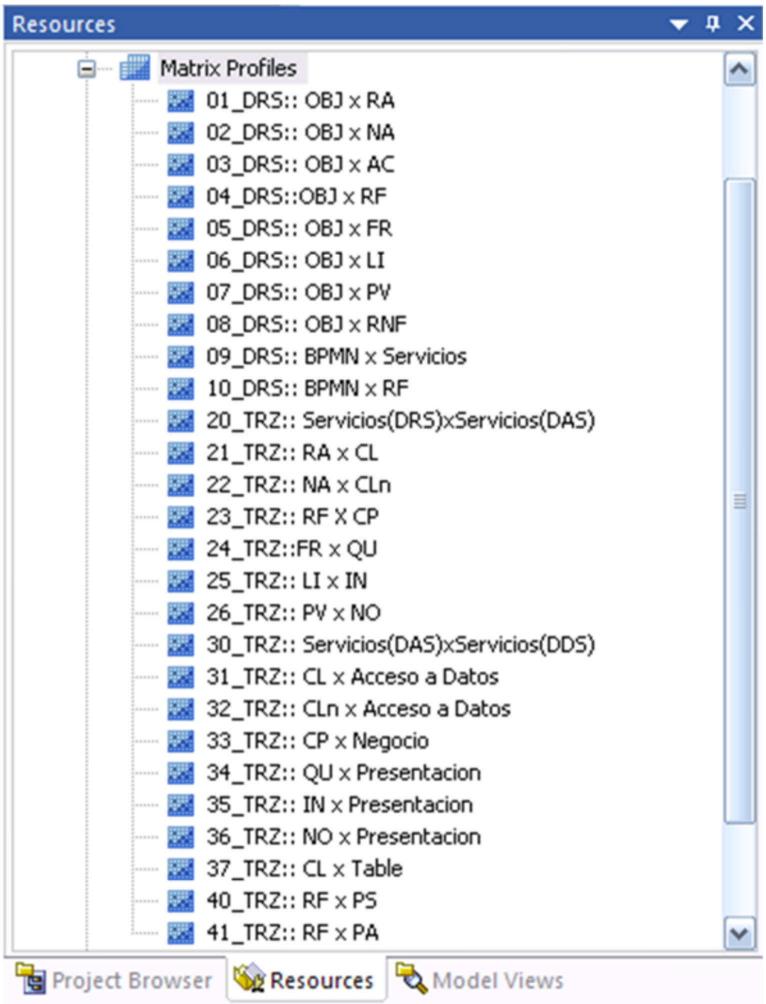

Figure 4. List of traceability matrixs

They are ordered by the phase to be completed and to access them simply double clicking it.

Table 3. Traceability matrixs Rules

Best practices

Traceability matrixs defined in the proyect can not be changed and should be completed those belonging to the phase where the project is

As in the case of the folder structure, traceability matrixs defined in the draft can not be changed in their structure, and all must be completed.

In Section 5.2 expands on the information on how to use traceability matrixs included in NDT-profile.

3.3. Baselines

At any time during the project, changes are required to compare the different versions that are emerging in the NDT-Profile. To solve the problem using the reference or baseline state as it is called in Enterprise Architect. The baselines are images of a point in the process. You can create as many baselines as needed (see Figure 5). It is used in different versions of each profile and each phase of NDT. That is, you can create baselines generally (Profile) or locally (EVS, DRS, DAS, DDS, DPS).

An example of creating a baselined for the entire project, once created artifacts. You select the profile or the phase in the Project Browser, and with the right mouse button you select **Package Control> Manager Baselines.**

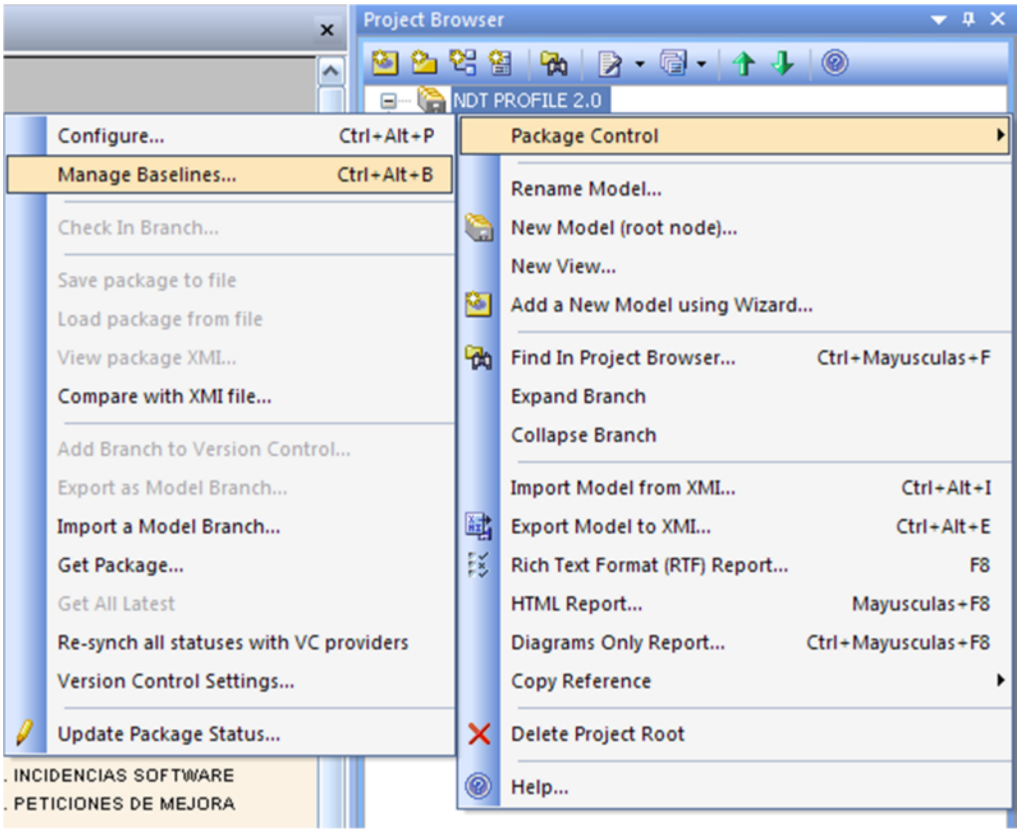

Figure 5. Accessing the management baselines

If needed, add a new *baseline* version by clicking on the button **New Baseline.**

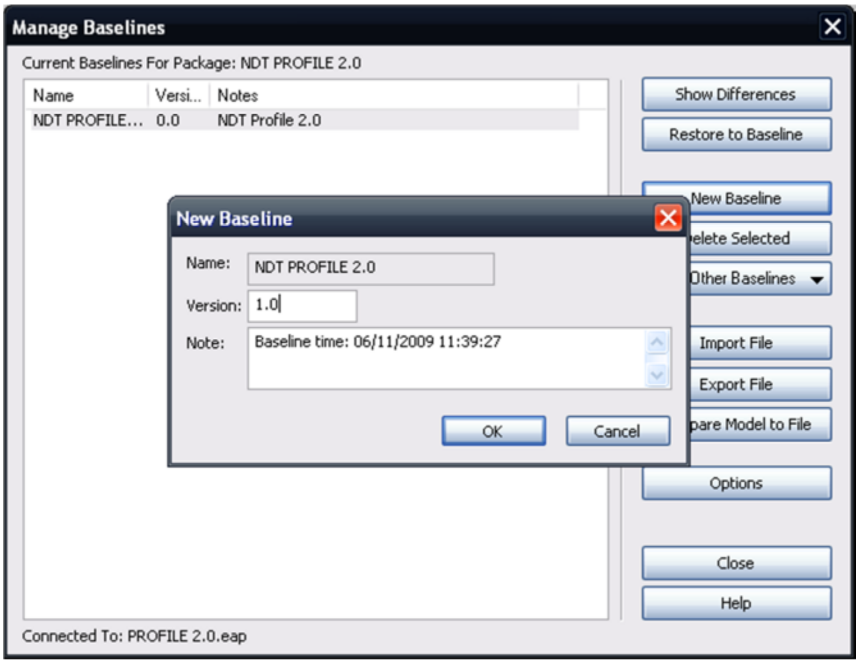

Figure 6. Baselines versioning

Add a version and the description "Note" is optional and always under the NDT standard. Once the baseline is established and can continue working. At any time you can see the status prior to the changes. You can see the changes as shown in Figure 7:

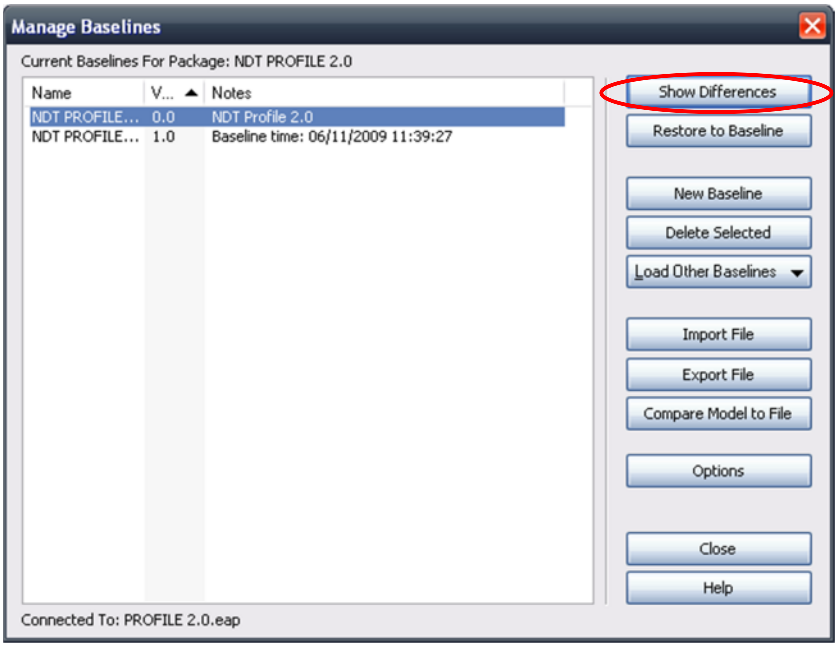

Figure 7. Comparing with the previous state

Select the baseline and press **Show Differences** button. From now displays a tree of the entire profile or the phase that has undergone a baseline as shown in Figure 8:

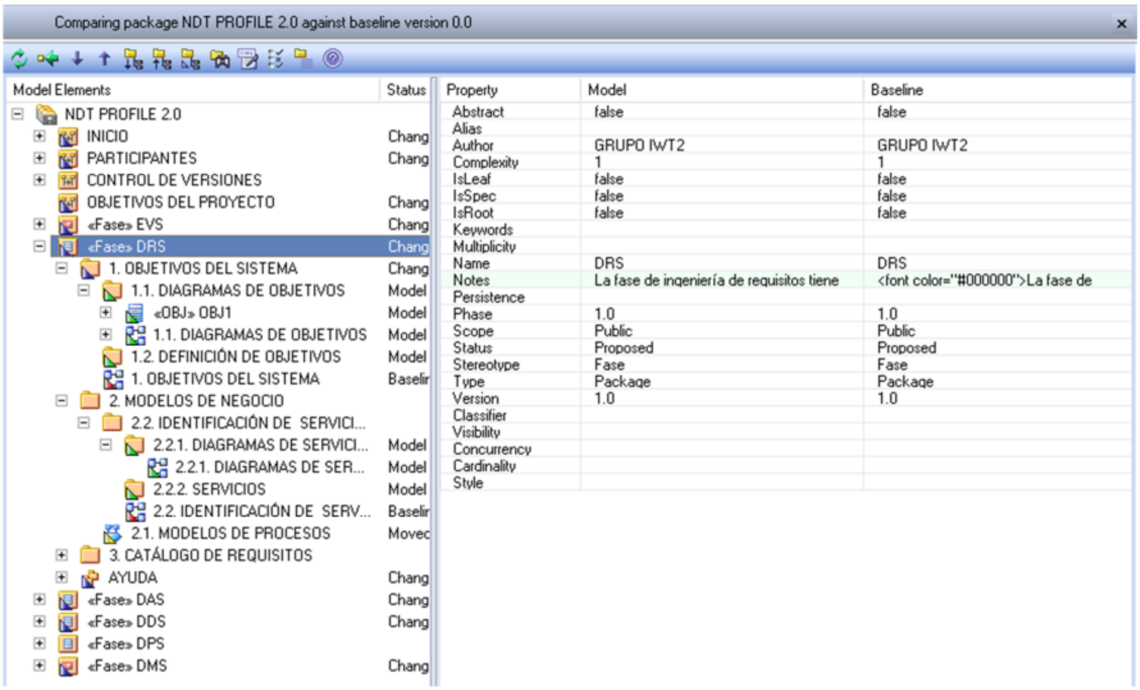

Figure 8. Full status display of each of the artifacts

And depending on the changes made, some triangles are displayed on the icons of the artifacts with different colors: green indicates that the object is new compared to the baseline or previous version, red indicates an object has been deleted, blue is a modified object and finally the yellow indicates a moved object folders. You can also view the status of the properties and attributes of each appliance. On the right side of the window of comparison, the properties of the elements that have been modified are shaded in blue.

Above the window of comparison is a series of buttons with different uses. The button bar is shown in Figuea 9:

Figuea 9. Buttons comparison baselines

For example, the second button, having selected an item, it will revert to its current state as it was used as the baseline for comparison. The tenth button displays a menu of options to configure the items displayed in the window of comparison in order to restrict what types of items are displayed.

You can create as many as deemed appropriate baselines necessary, and always according to the versions of the deliverables that are made.

Note: The baselines occupy space within the file *.eap, since they are stored inside. To avoid excessive increase in size eap and corrupt is recommended to be exported to xml and disposed of eap. To export you should select the appropriate baseline, click the **Export File** button and select the location to save. When you need to see the baseline, just import the xml file which you want to compare the current model. To do so click on the **Import File** button and select the xml file on your hard drive. The location of these buttons is shown in Figura 10.

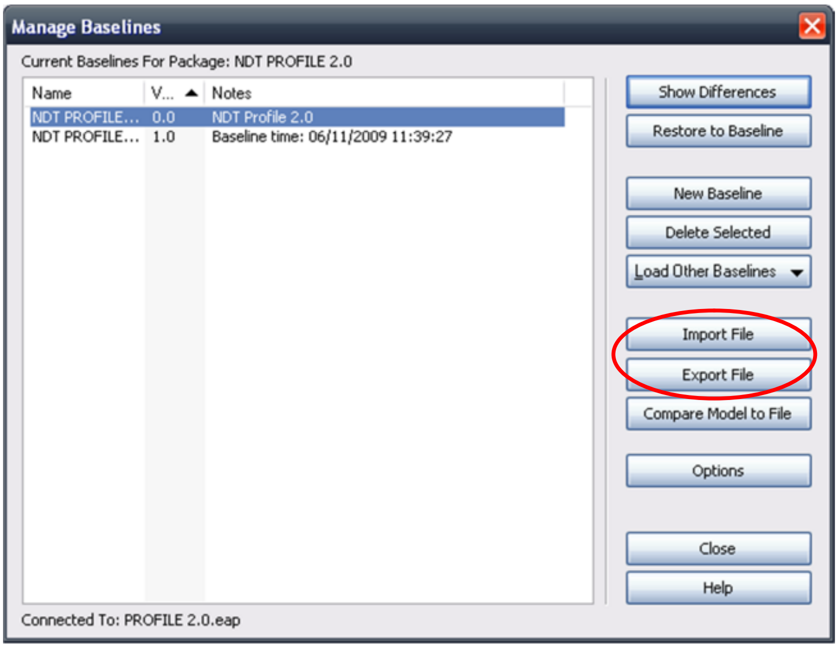

Figura 10. Import and export baselines

Table 4. Rules of the baselines

Best practices

With each deliverable/document is to be delivered in a baseline xml encompassing the entire profile indicating the version and date.

4. UML Modeling with Enterprise Architect. General rules

The following are proposed general rules of the different elements of UML that we find in NDT-Profile.

4.1. Diagrams

No limit on the number of diagrams or new packages can be created within a package. To create diagrams in any of the sections of the project works as follows: select the package where you want to create the diagram. Then select the icon (Figure 2) of the project toolbox and select the type of diagrams corresponding to the phase where you are. Can also be added by selecting the folder in which you want to add the diagram and clicking the right mouse button positioned on **Add** and select **Add diagram ...** The result is shown in Figure 11.

There are several groups of diagrams defined, one for each phase of NDT-Profile. Within each there are a set of diagrams, as shown in Figure 11. There are other groups predefined by the tool diagrams, including UML diagrams are.

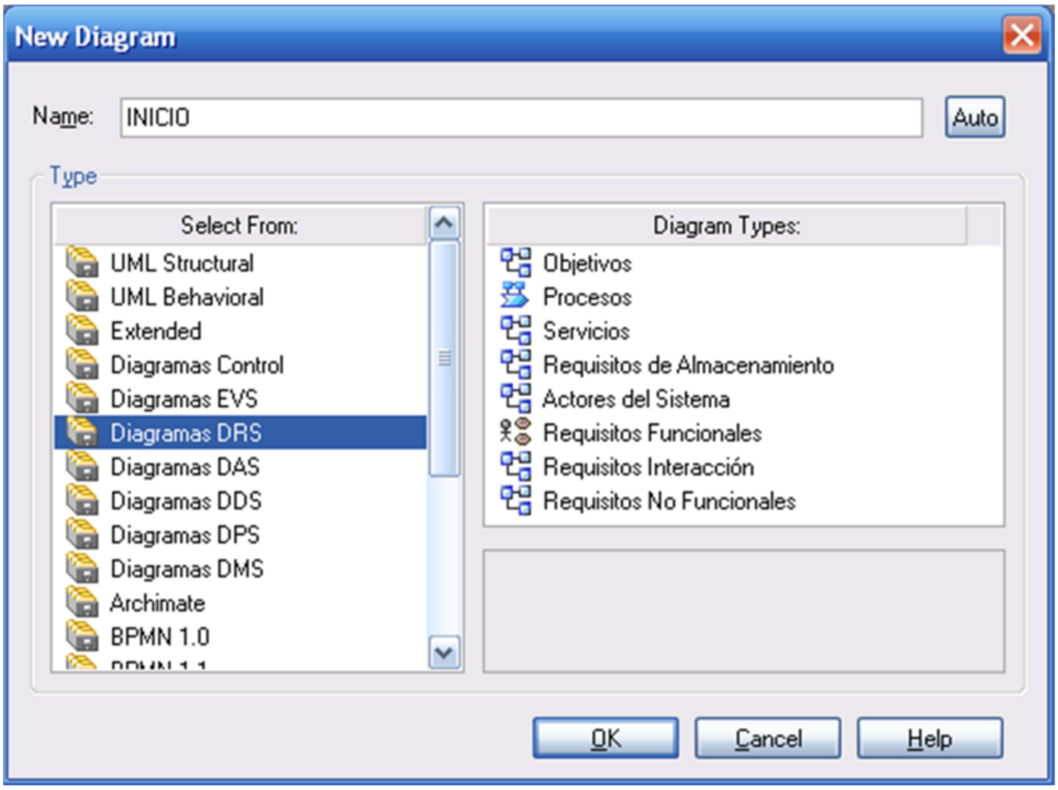

Figure 11. Choosing the Chart Type

The correlation between the sections of the project and the different types of diagram of the Enterprise Architect tool are shown in Table 5. The sections are not included in the table does not have any associated diagram, can be used as a type of diagram free choice or use a text document Annex.

Table 5. Types of diagrams for sections of the draft

Paragraph	Chart Type
PARTICIPANTS	Control Diagrams - NDT Participants
VERSION CONTROL	Control Diagrams - NDT Version Control
EVS - Storage Requirements	Diagrams EVS - Storage Requirements
EVS - Actors	Diagrams EVS - System Actors
EVS - Functional Requirements	Diagrams EVS - Functional Requirements
EVS - Interaction Requirements	Diagrams EVS - Interaction Requirements
EVS - Nonfunctional requirements	Diagrams EVS - Nonfunctional requirements
DRS - Objectives	Diagrams DRS - Objectives
DRS - Process Models	Diagrams DRS - Processes
DRS - Identification Services	Diagrams DRS - Services
DRS - Storage Requirements	Diagrams DRS - Storage Requirements
DRS - Actors	Diagrams DRS - Actors
DRS - Functional Requirements	Diagrams DRS - Functional Requirements
DRS - Interaction Requirements	Diagrams DRS - Interaction Requirements
DRS - Nonfunctional requirements	Diagrams DRS - Nonfunctional requirements
DAS - Definition of Services	Diagrams DAS - Services
DAS - Content Types	Diagrams DAS - Content Types
DAS - Process Classes	Diagrams DAS - Process Classes
DAS - Actors Studio	Diagrams DAS - Sequence Diagram
DAS - Navigation Classes	Diagrams DAS - Actors Studio
DDS - Design Services	Diagrams DAS - Navigational Model
DDS - Design Services (1)	Diagrams DDS - WSDL
DDS - Design Services (2)	SOAML - SoaML Component Diagram
DDS - Design Services (3)	SOAML - SoaML Sequence Diagram
DDS - Technological Environment	Diagrams DDS - Technological Environment
DDS - Design Classes	Diagrams DDS - Design Classes
DDS - Physical Data Model	Diagrams DDS - Data Modeling
DPS - Implementation Testing	Diagrams DPS - Implementation Testing
DPS - System Testing	Diagrams DPS - System Testing
DPS - Acceptance Tests	Diagrams DPS - Acceptance Tests
DMS - Software Issues	Diagrams DMS - Maintenance
DMS - Improvement Petitions	Diagrams DMS - Maintenance

(1) Each service must have an associated diagram WSDL
(2) Each service must have an associated SoaML Component diagram.
(3) Each service can have an associated SoaML Sequence diagram.

These diagrams are already created in their appropriate folder, so just go adding artifacts to the diagram. When you open the diagram, the toolkit will be charged for the phase you are working, making it easy to add appliances as they will be available only those that can be added to the diagram, and possible connections. For example, if you want to complete that section of storage requirements should open the diagram in Section 3.1.1 of the DRS folder. Once opened, to add a new storage requirement is selected in the toolbox the RA artifact and drag it to the center panel. When you add an item, this is incorporated into the folder where you will find the diagram in the above example (Figure 11) in the folder 3.1.1. This artifact must then move to the folder 3.1.2, which is where the storage requirements defined.

4.1.1. Use Case Diagrams

When describing both the functionality of the system, as all tests are used use case diagrams. These diagrams have to meet certain basic rules modeling so they can consider valid, something not always seen when you do this type of diagram. In Figure 12 we can see an well example.

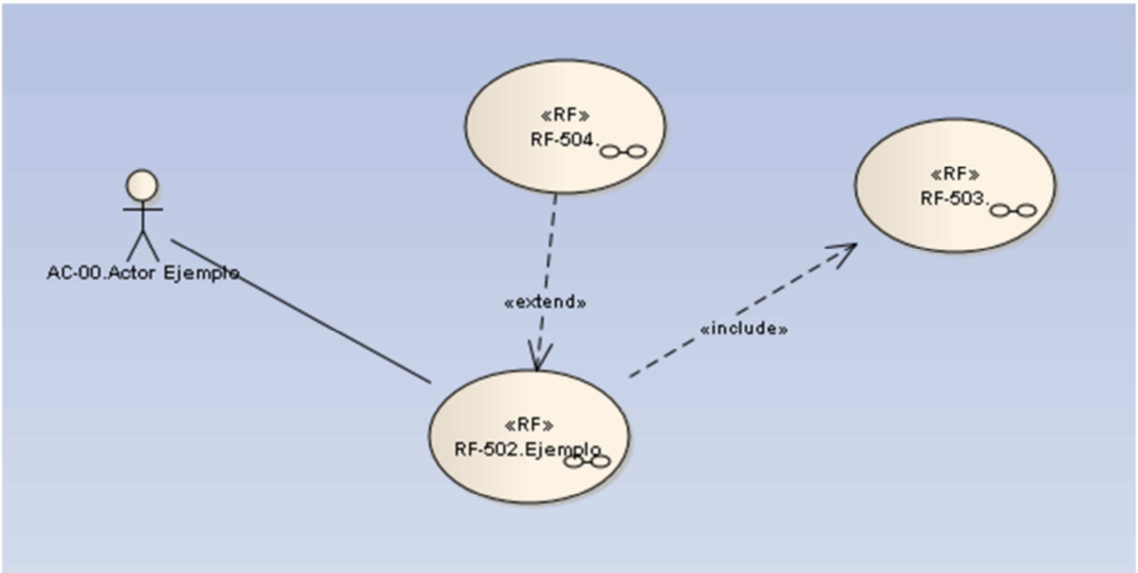

Figure 12. Example use case diagram well built

First, in a use case diagram must appear at least one actor, since it has to specify who can perform the functionality. In addition, all use cases should have a link to the actor, or have a relationship include or extend (both are defined by UML) with a use case linked to an actor. The link to the actor has to be of use and their properties can be edited as discussed later in section 4.3.

It is also a fundamental requirement in the diagram (or all charts) is to appear all use cases have been defined.

4.1.2. Activity diagrams

You can use activity diagrams in the definition of functional and performance requirements on the behavior of test cases.To help build some activity diagrams semantically correct, and then identifies a series of practices must be met when developing these types of diagrams.

Table 6. Activity Diagram Rules

Good practice for activity diagrams
Each activity diagram must have a unique starting point. In addition, there must be one and only one diagram flow between the start and the first activity.
Each activity diagram must have at least one end node. This node should have no outflow.
All activity will take one and only one outflow. This flow can not have any saves.
All outflows from a decision should have a different guard. That is, a decision may not have more than one output stream with the same care.
All activities of an activity diagram must be defined in their own activity diagram.
There can be no more than an activity diagram with the same name.
All activities and decisions involved in the activity diagram to display in the activity diagram. In addition, all associations involving members of the activity diagram to display in the activity diagram.
There can be more of a flow between the same two elements of an activity diagram
There can be no more than one activity with the same name in the same activity diagram.

4.2. Artifacts

Double clicking on any artifact are standard properties, as shown in Figure 13. The standard properties are those properties already identified by Enterprise Architect for each appliance. When adding a new artifact, you must fill out the properties defined as mandatory in this manual. Also highly recommended is not compulsory to fill in the properties. Figure 13, shows all possible standard properties, which are available for all artifacts that inherit from class (OBJ, Service, RA, NA, CL, CLn, NO, QU, IN, ME, PR, NE, AD, PM and IS). The artifacts that inherit from another type of element (for example, AC or RF) will not have any of the tabs, but they have, match those defined below.

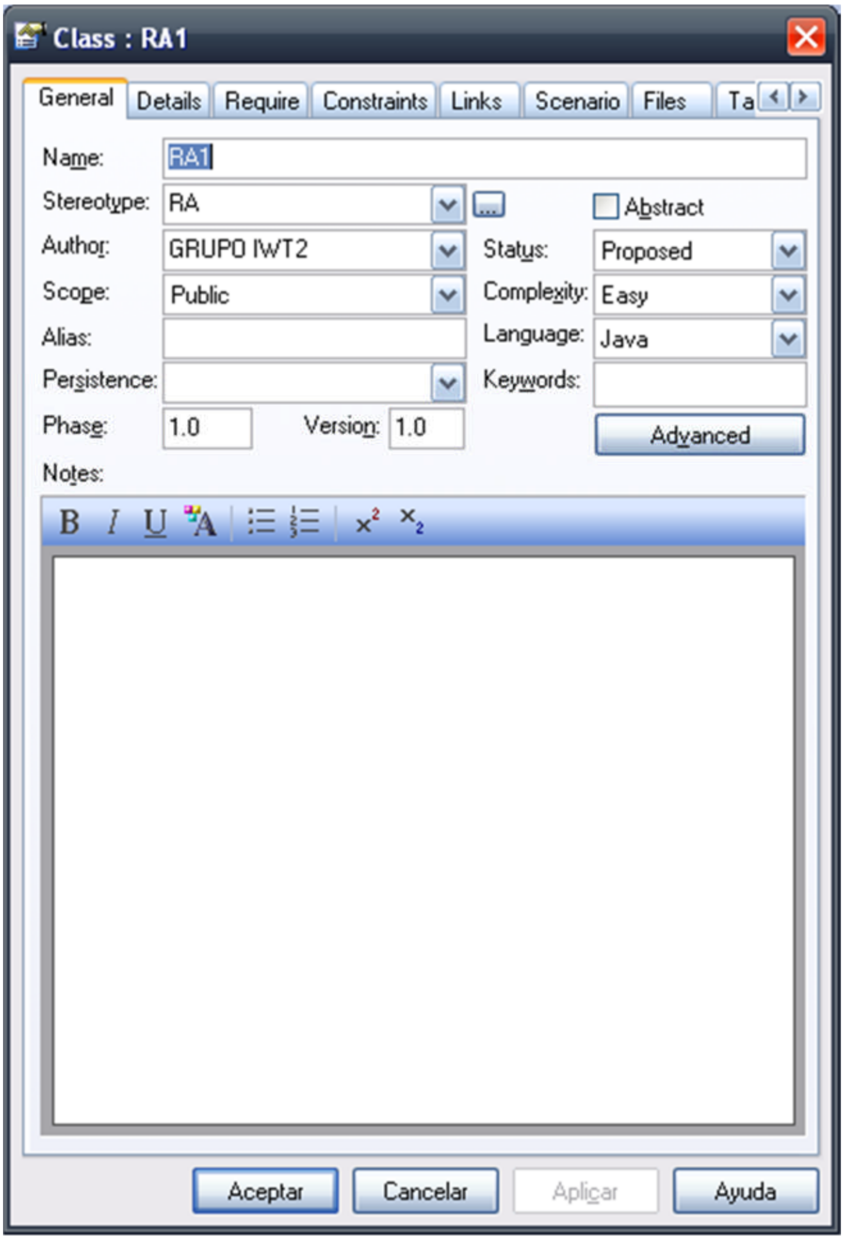

Figure 13. Properties Window (General tab)

The Name field must contain the name of the element to create, according to the specific nomenclature, which will be detailed in the tables of nomenclature of the different phases. Stereotype field is automatically filled with the stereotype of the element we created. Author field must contain the name of the company or person who has shaped the item. In the Notes field should develop a description of the item. In the field Language should indicate the language of the artifact, being in EVS and DRS language NDT-Requisitos, DAS and DDS the chosen programming language (currently, NDT-Driver works only with C# and Java) and the model data the database you are using (Oracle, MySQL and SQLServer). These properties are (unless otherwise noted) a subset of the required fields to fill in the modeling elements.

The Alias field is used to reference other artifact from which the element has been built that want to model (generally used in automatic transformations performed by NDT-Driver). This field is required for some types of artifacts (NO, QU, IN, ME, PR, NE and AD). The Status field can be used to indicate the status of the item. The Version field can be used to indicate the version of the item. These fields are optional, unless otherwise noted, the NDT modeling elements.

The other fields have no use in NDT, although its use is not restricted.

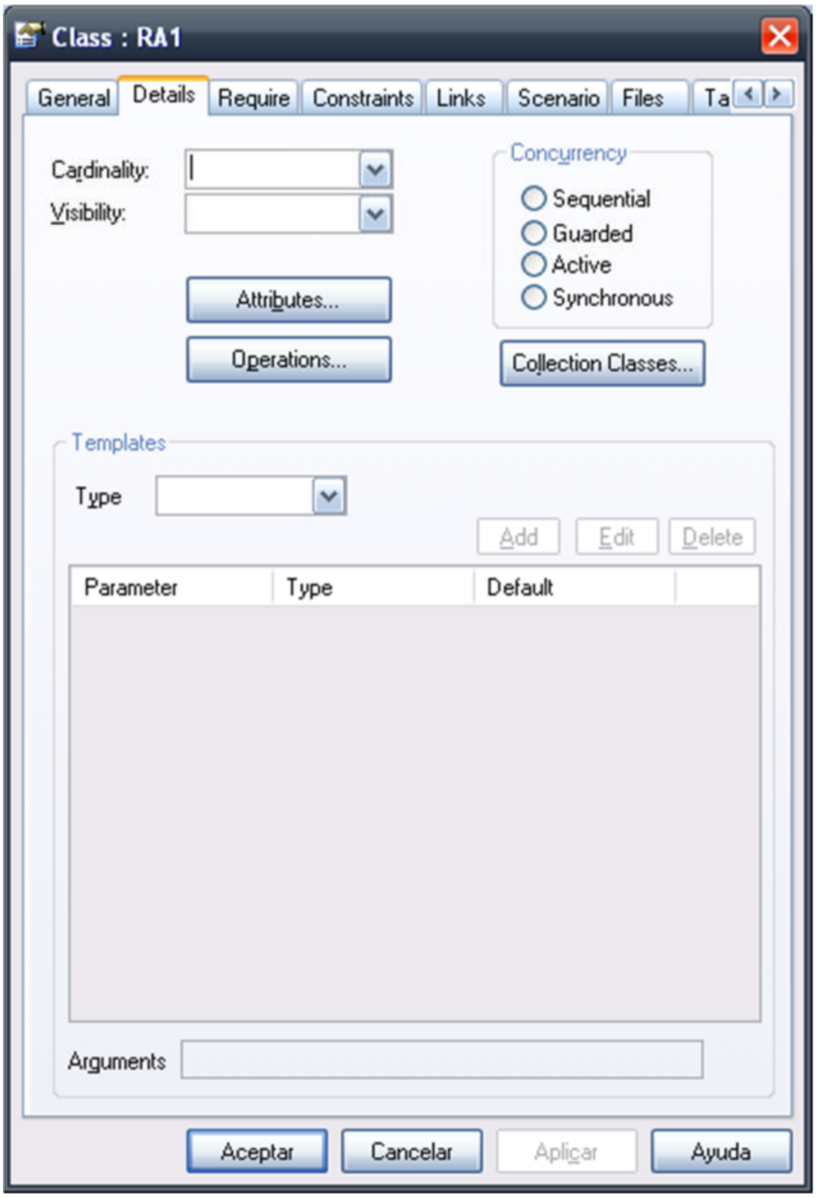

Figure 14. Choosing details (Details tab)

Through the Details tab (Figure 14) can access the attributes and operations of the artifacts through the buttons Attributes and Operations, respectively. The attributes are required in various artifacts (RA, NA, Services, CL, CLn, NO, QU, PR and AD) and operations are required in the NO and NE. Furthermore, through the Collection Classes button specifies whether the artifact is a collection. The other fields and functions in the Details tab have no specific use in NDT and therefore not relevant in this guide.

In Figure 15 we can see the screen through which you can add attributes to an artifact, the General tab.

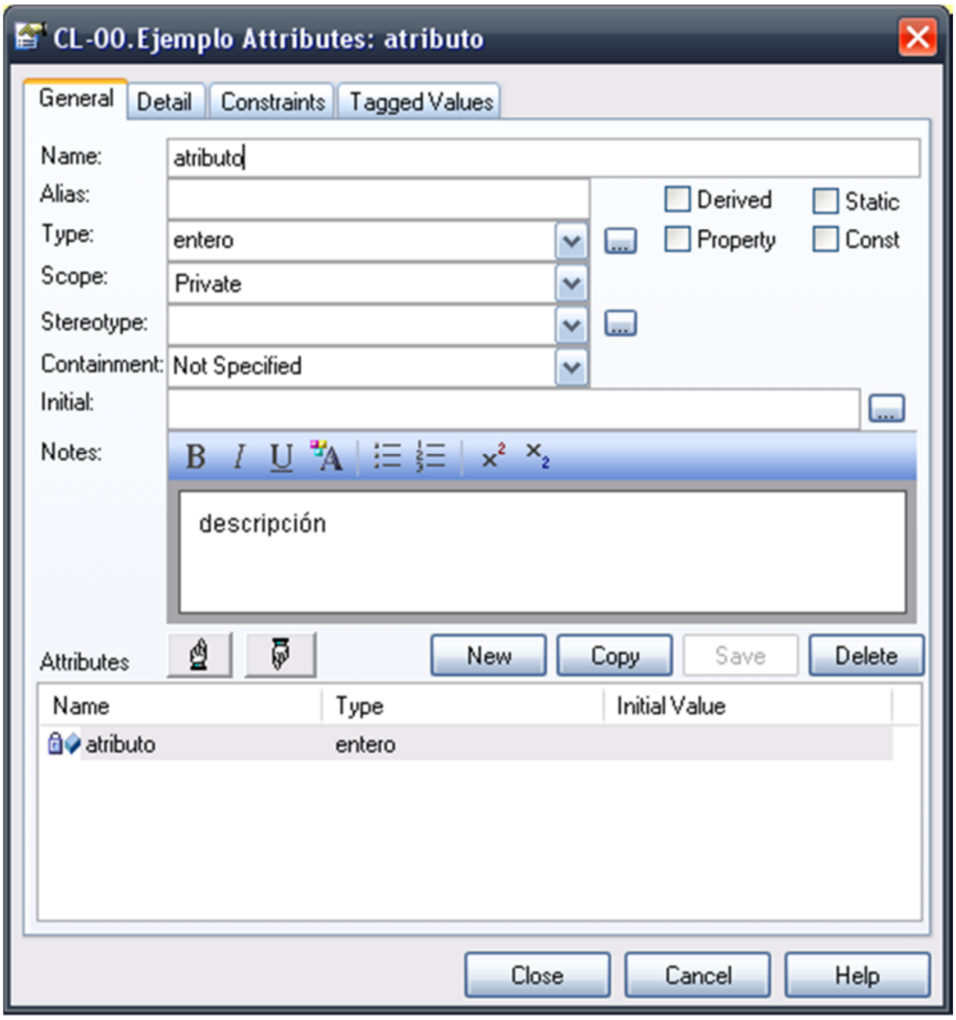

Figure 15. Screen attributes

When you define an attribute, it must have a mandatory name, type, description, and cardinality. The cardinality is completed in the Detail tab, as shown in Figure 16.

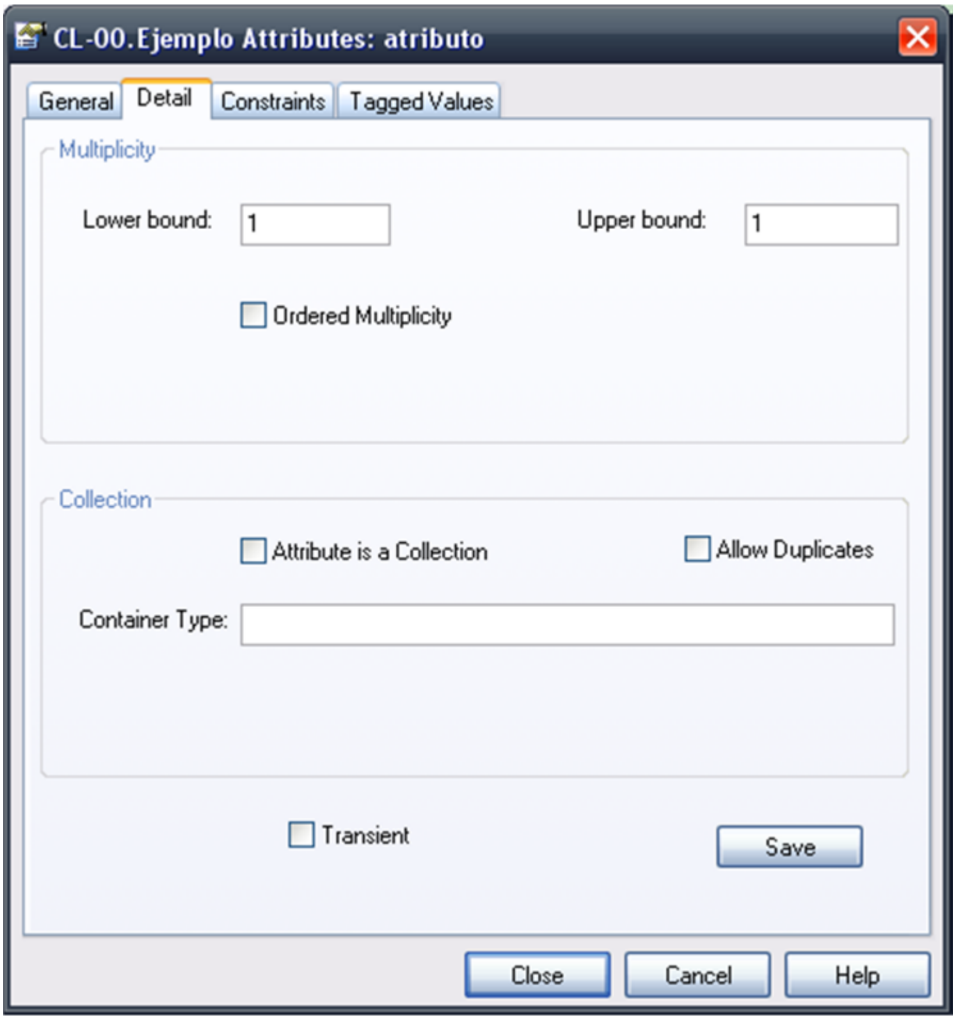

Figure 16. Entry screen cardinality

The cardinality has to be introduced in the fields Upper bound and Lower bound. For example, if the cardinality 0 ..*, would have to enter a 0 in Lower bound and a * in Upper bound. If these fields are not filled, the cardinality shown is 1 .. 1.

In Figure 17 we can see the screen through which you can add operation to an artifact, the General tab.

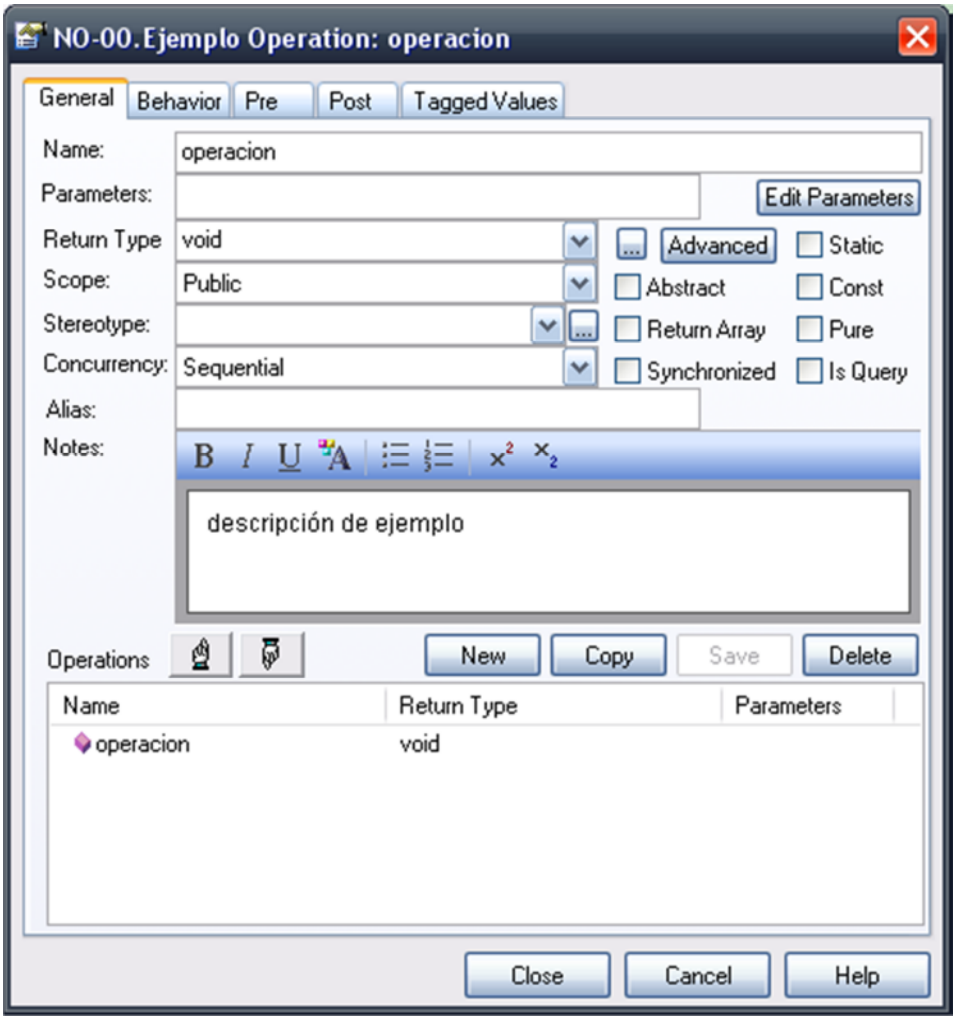

Figure 17. Screen operations

Required fields in operations are the name, description and alias. In the alias field must be the name of the RF artifact referenced by the operation. The other fields and tabs do not have a defined use in NDT, though, can be used to enhance the description of the operations.

Require tab has no functionality in NDT, so it will not be defined in this guide. Constraints tab is where you can define the different conditions that apply to the artifacts that define it. In Figure 18 we can see the contents of the tab.

Figure 18. Screen artifact conditions

A condition must be defined name (Constraint text box), the type and description. The type has to be invariant (Invariant) for RA and NA, or Pre-Condition or Post-Condition for RF and PS.

On the Links tab displaying links to other artifacts. This tab allows you to sort the artifacts linked by name, stereotype or connection, among others, as shown in Figure 19.

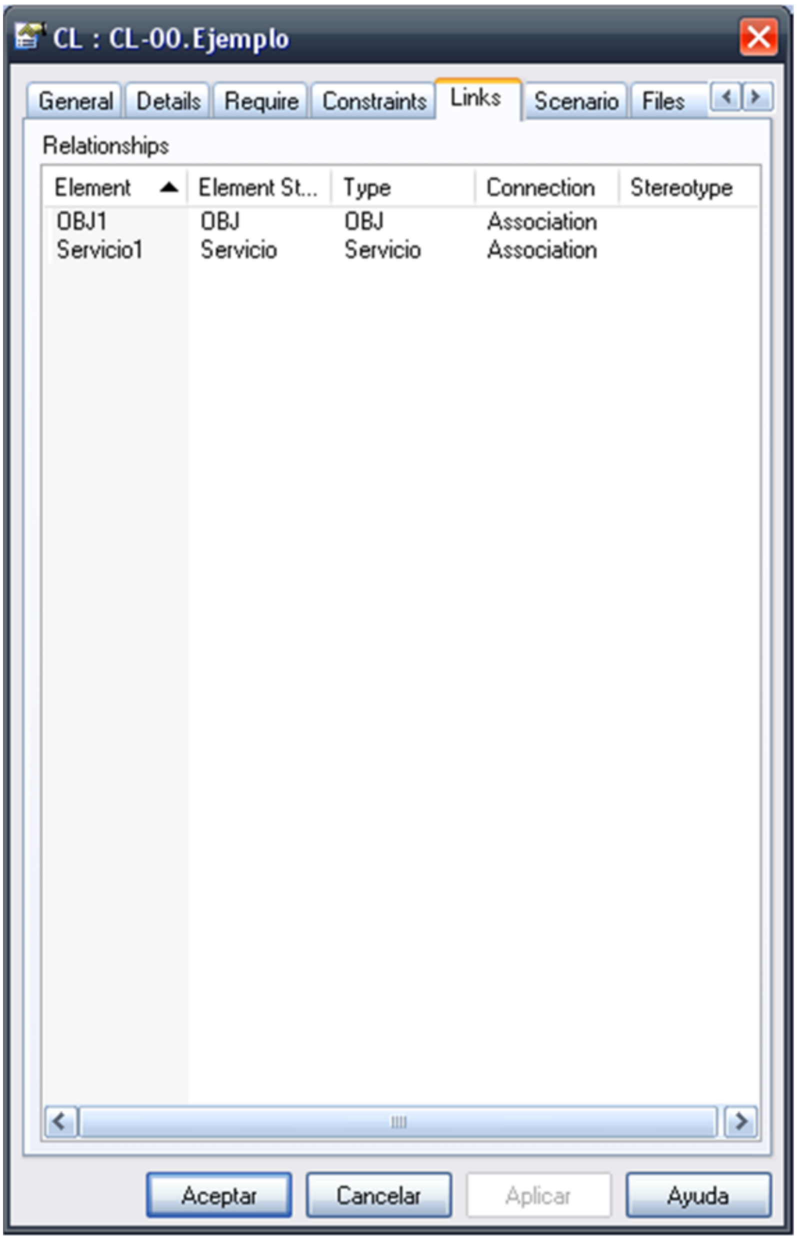

Figure 19. Visualizing the links of an artifact

Scenarios tab is used to describe the behavior of an RF through scenarios. As explained below, specifically in paragraph concerning Functional Requirements, a RF can be defined by using activity diagrams or scenarios. If you choose this second option, through this tab, shown in Figure 20, we can describe the scenarios.

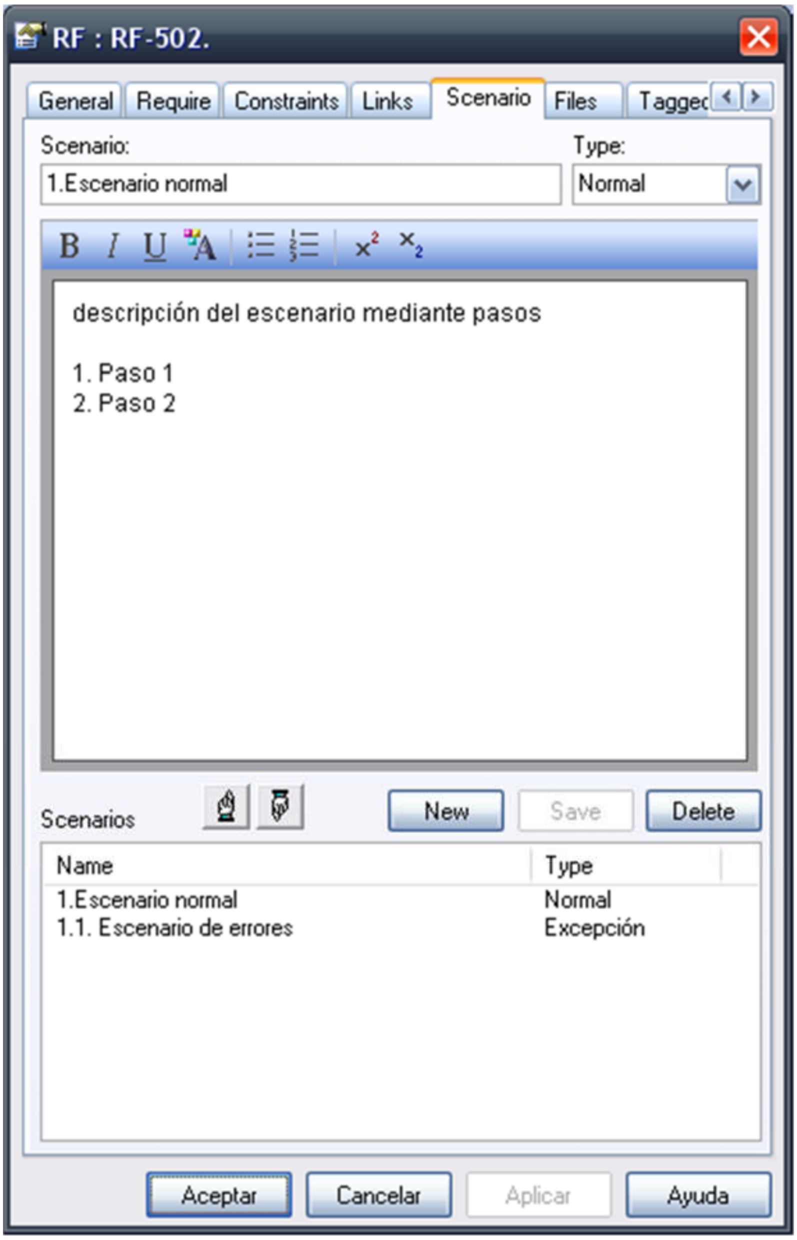

Figure 20. Defining the phases of an artifact

The scenarios should have a name, type (must be "Basic Path", Alternate or Exception, you cannot add more types in NDT) and a description detailing the steps in this scenario. There must always be a scenario like "Basic Path".

The sequence of the normal scenario is to be by consecutive numbers (1, 2, 3, ...), however, the sequence of scenarios of exception must be a composite number (2.1, 3.1, 3.2, ...). In short, the scenarios should be an ordered sequence of numbered steps and jumps should not appear in it. If desired, you can use the structured specification that provides Enterprise Architect 8.

The next tab (Files) refers to files that are linked to an artifact, either a document, image or any other file type. The interface offers this tab is the one shown in Figure 21.

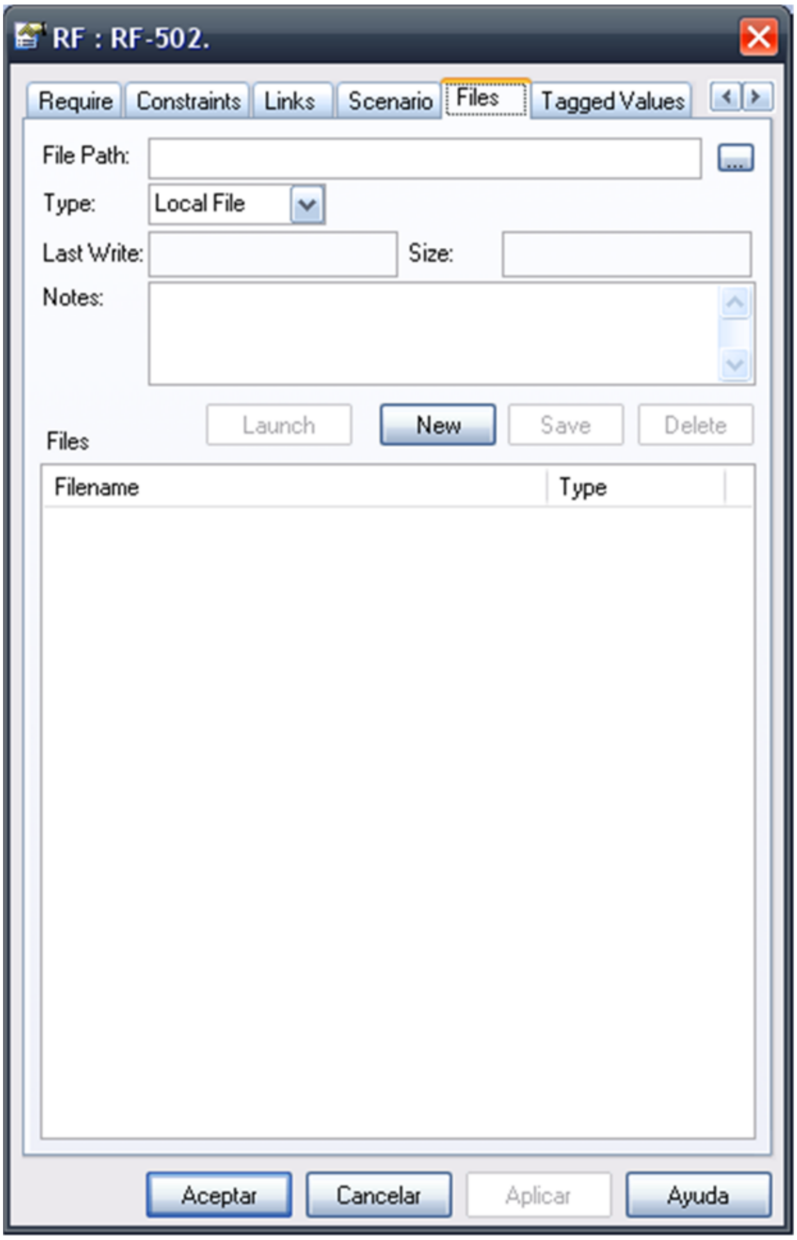

Figure 21. Tab File links

These files are external and, unlike documents that are appended to a document, no space in the project EAP. That yes, if you choose to link a local file, if the project is desired EAP send a person who has to open from another network, you must send the EAP along with the documents which have been linked. It is also recommended that along with the EAP, is to create a series of folders to hold documents that are required in the different phases that defines NDT, introduced in the File Path on the file path, not just the full path initially shown as Enterprise Architect. The folder structure we propose is the following: along with the EAP, it creates a docs folder, and within the folders control, evs, drs, das, dds, dps and dms. Each folder corresponds to each of the phases defined in NDT-Profile.

The next tab is the Tagged Values tab. This tab is shown as observed in Figure 22.

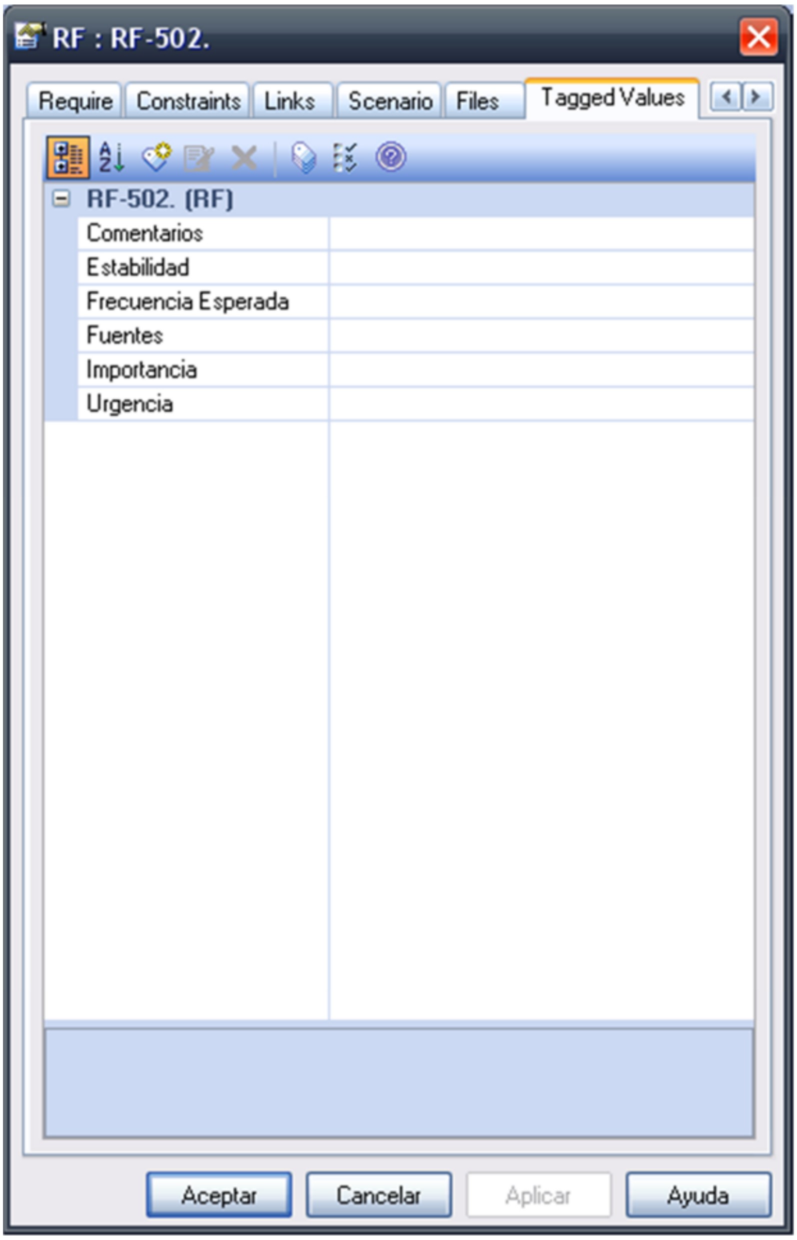

Figure 22. Display tab labeled values

The functionality of this tab is to show tagged values to be defined for each artifact. Displays the same information as the Tagged Values tab as described in paragraph **¡Error! No se encuentra el origen de la referencia.**, that is, the left is the name, and takes the value right.

4.3. Connectors

In NDT artifacts can be linked through the connectors. The connectors are links between two artifacts with a number of properties described in this section. We can see an example of two artifacts (in this case, an actor and a functional requirement) linked to a Use type connector in Figure 23.

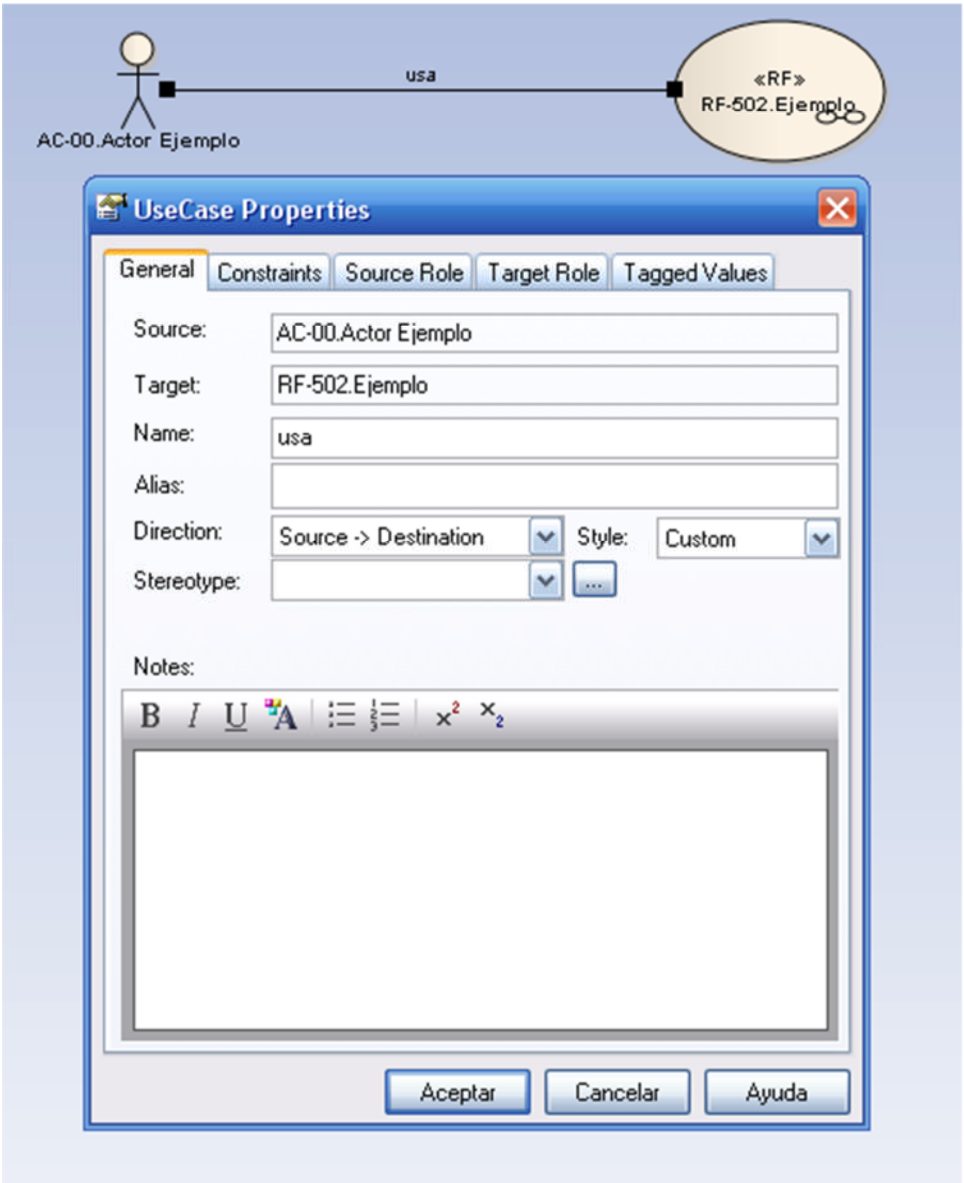

Figure 23. General properties of a conductor

On the General tab to observe the basic properties of the connector. In the title of the window shows the name of the type of connector that has been used in this case UseCase. This type of connector is generic UML, although there are many other connectors that are specifically defined UML extensions in NDT-Profile. These connectors are specified within the artifacts that make use of them in the paragraphs of the life cycle phase it is concerned, although the properties of the connectors are common to all.

In the Source field shows the artifact of origin of the connector, and in the Target field shows the destination of the connector artifact. These fields are not editable because it is an intrinsic property of the connector. In the Name field can give a name to the connector, which will appear next to the line drawn in the diagram. In the Direction field we can select the direction in which the connector (Source-> Destination, Destination-> Source, Bi-Directional and Unespecified), there is an arrow indicating the direction in those certain types of connectors. In the field Stereotype can give the link. In the case of links defined specifically for NDT-profile, this field displays the stereotype that extends the link, no amendment is advisable. In addition, the Notes field you can add a description to the link, but is not binding in NDT-Profile.

In the next tab, Constraints, as shown in Figure 24, we can define conditions for the link.

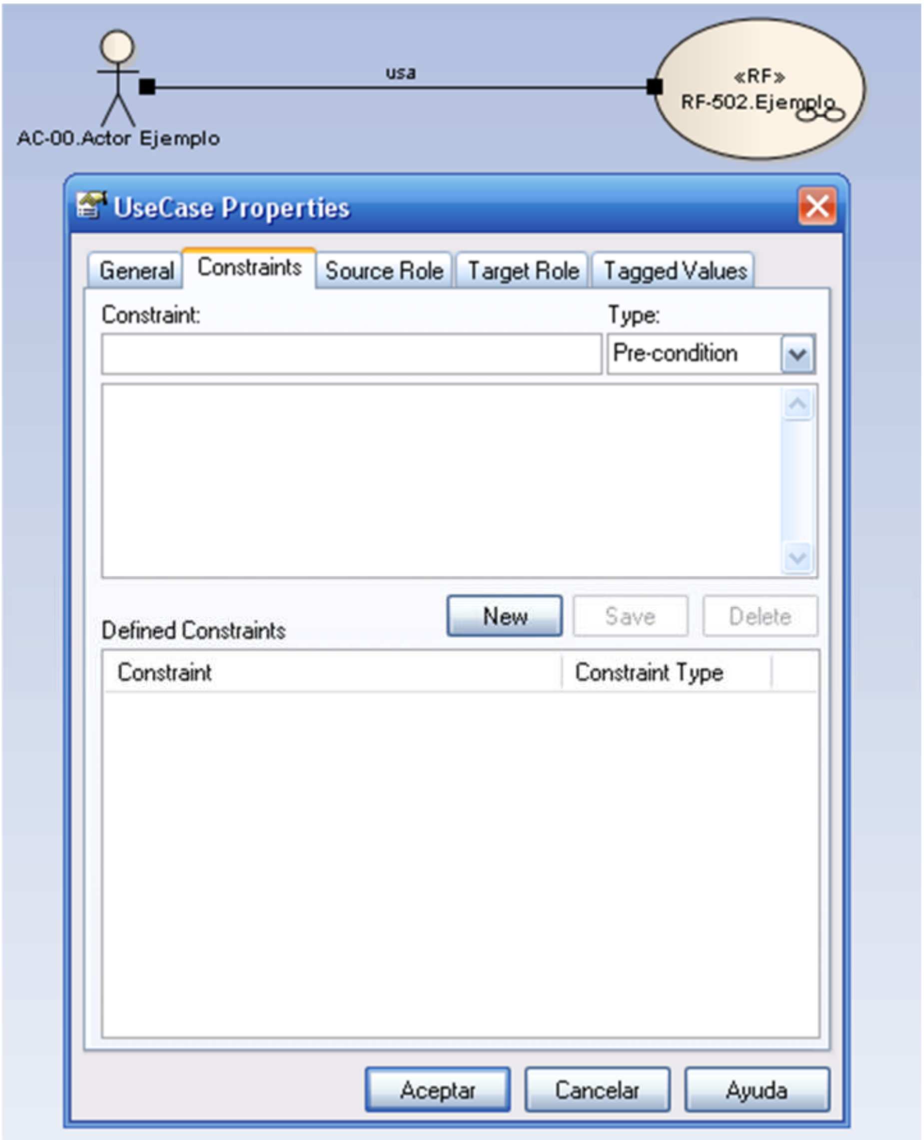

Figure 24. Tab to define conditions in a connector

This tab is exactly equal to the Constraints tab of the artifacts, as described in the preceding paragraph.

The two tabs below, Source and Target Role Role, refer to the role of origin and destination of the link. Both tabs are equal and can be seen in Figure 25.

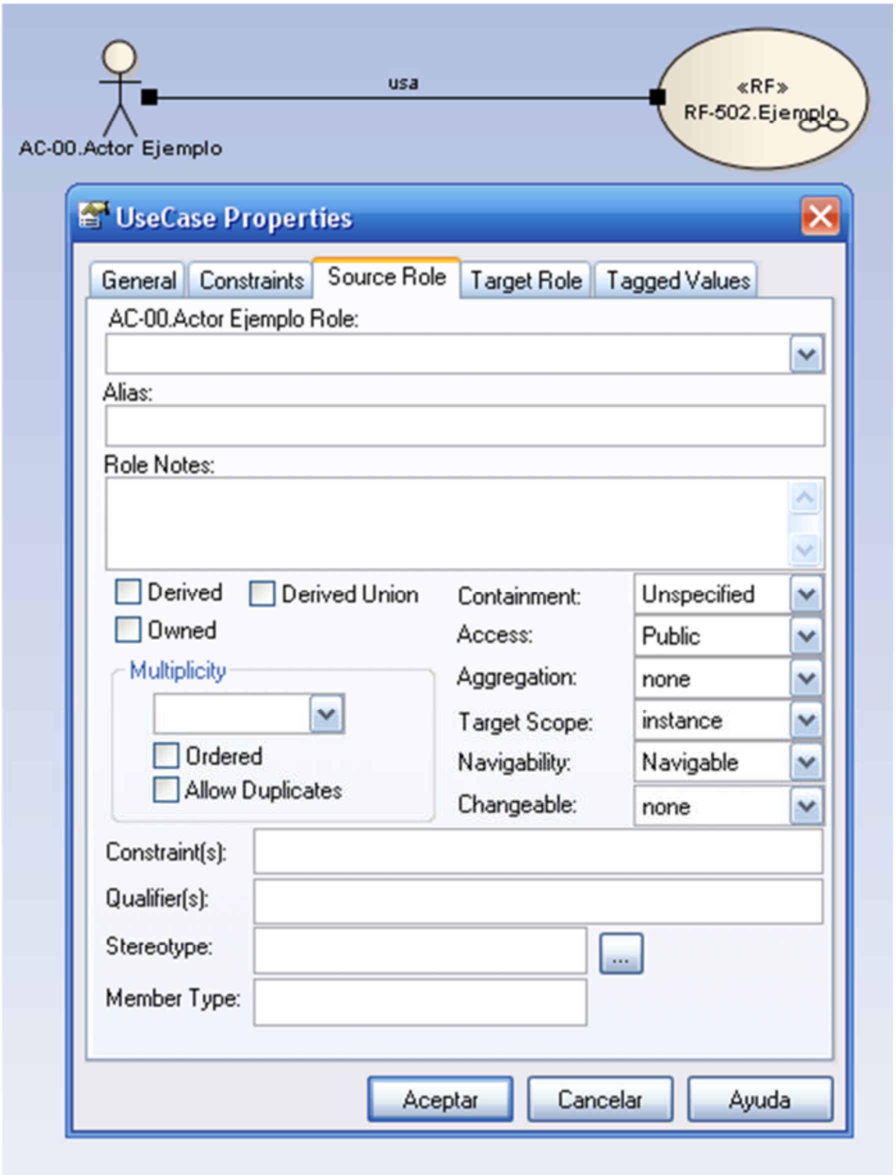

Figure 25. Tab target source and role of a connector

This tab has special relevance within NDT-Profile, in the transformation of storage requirements (RA and NA) to artifacts of the content model of the analysis phase (CL and CLn).

If using NDT-Driver for automatic processing, and focusing on the transformation RA to CL, there will be a link between two CL if any of the RA of which must have a specific data type another RA. Then, the role will link the name of the specific data as cardinality (Multiplicity), the cardinality of specific data. If this were done manually, the role should be selected with the first combobox that appears, which show the attributes of the source element in the case of being in the Source tab Role, or destination, should be Role the Target tab. The cardinality of the link is selected with the combobox Multiplicity.

The remaining fields, and Tagged Values tab, do not have specific functionality NDT-Profile, although its use is not restricted and is allowed to enrich the description of the connectors.

5. Good practices common to all elements

5.1. Subsystems

In some large systems may be convenient to separate each set of artifacts in different folders or sub-systems. To do this, there is a toolbox for each subsystem artifact that creates a new folder and within this, the diagram for the type of artifact that created it. So, we got to classify each type of artifact in different folders. These artifacts, despite being in different folders can be dragged to any diagram, besides being able to trace through matrix or diagram a more organized way.

5.2. Traceability Matrixs

A traceability matrix is an Enterprise Architect screen (Figure 26) where the rows represent the elements of a model defined by the user and the columns represent other elements of another model (or even the same model) is also defined by the user. Selecting a cell (intersection of a row with a column, ie intersection of two elements) Enterprise Architect automatically adds a link (in this case a traceability relationship) between the two.

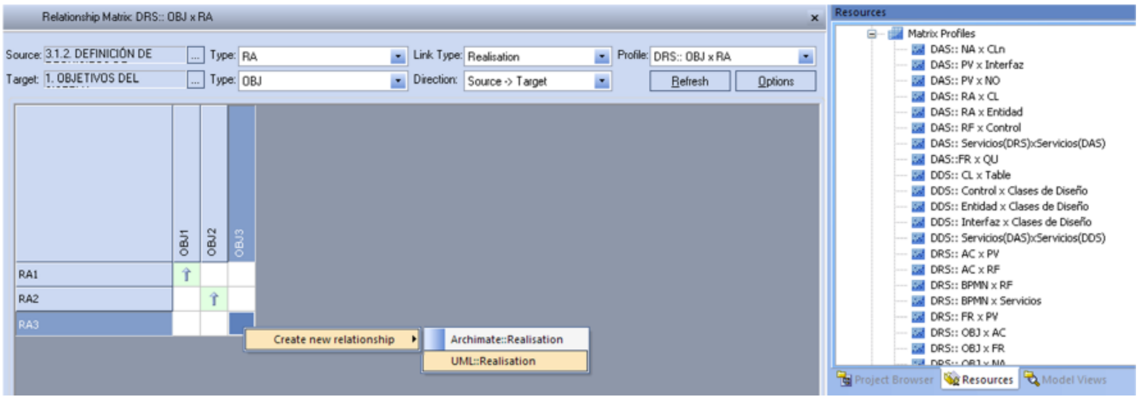

Figure 26. Sample traceability matrix

All artifacts should have defined traceability relationships. Thus, at any time is possible to know what objectives to satisfy the requirements, what requirements implement the classes, which classes persist in the database tables, etc.

The origin of the traceability relationship should be the element that implements and destination of the item implemented. For example, between a functional requirement of the requirements phase and a class analysis of traceability there is an association with class origin and destination functional requirement if the class participates in the implementation of behavior defined in the functional requirement.

Traceability matrixs included in NDT-Profile are described in Table 7:

Table 7. Traceability Matrix

Phases	Matrix	Description
DRS	OBJ x RA	Traceability between the objectives of the system and related storage requirements
DRS	OBJ x NA	Traceability between system objectives and new natures
DRS	OBJ x AC	Traceability between system objectives and actors
DRS	OBJ x RF	Traceability between system objectives and functional requirements
DRS	OBJ x FR	Traceability between system objectives and frases
DRS	OBJ x LI	Traceability between system objectives and listings
DRS	OBJ x PV	Traceability between the objectives of the system and display prototypes
DRS	OBJ x RNF	Traceability between the objectives of the system and non-functional requirements
DRS	BPMN x Servicios	Relations between business model and services identified
DRS	BPMN x RF	Relations between business model and requirements functions
DRS	BPMN x Servicios	Relations between business model and services identified
DRS-DAS	Servicios (DRS) x Servicios (DAS)	Traceability between services requirements and services analytical
DRS-DAS	RA x CL	Traceability between the requirements of storage and persistence classes
DRS-DAS	NA x CLn	Traceability between the new nature and persistence classes
DRS-DAS	RF x CP	Traceability between the functional requirements and the kinds of processes
DRS-DAS	FR x QU	Traceability between the frases and queries
DRS-DAS	LI x IN	Traceability between lists and indexes
DRS-DAS	PV x NO	Traceability between prototypes and visualization nodes
DAS-DDS	Servicios (DAS) x Servicios (DDS)	Traceability between the services of analysis and design services
DAS-DDS	CL x AD	Traceability between classes of content and data access classes
DAS-DDS	CLn x AD	Traceability between classes nature of content and data access classes
DAS-DDS	CP x NE	Traceability between classes and types of business processes
DAS-DDS	QU x PR	Traceability between queries and presentation classes
DAS-DDS	IN x PR	Traceability between the indexs and types of presentation
DAS-DDS	NO x PR	Traceability between the nodes and presentation classes
DAS-DDS	CL x Table	Traceability between content classes and the tables of physical data model
DRS-DPS	RF x PS	Traceability between the functional requirements and system testing
DRS-DPS	RF x PA	Traceability between requirements and functional acceptance tests

5.3. Best practices for artifacts

These rules apply to all artifacts to fill in all phases Documents EVS, DRS, DAS, DDS, DPS y DMS.

All artifacts (requirements, use cases, classes, test cases, packages, etc..) Should have a name and description that provides value. In addition, as discussed below, all elements have a numerical code. This code may not match the numbering of two artifacts of the same type.

Every element must have a description, including specifications, scenarios, etc.

5.4. Linking the documents

Along with the delivery of the file with Profile EAP is to attach a folder called docs, which contains all documents attached in the different Document type artifacts that appear in different folders for each phase. Thus, in the docs folder will turn the following folders: control, evs, drs, das, dds, dps and dms.

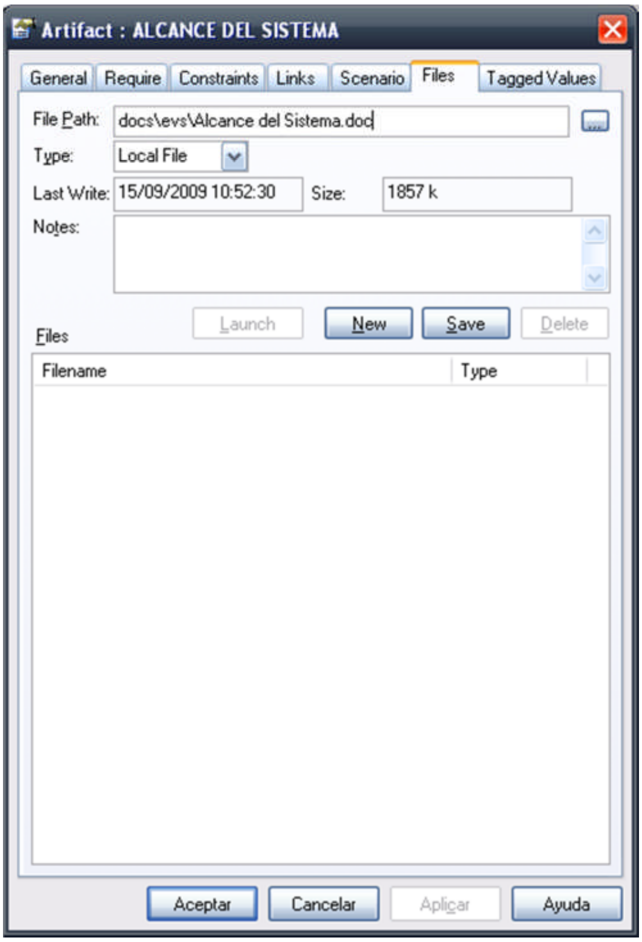

Figure 27. Linking the documents

To Link a document you have to double click on the Document appliance, go to the Files tab and the File Path field select the file you want to attach. Then you have to leave only the relative path (eg docs \ drs \ Documento.doc), since this way you can open the document from Enterprise Architect. This can be seen in Figure 27.

If the document is not very extensive, you can enter in the corresponding fixture Notes field, since from version 7.5 of Enterprise Architect has a WYSIWYG editor that lets you several options for formatting text.

Table 8. Rules Documents

Documents
Document type artifact must have a document link, or if it is not too large, the content is entered in the Notes field.
File Path field must contain a relative path, so you can open the document directly from the Enterprise Architect.

6. Participants

This section should include all instances of participants indicating their data, roles, and so on. ... This will be used artifacts Participants toolbox, selecting the appropriate artifact depending on the category of participant. Figure 28 shows the toolbox Participants.

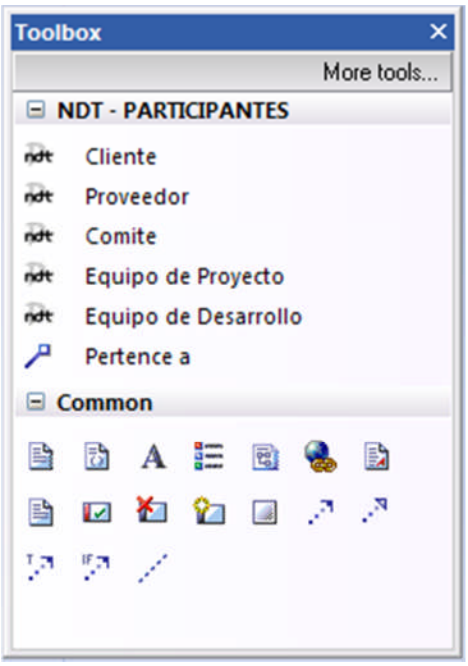

Figure 28. Toolbox Participants

The following table lists the artifacts in Participants and structure of the name.

Table 9. Nomenclature of Participants

Requirement	Name
Client	Name
Provider	Name
Committee	Name
Project Team	EP-XX. Name
Development Team	ED-XX. Name

The artifact represents the Customer of the final product, so will the name of the agency or company represents.

The artifact represents the company provider that develops the system.

The artifact Committee representing each of the persons belonging to the various committees that had the project (eg, monitoring, work, etc.). May be associated with the Client.

The artifact Project Team represents the people involved in the project. Typically there is a role defined by the value labeling. May be linked to Provider or Client.

The artifact development team representing the developers of the system. May be linked to Provider or Client.

All artifacts have the same standard properties (name, author, description). Here, we showed one of them:

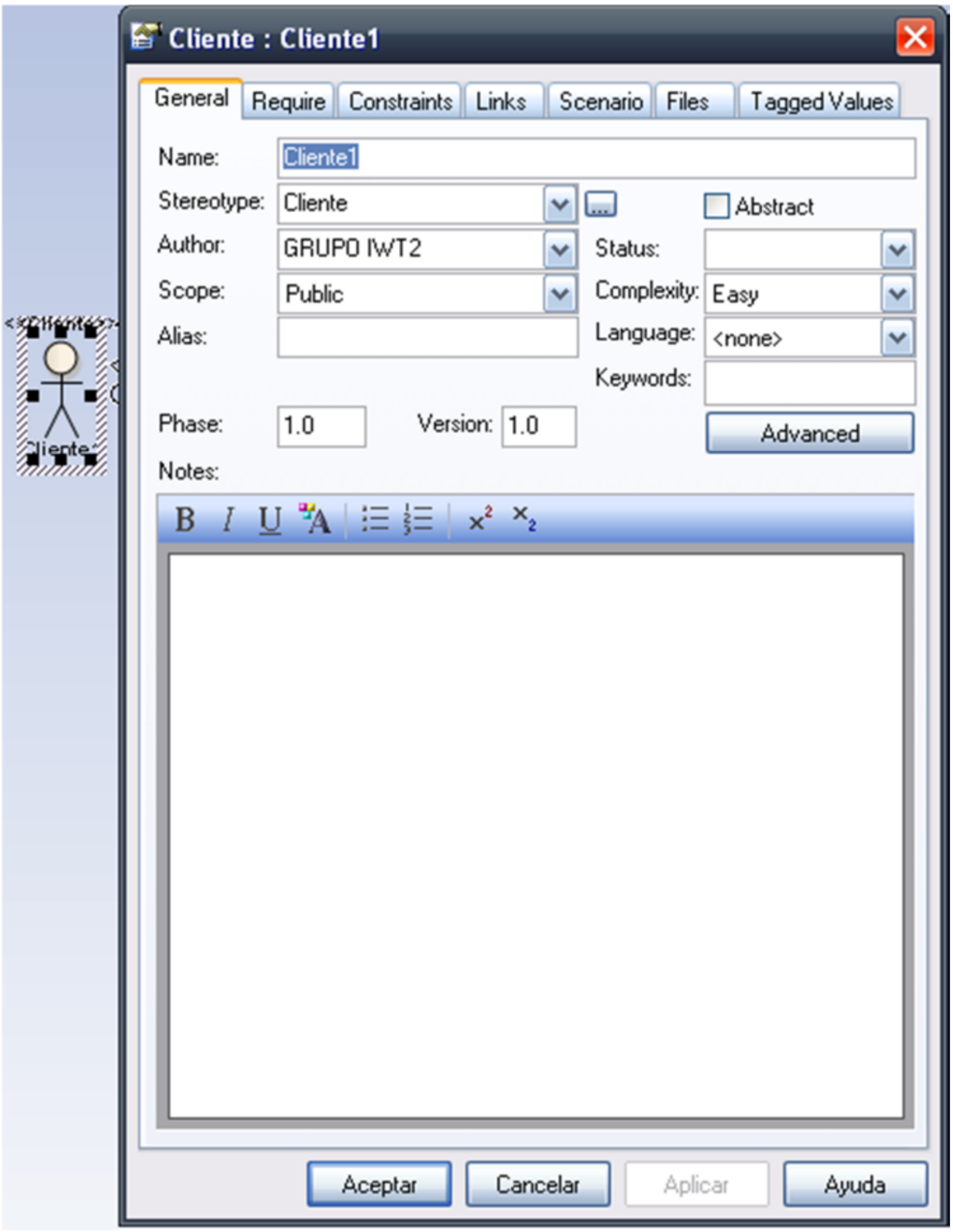

Figure 29. Standard properties of Participants

No tagged values for participants Client, Provider and Committee, but there is a tagged value type participants Participant Team and Development Team Project. For these, the value has to be completed mandatory labeling.

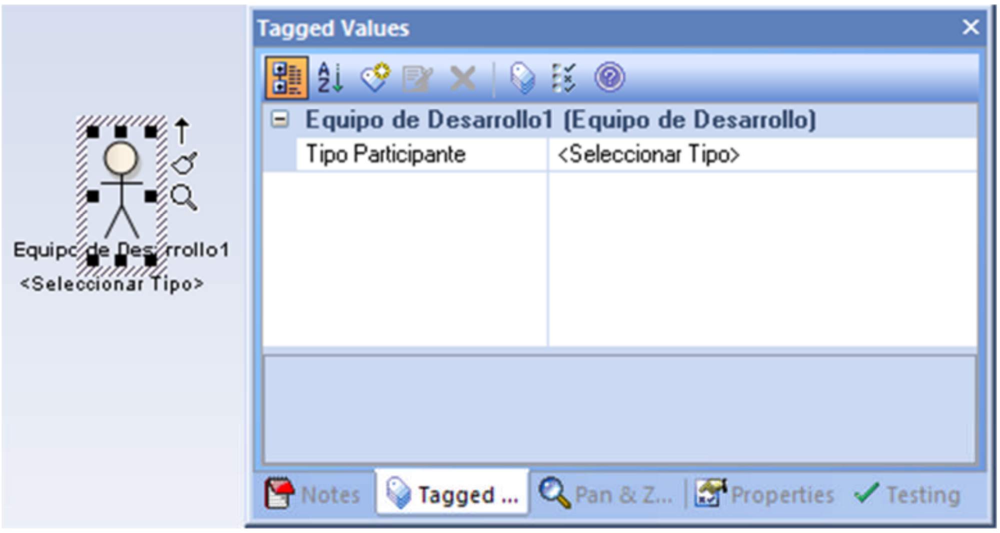

Figure 30. Participants tagged values

Table 10. Rules of Participants

Diagrams of Participants
Must contain all participants according to the outline of the organization
The only links that should appear are those of the toolbox
The artifact name of the Project Team is an EP-XX (X).Name
The name of the Artifact Development Team is ED-XX (X).Name
Project Team and Development Team should be selected on the values labeled type

7. Version Control

The change control sheet, has a description of each of the versions to be delivering the project. You will not need to enter a profuse detail because it is managed through baselines, as explained in paragraph 3.3, but I shall explain the changes to the generic level.

For this section offers only an artifact, the artifact Version Control. Below are properties Version Control standards.

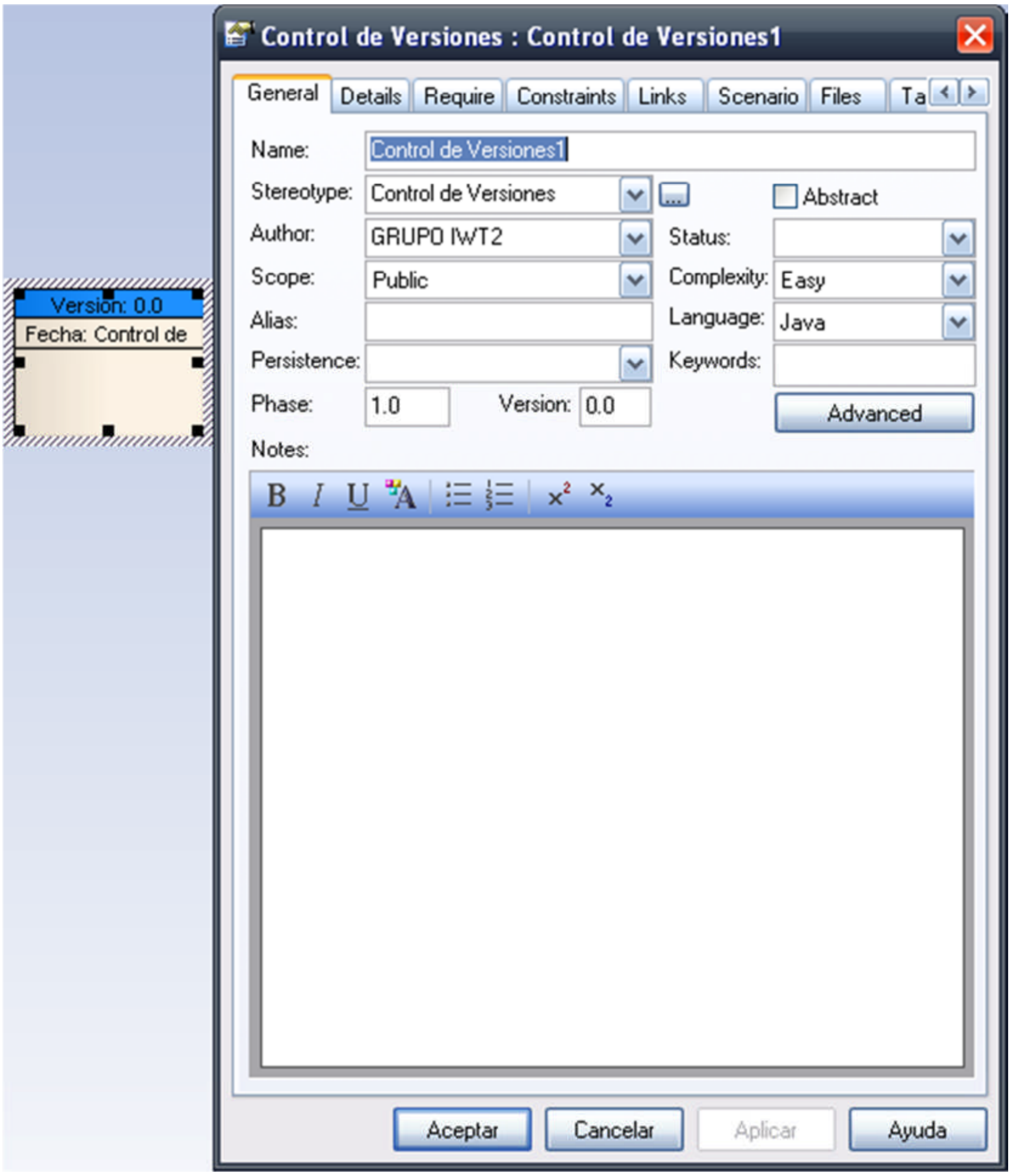

Figure 31. Properties Version control standards

The standard properties are the name (which must be present), version, description and baseline file (which will be associated as if a document is involved, as explained in paragraph5.4).

Table 11. Version Control Rules

Version Control Diagrams
There must be many version control artifacts as there are versions of the document.
There must be many baselines (xml) as there are versions of the document.
Each version control artifact must have a baseline linker. This document should be in the folder docs\baseline\. In the Files tab in the File Path field must have the relative path.

8. Project Objectives

This paragraph shall include either a linked document or a generic description of the project in the Notes field, if it is not very extensive

In any case, the information provided should not be too detailed and should provide an overview of the objectives to achieve with the project.

Table 12. Objectives Rules

Document Objectives
Link it must have a document. This document should be in the folder docs\objectives\. In the Files tab in the File Path field must have the relative path.
If the document is short, you could fill in the Notes field and would not require the linking of the document.

9. System Viability Study (EVS)

The structure of the documentation of the Viability study is shown in Figure 32.

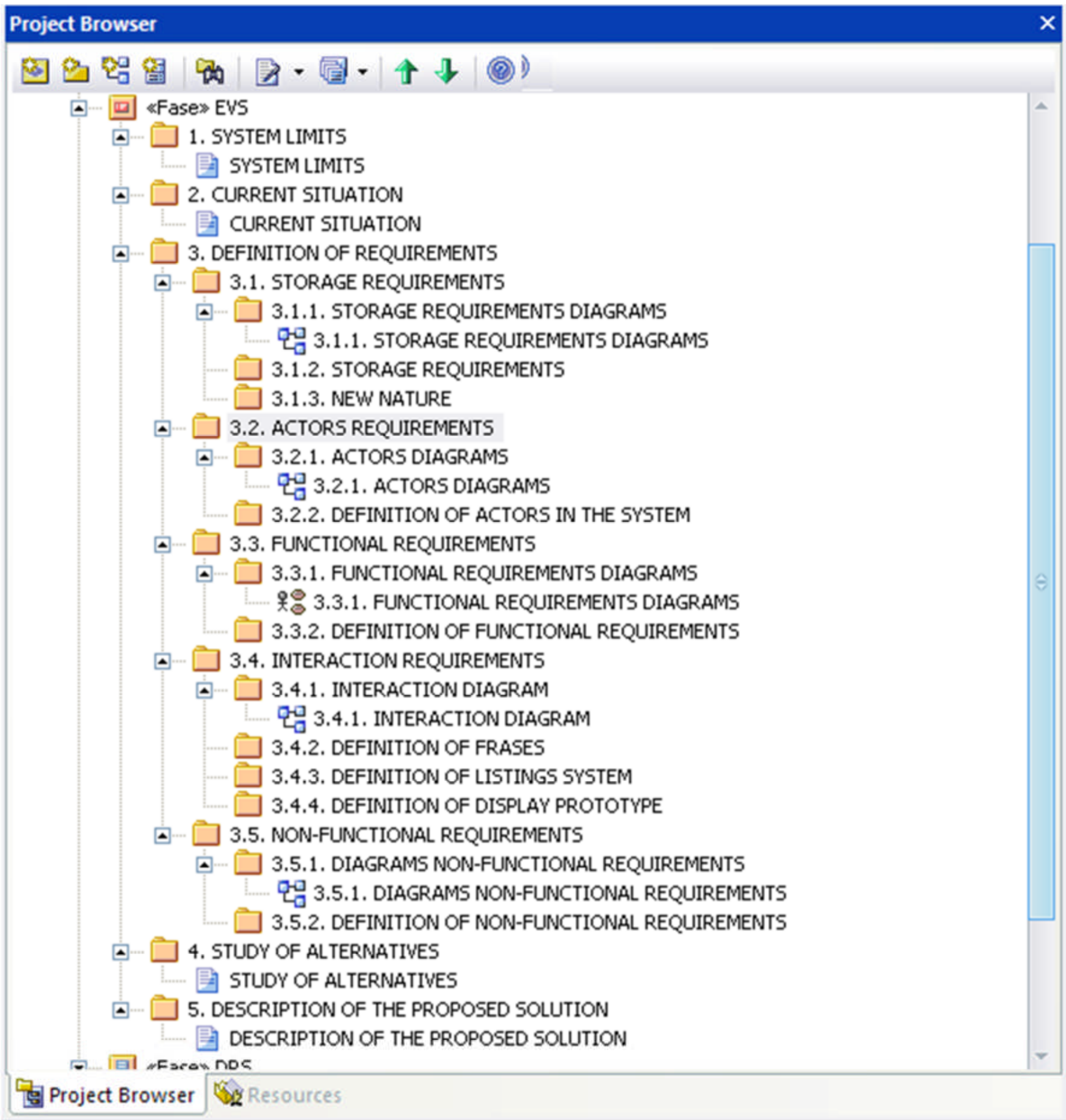

Figure 32. EVS Structure

In addition, it has been defined in the profile of NDT a set of tools for the study of Viability. For the definition of the artifacts should be used artifacts existing set of tools that will be charged to open the diagram you want to describe.

The following table lists the artifacts in the Viability study phase, and the structure of the name.

Table 13. Viability Study Nomenclature

Requirement	Name
Storage Requirement	RA-XX.Name
New nature	NA-XX.Name
Actor	AC-XX.Name
Functional Requirement	RF-XX.Name
Frase	FR-XX.Name
Display Prototype	PV-XX.Name
Listing	LI-XX.Name
Nonfunctional requirement	RNF-XX.Name

In this section, as well as setting the corresponding artifacts, it should create a series of documents such as System Limits, Current Situation, Study of Alternatives and Description of the Proposed Solution. These documents should be attached to the EAP file as described in Section 5.4.

The artifacts of the Viability Study are exactly the same as defined in the following paragraph (requirements), although at the phase of Viability study are not described as deeply.

If a project is not implemented Viability Study, this folder has been deleted.

10. Requirements

The following describes the various NDT artifacts defined for the Requirements, how to enter the information into the tool Enterprise Architect and rules to follow.

As mentioned earlier, each artifact has an identification code. The codes for each artifact are shown in Table 14.

Table 14. Nomenclature Requirements

Requirement	Name
Objective	OBJ-XX.Name
Service	Service -XX.Name
Storage Requirement	RA-XX.Name
New nature	NA-XX.Name
Actor	AC-XX.Name
Functional Requirement	RF-XX.Name
Frase	FR-XX.Name
Display Prototype	PV-XX.Name
Listing	LI-XX.Name
Nonfunctional requirement	RNF-XX.Name

The structure of the system requirements document (DRS) shown in Figure 33.

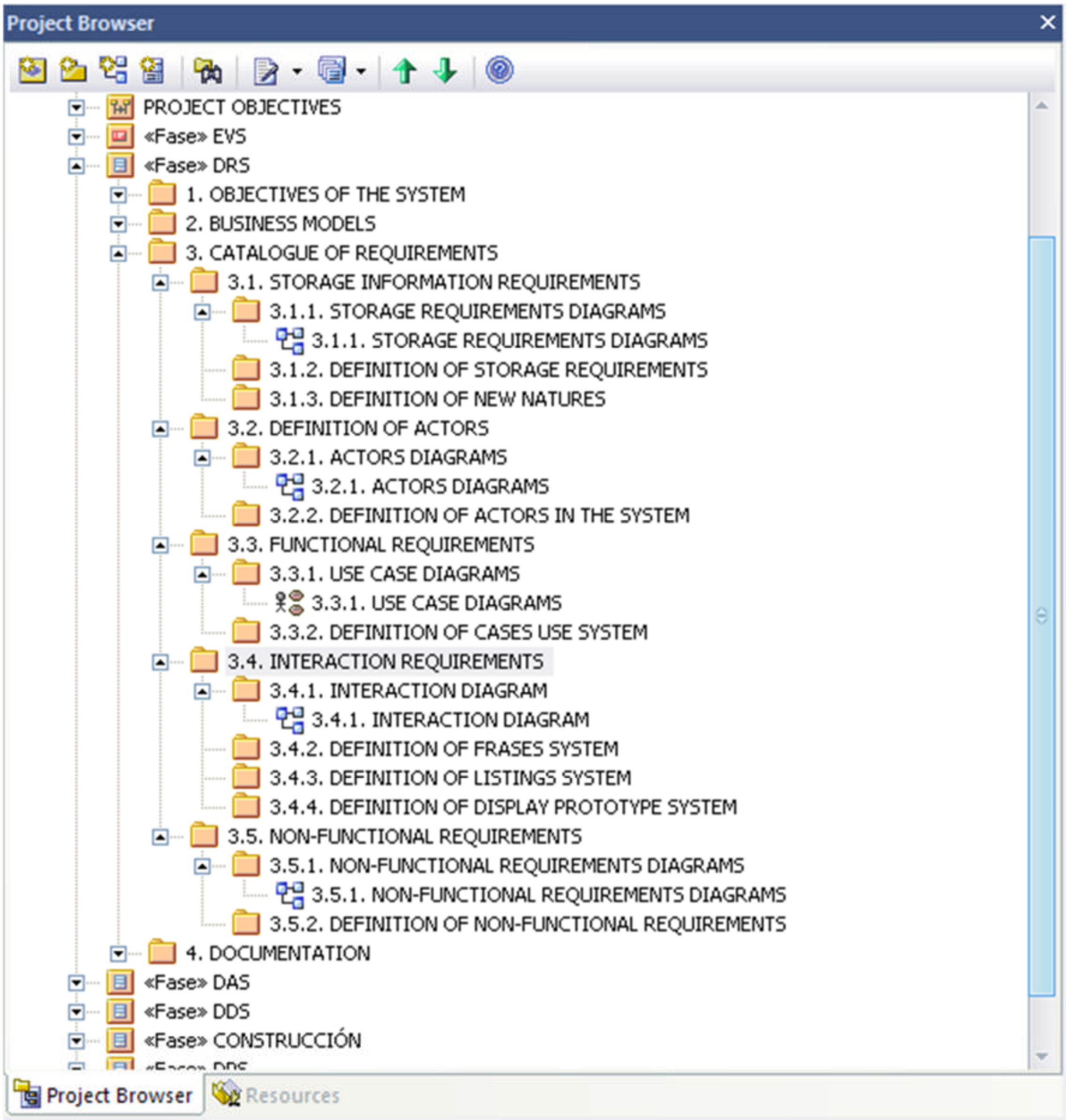

Figure 33. DRS Structure

Corresponding tools for the definition of requirements artifacts are shown in Figure 34.

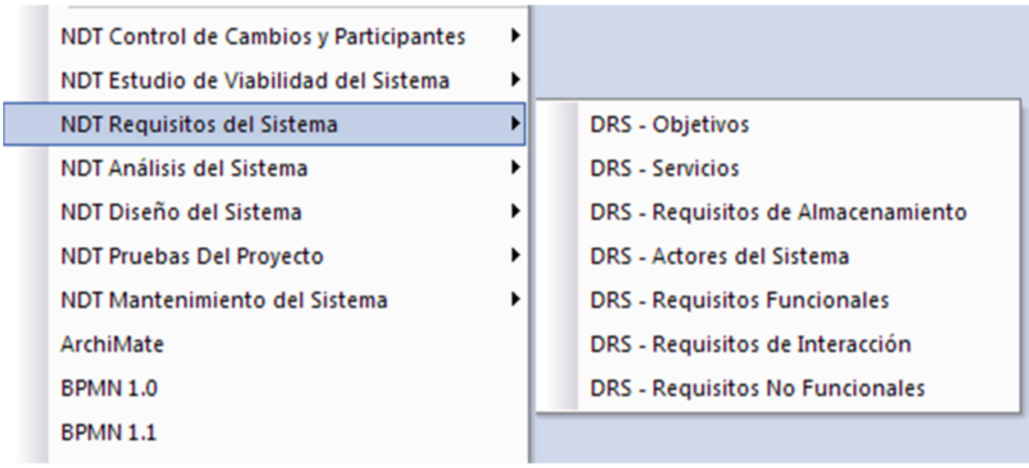

Figure 34. Toolboxes customization

10.1. System Objectives

The objectives describe the needs to the system. These objectives are identified in interviews with clients and users, and defined by the artifact that provides OBJ in NDT-Profile. Figure 35 shows the objectives defined for the toolbox.

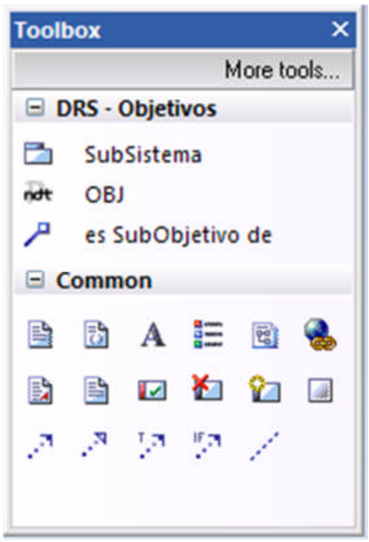

Figure 35. Toolbox for goals

To create an objective, just click on the toolbox in the OBJ and drag to the diagram where you want to model. Figure 36 shows an objective standard properties and tagged values.

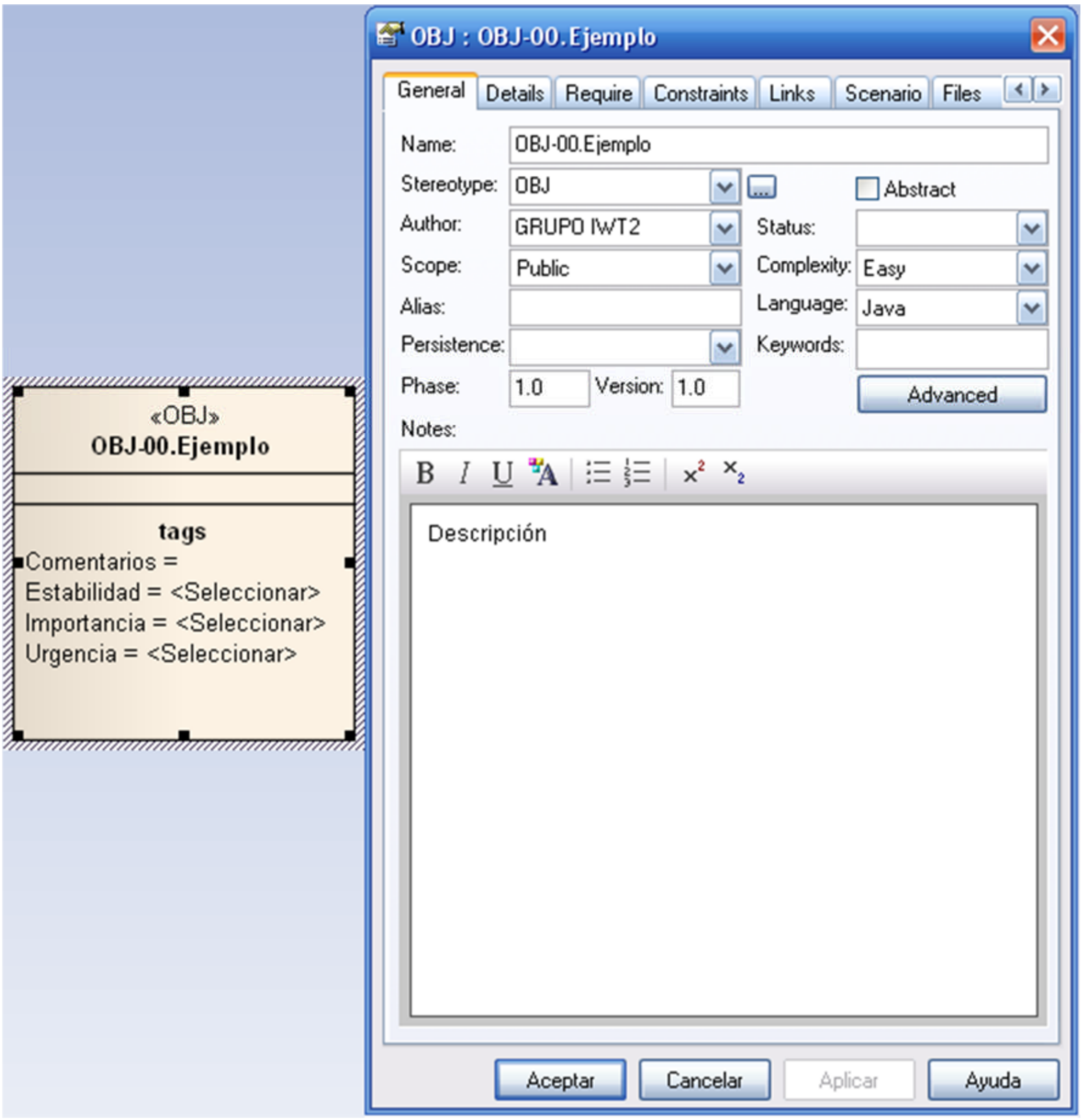

Figure 36. Properties of an objective standard

In the artifact OBJ there are a number of fields that are mandatory for the description of the objectives is considered correct. This series of fields are:

- **Name** (Name field): Each objectives hould be classified with a code and a descriptive name. As shown in Table 14, the name must meet the following format OBJ-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of objectives.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the target.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the objective being treated.
- **Stability** (tagged value): This value reflects the probability of labeling an object through changes in its definition.
- **Importance** (tagged value): This value indicates the importance of the fact that the system meets the goal for the client.

- **Urgency** (tagged value): This value indicates the urgency.

There are also other fields are optional, but recommended that the form is completed for a better definition of objectives. These are:

- **Status** (Status Field): This field contains the situation where the objective is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the objective. This field is meaningless when it comes to large systems where version management is a critical task.
- **Attributes** (in Details tab, Attributes button): Optionally you can add attributes to the objective if deemed necessary. How to add attributes is explained in paragraph 4.2 (Figure 15).
- **Comments** (tagged value): This field allows the author to the objective indicate any other information it deems appropriate.

Objectives can be linked together through the connector "**is SubObjective of**" which appears in the toolbox, as seen in Figure 35. This connector denotes an aggregation relationship between objectives. To use this connector, simply click on the toolbox to select, click on the source object and drag to the target destination in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Then, in Table 15, we make a short summary of those rules that must meet the definition of a NDT-Profile target.

Table 15. Objectives Rules (OBJ)

Object diagrams
Diagram has a artifact OBJ
All OBJ must be contained in the diagram
The only links that should appear are those of the toolbox
Defining Objectives
The name of the artifact is OBJ-XX (X) Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
There is no other with the same number OBJ
Stability of tagged values, importance and urgency are required
The only link between OBJ is the toolbox " is SubObjective of "

10.2. Business model

When the development process is oriented to services and SOA environments, or when there is no defined a Viability study and want to get an overview of the business models of the system, this section should detail NDT-Profile.

10.2.1. Process models

In the first of the folders that comprise the business model (process models) will document all system business process models using BPMN. In the folder find a diagram of processes. If we add more diagrams, there is no need to create it by selecting the type of diagram shown in Figure 37.

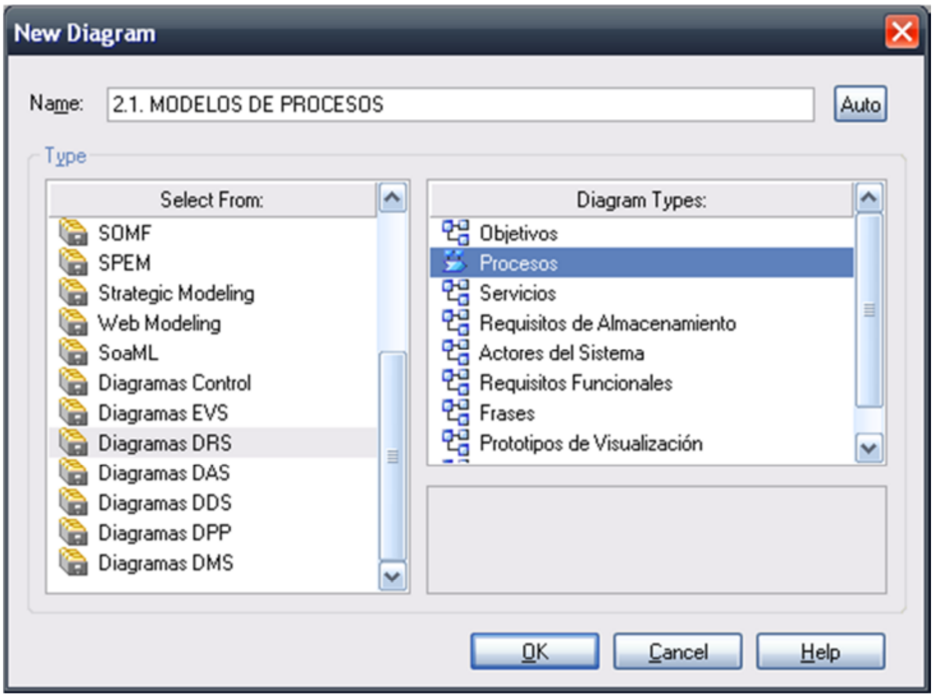

Figure 37. Create a new Process Diagram to define the business model

10.2.2.Identifying Services

Within the business model we also find a folder for the identification of services. Here we describe the existing services that have been identified as capable of being incorporated into the system. Figure 38 shows the defined toolbox for services.

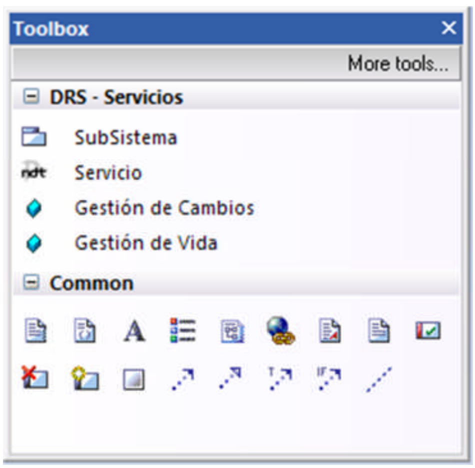

Figure 38. Toolbox for services

To create a service, just click on to the artifact Service toolbox and drag to the diagram where you want to model. In Figure 39 to observe the standard properties of services.

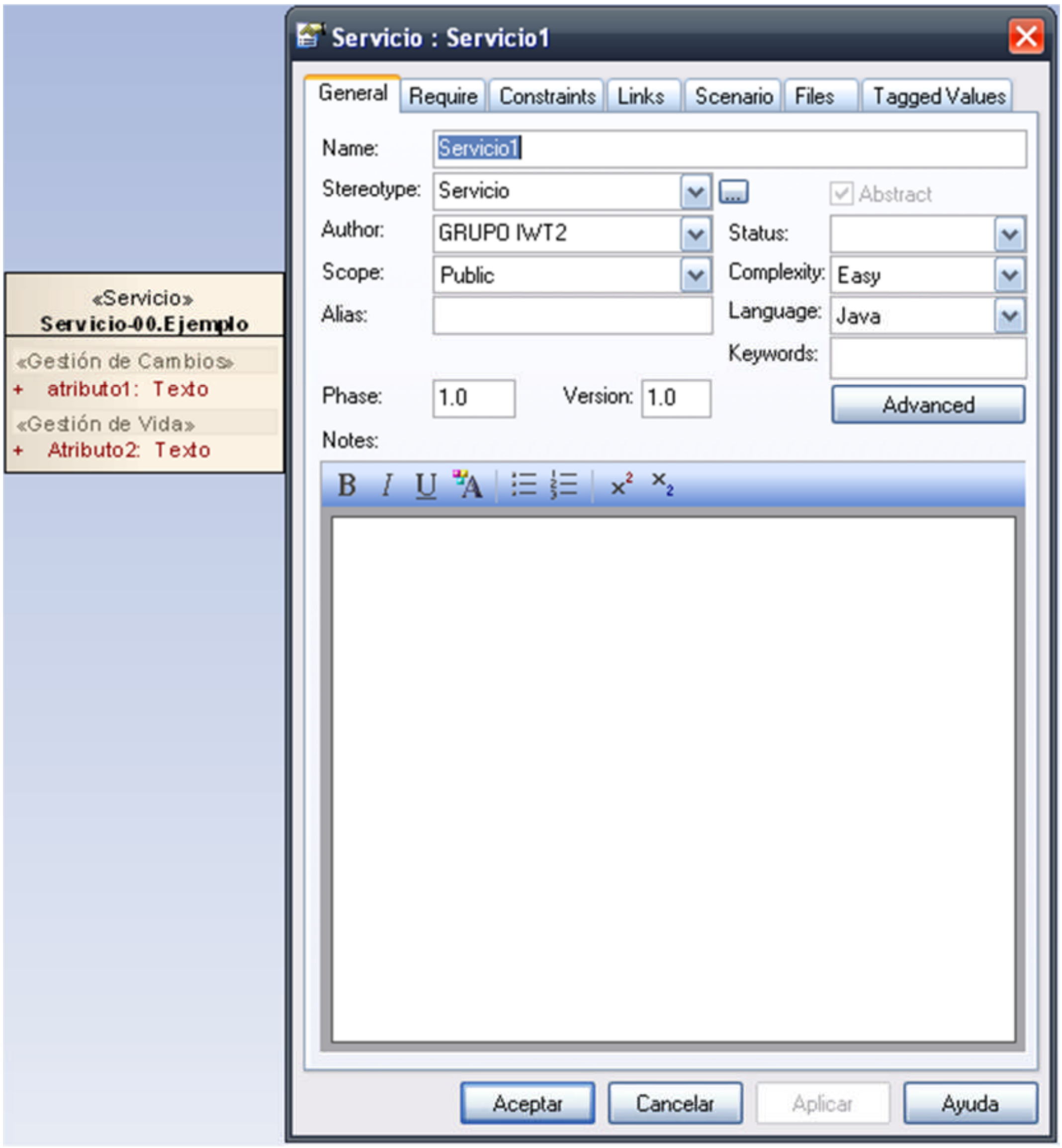

Figure 39. Properties of a Service Standards

Service in the artifact there are a number of fields that are required for the service description to be correct. In the case of service attributes, they are added to the artifact from the Toolbox DRS-Services, seen in Figure 38. To do this, click on the toolbox is the attribute to add and drag the desired service. A screen to assign the desired value.

These mandatory fields are:

- **Name** (Name field): Each goal should be classified with a code and a descriptive name. As shown in Table 14. , the name must meet the following format Service-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a high number of services.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the service.

- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the service that is being treated.
- **Change management** (attribute): It's data of the versions, branches of development, obsolete features, compatibility, date related, responsible for the changes, reasons for them, history of previous versions, etc. ..
- **Management of life** (attribute): Data related to the state it was found service in certain contexts (development, testing, integration, production). For example, maintains that the active versions of a service development are x and x +1, while the output is x-3. It also maintains data dependencies (should be minimal in the case of services) which may be by the membership of service to a BPM.

There are also other fields are optional, but recommended that the form is completed for a better definition of services. They are:

- **Status** (Status Field): This field contains the situation where the service is in its development process. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the service. This field is meaningless when it comes to large systems where version management is a critical task.

Table 16. Service Rules

Service Diagrams
You must have a type diagram Services
All Services are to be contained in the diagram
Definition of Services
The name and description must be filled
Each service that is not the repository of services has to be at least the attributes that appear in the toolbox
If a service is in the service repository, you must have all data and has to crawl to the diagram (not to be in the project folder browser)

10.3. Storage Requirements

The storage requirements, along with new natures, determine all the storage needs are identified during the interviews. Specifically, a storage requirement represents an important concept for the need to store information on your system.

Figure 40 shows the defined toolbox for storage requirements, also shared for new natures.

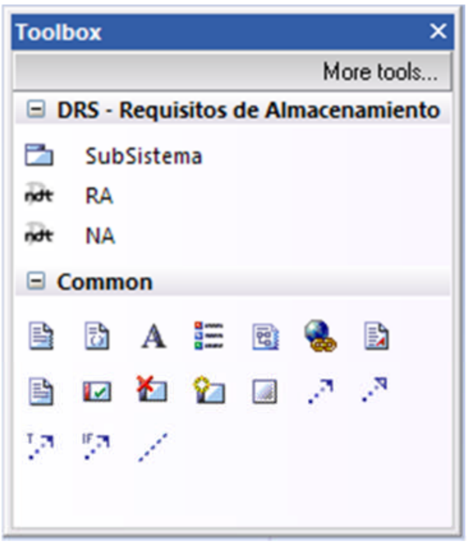

Figure 40. Toolbox for information requirements

To create a storage requirement, simply click on the toolbox in RA and drag the artifact to the diagram where you want to model. Figure 40 shows a storage requirement and property standards.

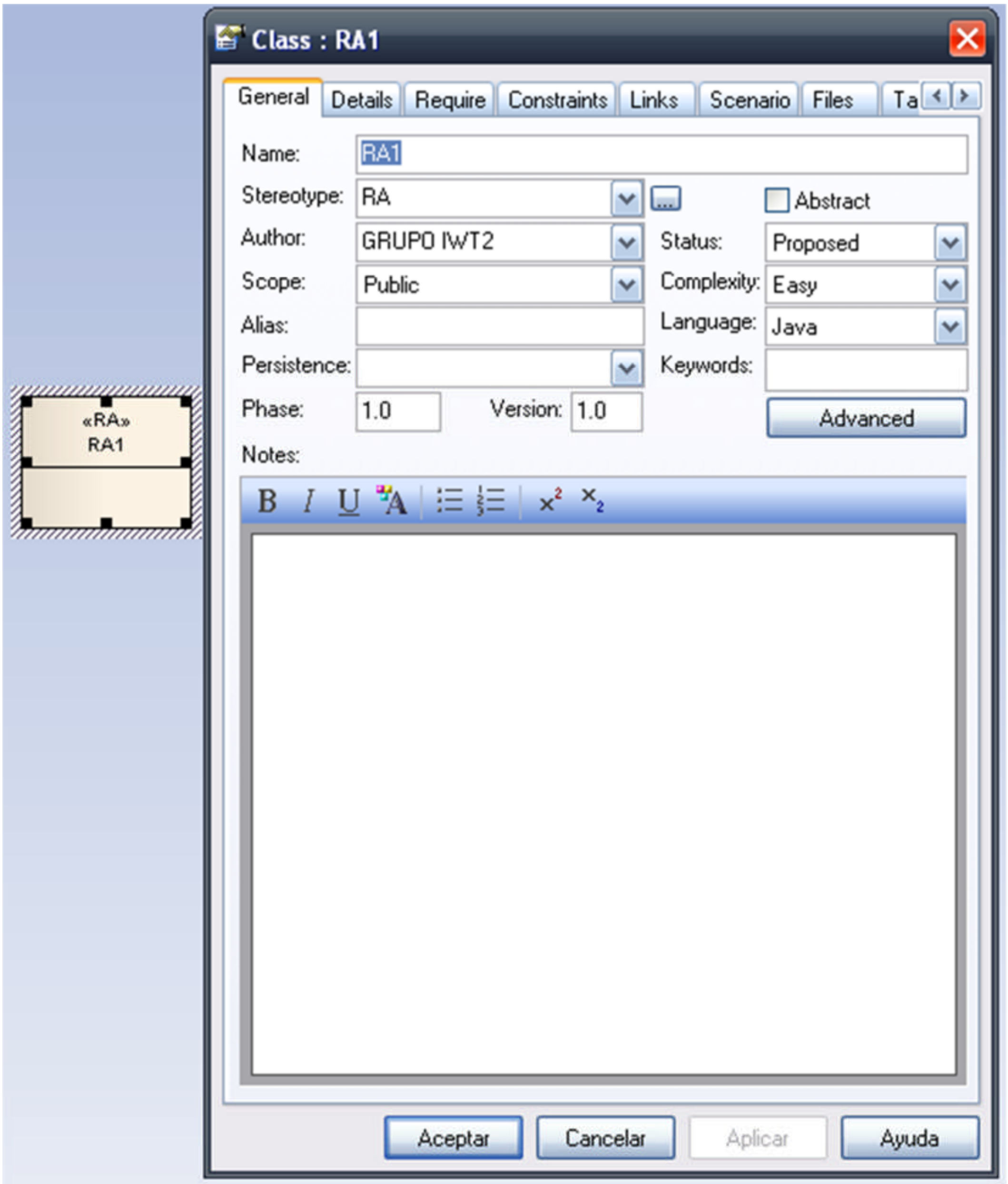

Figure 41. Standard properties of a storage requirement

In the RA artifact there are a number of fields that are mandatory for the description of the requirement for storage is correct. This series of fields are:

- **Name** (Name field): Each artifact should be classified with a code and a descriptive name. As shown in Table 14, the name must meet the following format RA-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there are a large number of storage requirements.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.

- **Language** (field Language): This field specifies the language in which it is the artifact. There might NDT Requisitos.
- **Specific data** (within the Details tab, Attributes button): The specific data set of descriptors representing a RA from the point of view of users and the client. How to add attributes is explained in paragraph 4.2 (Figure 15). Remember that a mandatory attribute must have a name, a type (see Figure 42), a description and cardinality.

There are also other fields are optional, but recommended that the form is completed for a better definition of the storage requirements. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Importance** (tagged value): This value indicates the importance that the system is the concept that models the artifact to the customer. Must be chosen one of the presets.
- **Urgency** (tagged value): This value indicates the urgency.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.
- **Stability** (tagged value): This value reflects the probability of labeling the appliance undergoes changes in its definition. Must be chosen one of the presets
- **Sources** (tagged value): This value reports the sources of the version of the appliance.
- **Interval time** (tagged value): This value indicates how long the artifact is effective. It can take two values, Present and Present and past.

There is define the type of specific data defined. Since this is a phase in the early life cycle, the type must be detailed in the NDT-Requisitos language. Next, Figure 42, shows the different types of data allowed in NDT-Profile for storage requirements.

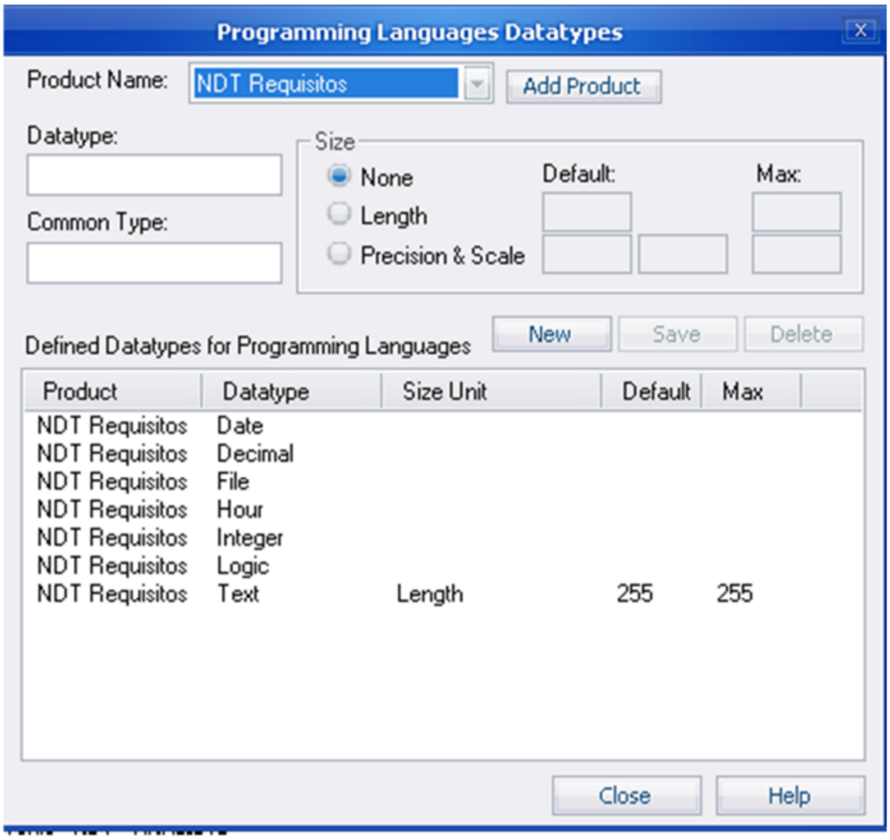

Figure 42. Data Types

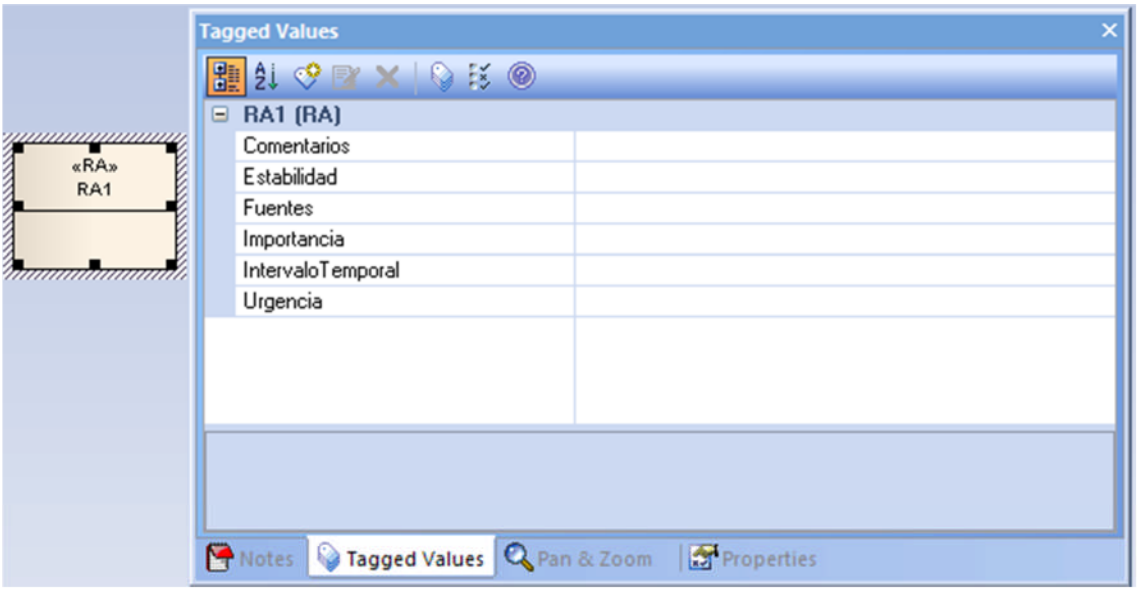

Figure 43. Tagged values of a storage requirement

Table 17. Rules of Storage Requirements (RA)

Storage Requirements Diagrams
You must have a diagram of type storage requirements
All RA and NA should be contained in the diagram
Storage Requirements Definition
You must have a diagram of type storage requirements
All RA should be contained in the diagram
The name of the artifact is RA-XX (X). Name
The artifact name must be unique, using the CamelCase notation and starting in uppercase
The description is filled
Language is NDT Requisitos
There is no other RA with the same number
The attribute name is not empty
The name of the attribute must begin in lower case, using the CamelCase notation
The type attribute is not empty
The type of the attribute belongs to the language of the artifact
The description attribute is filled

10.4. New natures

The new natures, together with storage requirements, determine all the storage needs are identified during the interviews. The new natures are different storage requirements that the NA defined information requirements that already exist in other systems, which will make use of, or are new domains that are defined specifically for the system that being modeled.

The properties of a new nature are the same as those of a storage requirement, so the figures seen above are also valid for the new natures. The tagged values of nature are shown in Figure 44.

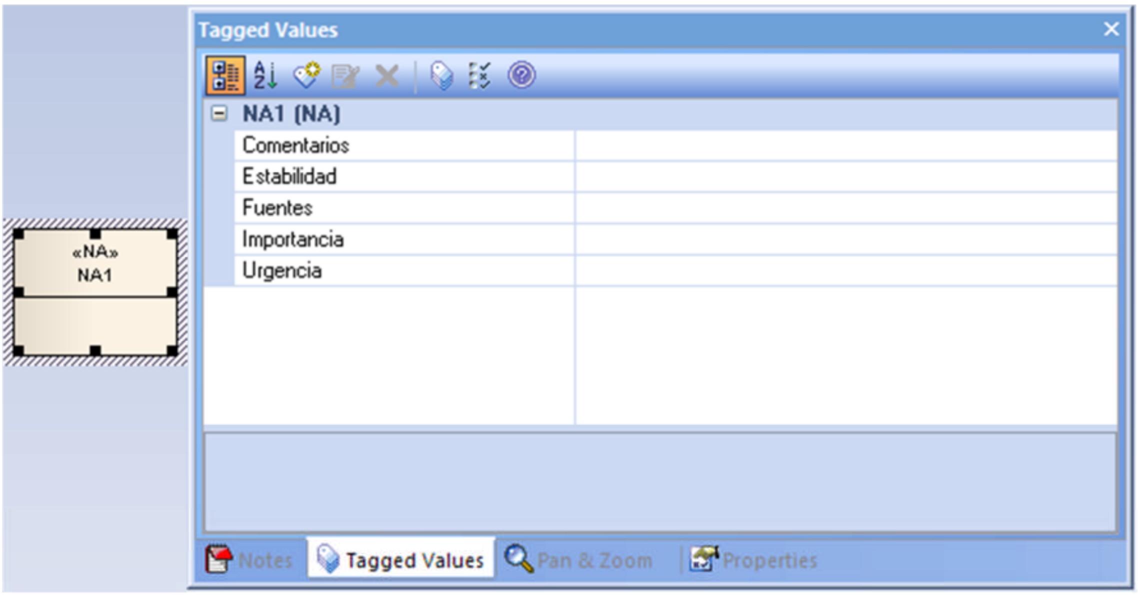

Figure 44. Tagged values of a new nature

Table 18. Natures New Rules (NA)

Defining New Natures
You must have a diagram of type New Natures
Every NA should be contained in the diagram
The name of the artifact is NA-XX (X). Name
The artifact name must be unique, using the CamelCase notation and starting in uppercase
The description is filled
There is no other with the same number NA
The attribute name is not empty
The name of the attribute must begin in lower case, using the CamelCase notation
The type attribute is not empty
The description attribute is filled
Language is NDT Requisitos
The type of the attribute belongs to the language of the artifact

10.5. Requirements of stakeholders

A basic actor is all actor that is individually identified on any criterion or point of view when interacting with the system. Experience says that to identify the basic agents that interact with a web system, there may be different criteria used. The implementation of each one of them results in the identification of a particular group of basics actors. Each basic actor corresponds to a role individualized interaction with the system software.

When considering this task does not have to worry about whether a single person or user can act as different actors, the actors must be considered in the application environment.

Figure 45 shows the model defined toolbox system players such as AC artifacts.

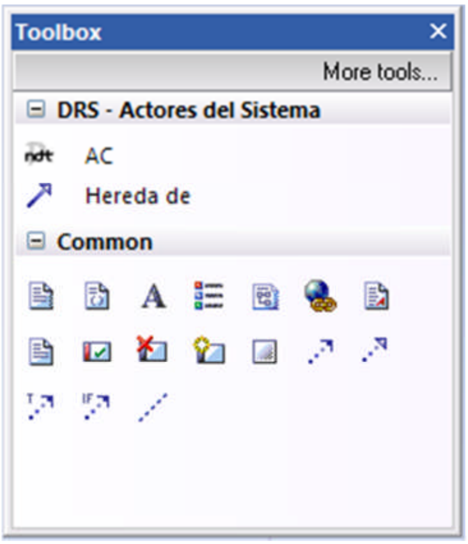

Figure 45. Actor's Toolbox

To create an actor, just click on the toolbox to the artifact AC and drag to the diagram where you want to model. In Figure 46 y Figure 47 shows an actor, property standards and values labeled.

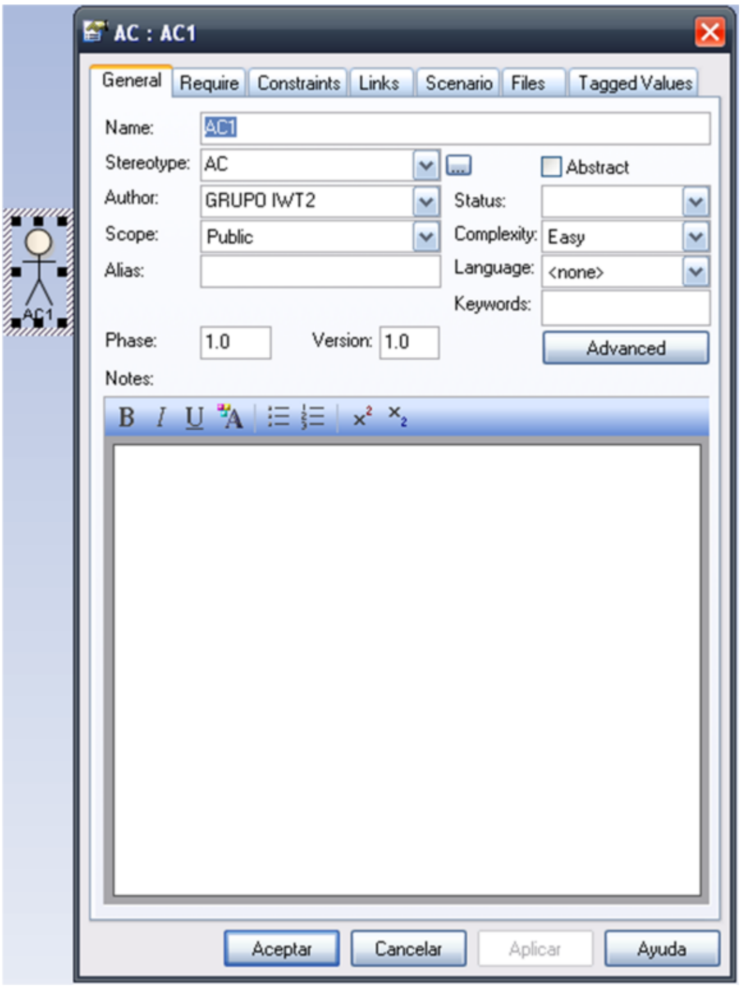

Figure 46. Standard properties of an actor

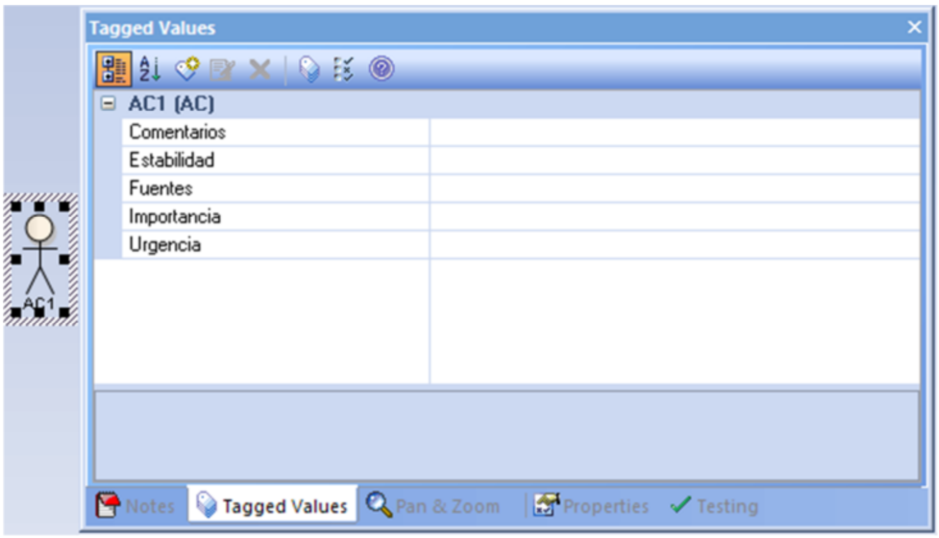

Figure 47. Tagged values of an actor

The AC artifact there are a number of fields that are mandatory for the description of the actor is considered correct. This series of fields are:

- **Name** (Name): Each actor should be classified with a code and a descriptive name. As shown in Table 14, the name must meet the following format AC-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there are a large number of players.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Language** (field Language): This field specifies the language in which it is the artifact. There might NDT Requisitos.

There are also other fields are optional, but recommended that the form is completed for a better definition of the actors. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Importance** (tagged value): This value indicates the importance that the system is the concept that models the artifact to the customer. Must be chosen one of the presets.
- **Urgency** (tagged value): This value indicates the urgency.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.
- **Stability** (tagged value): This value reflects the probability of labeling the appliance undergoes changes in its definition. Must be chosen one of the presets.
- **Sources** (tagged value): This value reports the sources of the version of the appliance.

The generalization between actors is permitted and is manifested by the connector "**Inherited from**" existing in the toolbox of Actors. To use this connector, simply click on it in the toolbox, click on the original actor and drag to the target actor in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Every actor should have at least one relationship with a use case or to extend to actors having at least one use case because if an actor does not define an entity outside the system who have no connection with the same, what is irrelevant.

Table 19. Rules of Actors (AC)

Actors Diagrams
You must have a type diagram Actors
All AC must be contained in the diagram
You may only link by link inherits from
Definition of Actors
The name of the artifact is AC-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
No other AC with the same number

Language is NDT Requisitos
The type of the attribute belongs to the language of the artifact

10.6. Functional requirements

A functional requirement defines the behavior of a specific functionality of the system. The systems also have to collect what is to be able to do with the information and the functional possibilities of it. The functional requirements answer the question of "*what can be done in the system?"*

To perform the capture and definition of functional NDT uses two techniques. On the one hand, proposes using use case diagrams [Jacobson 1995], which graphically represent the system functionality. However, the exclusive use of diagrams, may be too ambiguous in some cases. Therefore, NDT proposes these diagrams accompany textual information collected via patterns that clarify its meaning and what they represent.

Figure 48 se shows the toolbox model defined functional requirements.

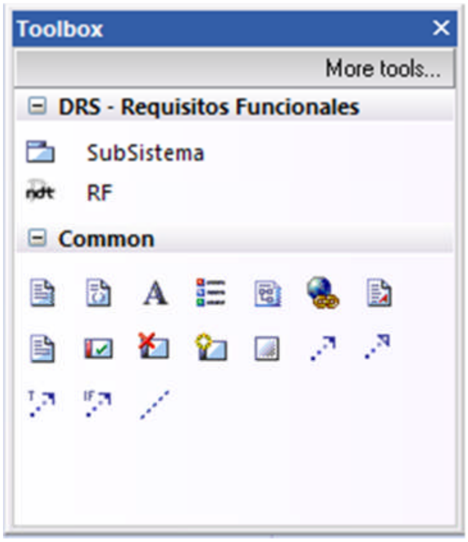

Figure 48. Toolbox for functional requirements

To create a functional requirement, simply click on the RF artifact in the toolbox and drag to the diagram where you want to model. For more information about working with use case diagrams, please read section 4.1.1 of this guide, which explains how to be these diagrams.

Figure 49 shows a functional requirement and its standard properties.

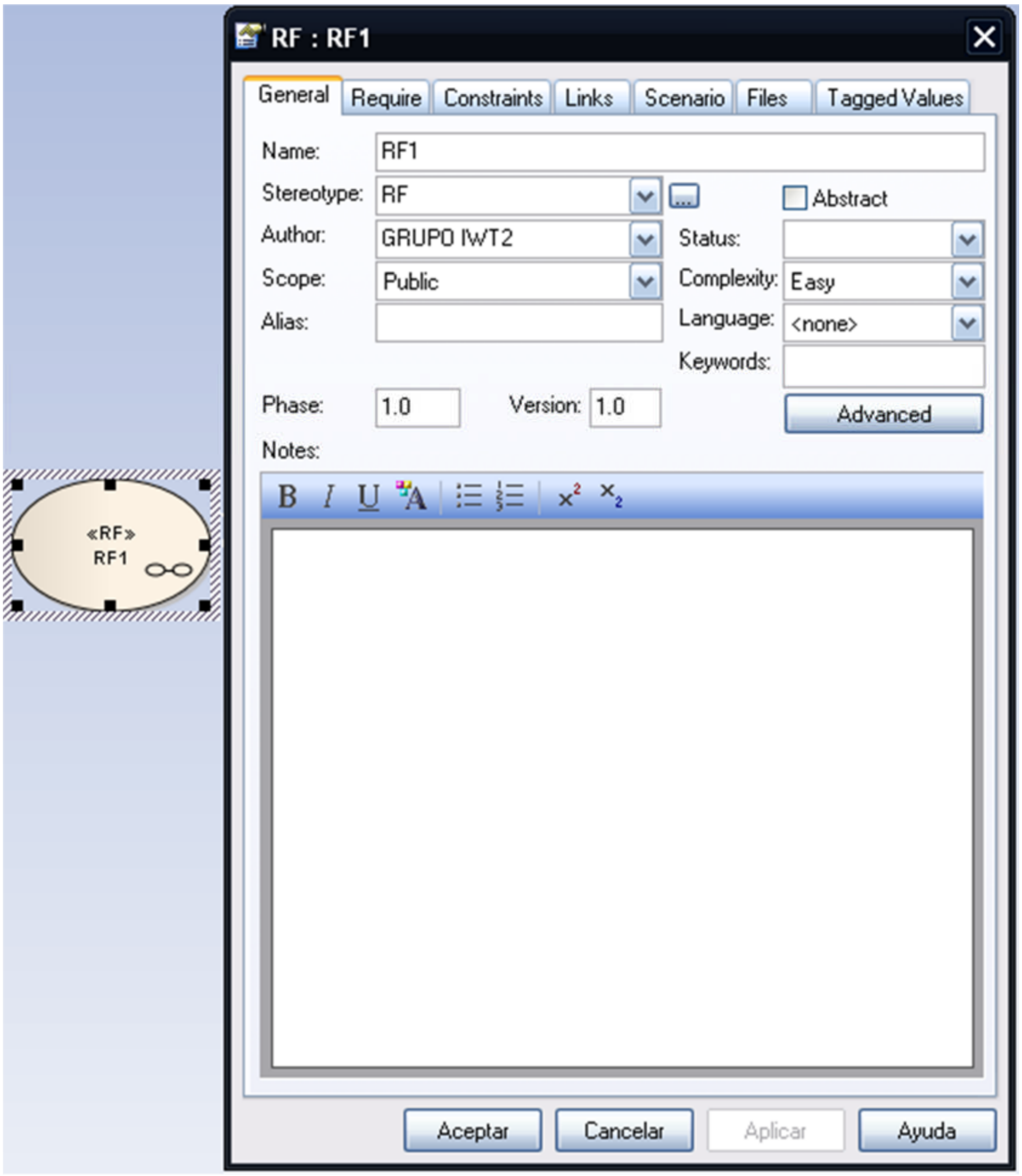

Figure 49. Standard properties of a functional requirement

The tagged values of a functional requirement are shown in Figure 50.

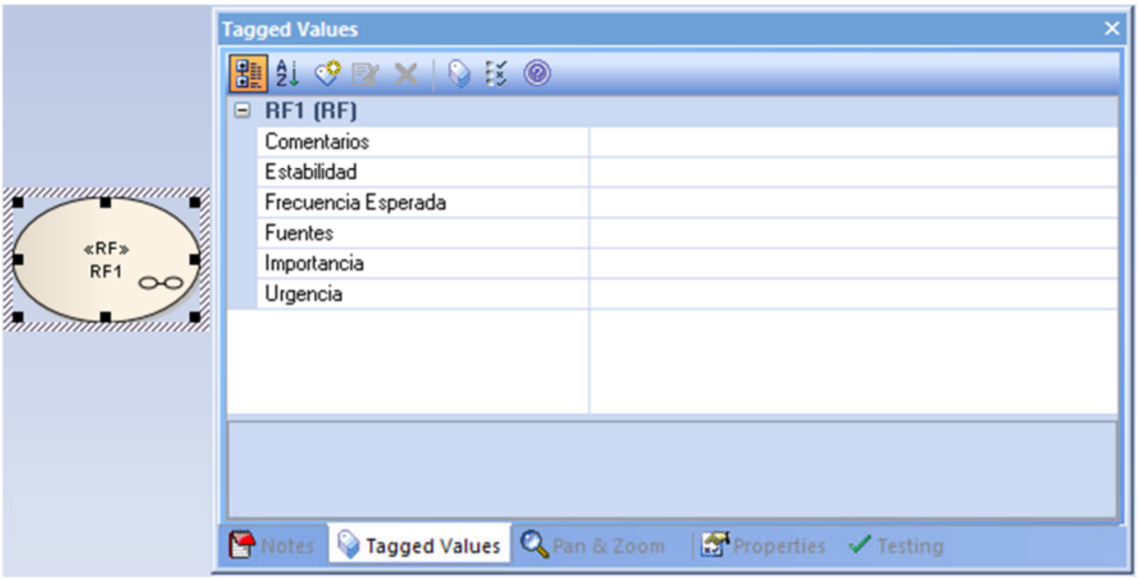

Figure 50. Tagged values of a functional requirement

In the RF artifact there are a number of fields that are mandatory for the description of functional requirement is considered correct. This series of fields are:

- **Name** (Name): Each functional requirement should be classified with a code and a descriptive name. As shown in Table 14, the name must meet the following format RF-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of functional requirements.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Constraints** (Constraints tab): In a functional requirement must be defined mandatory restrictions. This need to be detailed in the Constraints tab (as described in paragraph 4.2 of this guide, Figure 18) and the type may have are pre-condition and post-conditions. If the functionality does not have any restrictions, both as pre-condition post-condition will make clear that does not apply.
- **Diagram of activities or scenarios**: The functional requirements must necessarily be described by scenarios or an activity diagram. Both options are available, the first by Scenarios tab seen in Figure 20 and the second by double clicking on the RF. If you want the functionality that will be used to describe a complex activity diagrams mandatory, which are described further in Section 4.1.2.
- **Language** (field Language): This field specifies the language in which it is the artifact. There might NDT Requisitos.

There are also other fields are optional, but recommended that the form is completed for a better definition of functional requirements. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Complexity** (Complexity field): This field contains the complexity of the functional requirement. By default, the functional requirement is of low complexity, but it is recommended that there is at least one of high complexity.

- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Importance** (tagged value): This value indicates the importance that the system is the concept that models the artifact to the customer. Must be chosen one of the presets.
- **Urgency** (tagged value): This value indicates the urgency of the fulfillment of the concept that models the artifact. Must be chosen one of the presets.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.
- **Stability** (tagged value): This value reflects the probability of labeling the appliance undergoes changes in its definition. Must be chosen one of the presets.
- **Sources** (tagged value): This value reports the sources of the version of the appliance.
- **Frequency** expected (tagged value): this includes the frequency with which it is expected to be implemented the functionality that represents the artifact.

Table 20. Functional Requirements Rules (RF)

Functional Requirements Diagrams
Must have a diagram of type Functional Requirements
All RF must be contained in the diagram
Functional Requirements Definition
The name of the artifact is RF-XX (X).Name
The name of the RF must be in the infinitive
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
No other RF with the same number
The RF is an activity diagram, or scenarios
The scenarios should have name, description and type (normal or exception)
The scenarios must be numbered sequentially
The activity diagram must have a start node, end node and at least one activity
All objects in the activity diagram has to be named
The activities must have an outgoing link, and at least one inbound link
The starting node has to have an outgoing link
The end node must have at least one inbound link
Links from a decision node must have a value in their guards
If the RF has constraints, must have a name, description and a type (pre-condition or post-condition)
In the diagram, all have an associated RF AC or RF is part of another
Language is NDT Requirements
The type of the attribute belongs to the language of the artifact

10.7. Interaction requirements

10.7.1. Frases

The way in which models how the user wants to retrieve information is through the frases. A frase is an information retrieval approach in the system.

Figure 51 shows the toolbox model defined frases such as artifacts FR.

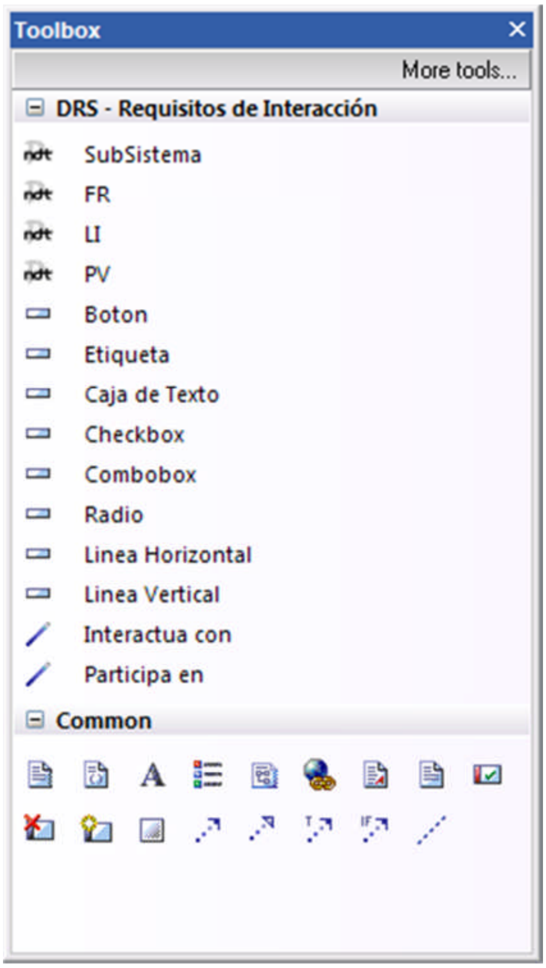

Figure 51. Toolbox for frases

To create a frase, simply click on the toolbox to the artifact FR and drag to the diagram where you want to model. To create a uicontrol on the FR element, simply click on the item and drag over the FR to which he wishes to associate. As storage requirements have attributes, such frases have uicontrol associated. Then, in Figure 52, explains how an element described uicontrol.

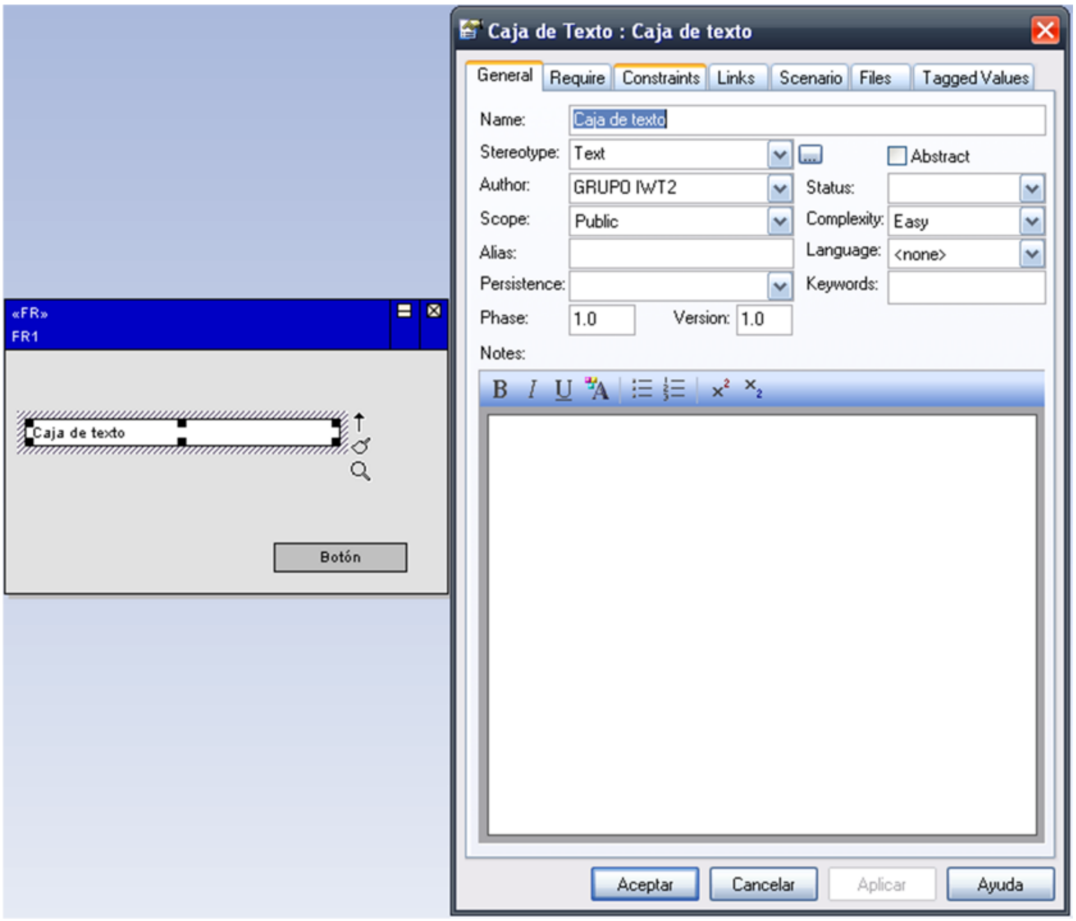

Figure 52. Uicontrol properties of an artifact

These artifacts are also uicontrol properties to be completed, such as the name (Name field), description (Field Notes) and the element to which reference (Alias field). In the event that the uicontrol is a button, it will in the Alias field the name of functional requirements to take the functionality. In the event that is of type text box and refer to some attribute of a storage requirement (or full-RA) also should be reflected in the Alias field. In addition, if an artifact uicontrol was only visible to one or more of the actors that is associated with the frase, we will use the pre-condition constraint (Constraints tab) for that.

Returning to the frases, the standard properties of this artifact are shown in Figure 53.

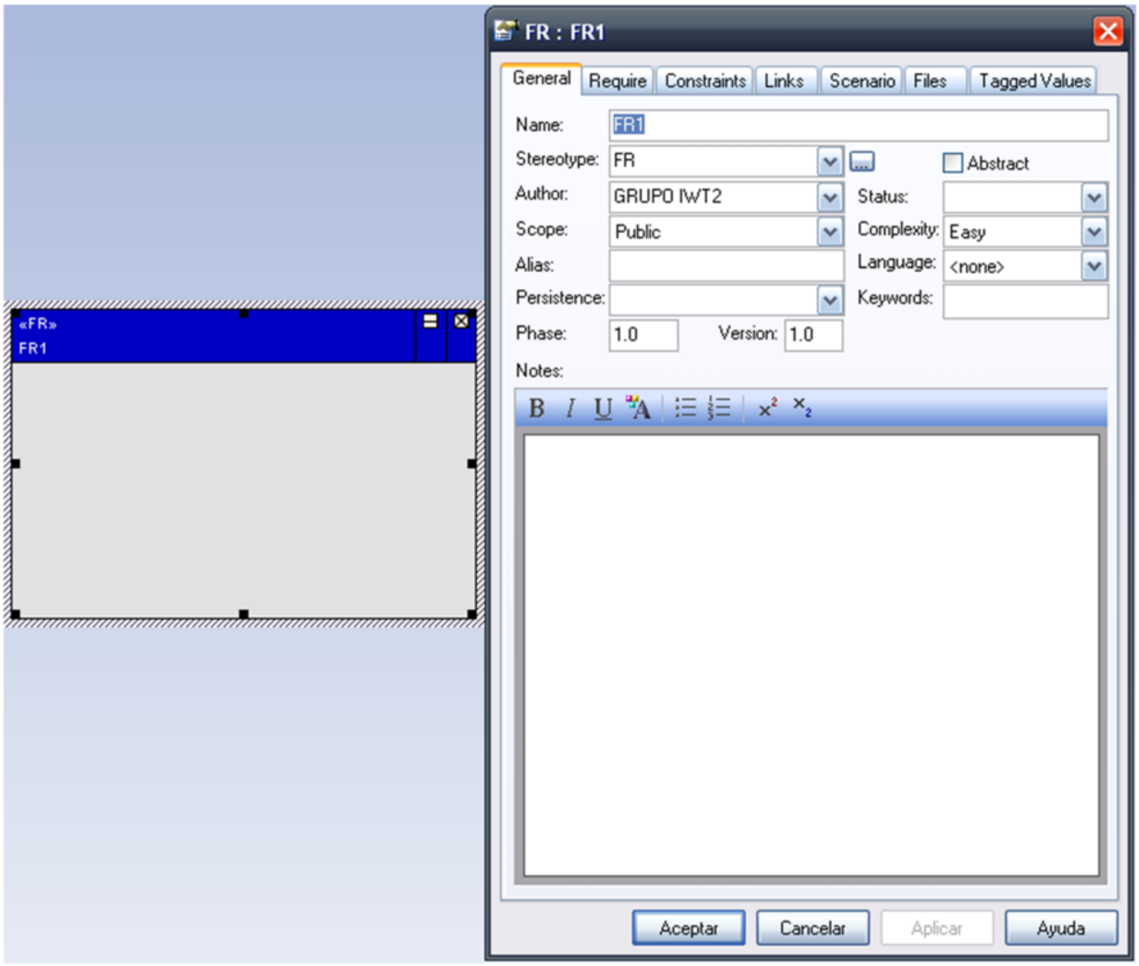

Figure 53. Standard properties of a frase

The tagged values are shown in Figure 54.

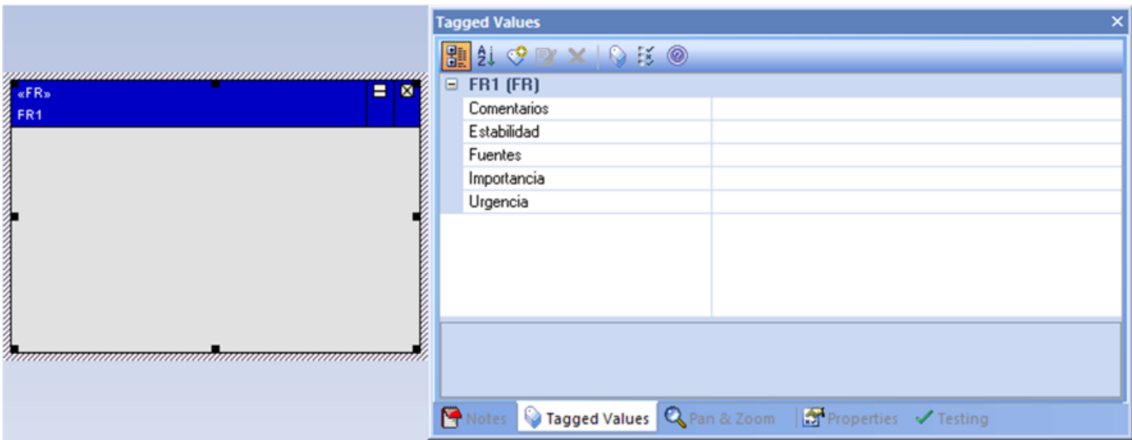

Figure 54. Tagged values of a frase

Required fields are the name, author, description, at least one button-type artifact and at least one artifact type box. In addition, each frase must have one or more associated actors, who are the ones who can perform these searches.

In the FR artifact there are a number of fields that are mandatory:

- **Name** (Name): Each frase should be classified with a code and a descriptive name. As shown in Table 14, the name must meet the following format FR-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of frases.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Artifact type button** (uicontrol): The frases should have at least one artifact type uicontrol button, indicating its functionality. This button has to be completed, mandatory, the name, alias and description. This information is extended with Figure 52.
- **Artifact type text box** (uicontrol): The frases should have at least one artifact type uicontrol text box, which indicate a data. This button has to be completed, mandatory, the name and description. If the information displayed comes from a reporting requirement, indicate in the alias. This information is extended with Figure 52.
- **Language** (field Language): This field specifies the language in which it is the artifact. There might NDT Requisitos.

There are also other fields are optional, but recommended that the form is completed for a better definition of the frases. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Importance** (tagged value): This value indicates the importance that the system is the concept that models the artifact to the customer. Must be chosen one of the presets.
- **Urgency** (tagged value): This value indicates the urgency of the fulfillment of the concept that models the artifact. Must be chosen one of the presets.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.
- **Stability** (tagged value): This value reflects the probability of labeling the appliance undergoes changes in its definition. Must be chosen one of the presets.
- **Sources** (tagged value): This value reports the sources of the version of the appliance.

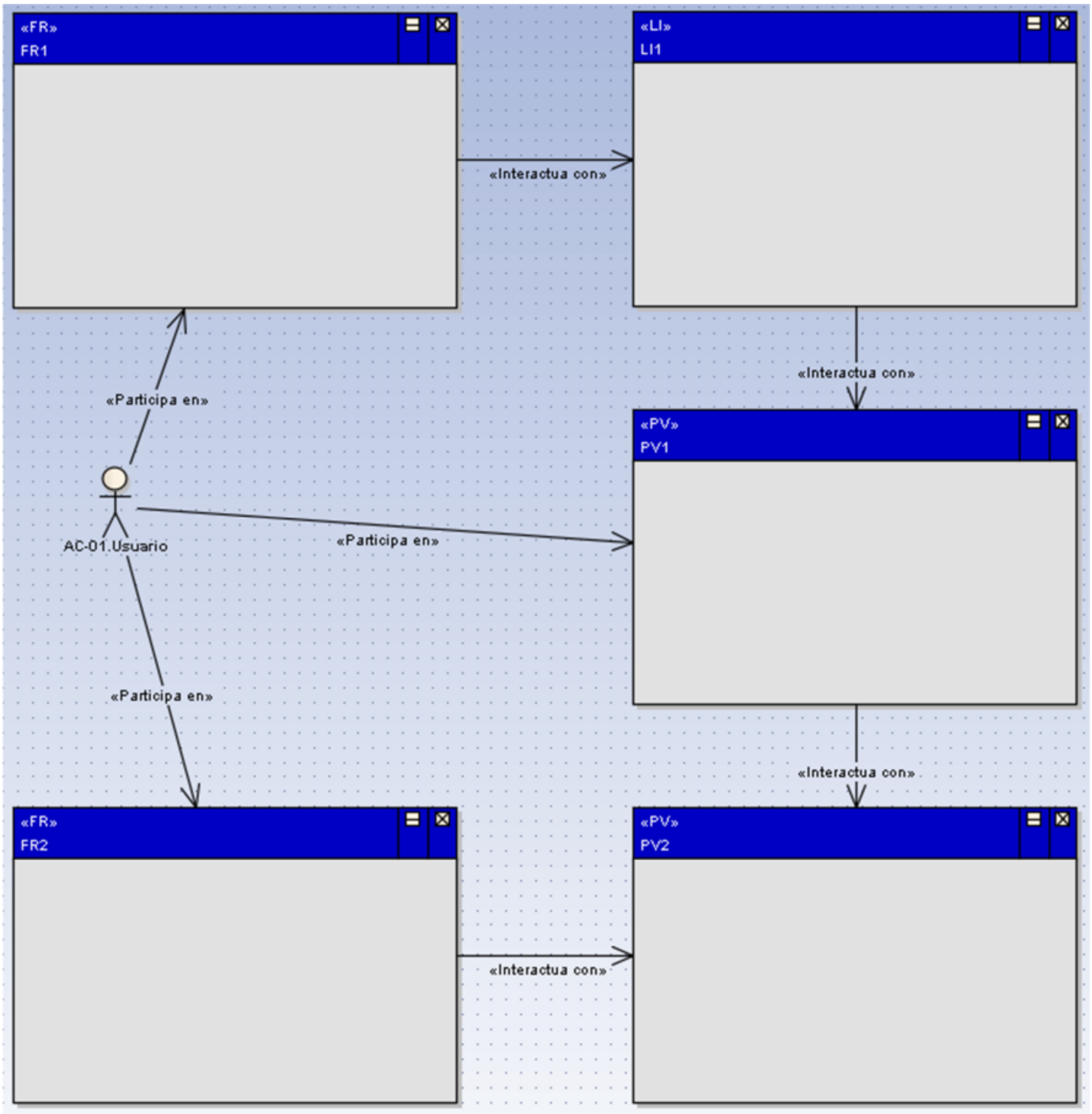

Figure 55. Connections interaction artifacts

Frases can be linked to any other artifact interaction (LI or PV) through the connector **"Interacts with",** which appears in the toolbox seen in Figure 51. This connector denotes a ship relationship between interaction artifacts, as shown in Figure 55. To use this connector, simply click on the toolbox to select, click on the source artifact interaction and drag to the destination artifact interaction in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

In addition to this connector, the frases are linking to the actors through the connector “**Join** "which appears in the toolbox, as seen in Figure 51. This connector denotes a relationship between interaction artifacts and actors, as shown in Figure 55. To use this connector, simply, first drag the actor to the diagram of prototype display, click on the toolbox to select, click and drag on the actor to display the prototype of the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Table 21. Frase Rules (FR)

Frase Diagrams
You must have a diagram of type of interaction requirements
All FR must be contained in the diagram
The FR can only link to other artifacts for interaction through the link interacts with
The FR can only bind to the AC using the link "Join"
Defining Frases
The name of the artifact is FR-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
There is no other with the same number FR
The attribute name is not empty
The attribute name must begin in lower case, using the CamelCase notation
The type attribute is not empty
The type of the attribute is an RA, or an attribute of this
The description attribute is filled
Language is NDT Requisitos
The type of the attribute belongs to the language of the artifact

10.7.2.Display Prototypes

A prototype display allows navigation to express the possibilities existing in the system, besides making reference to what data is displayed to each of the actors and what functionality is associated to each module of reporting. To get these prototypes, we should do a study of the objectives and interviews. Also, having defined the requirements and frases of the above tasks helps to identify them better.

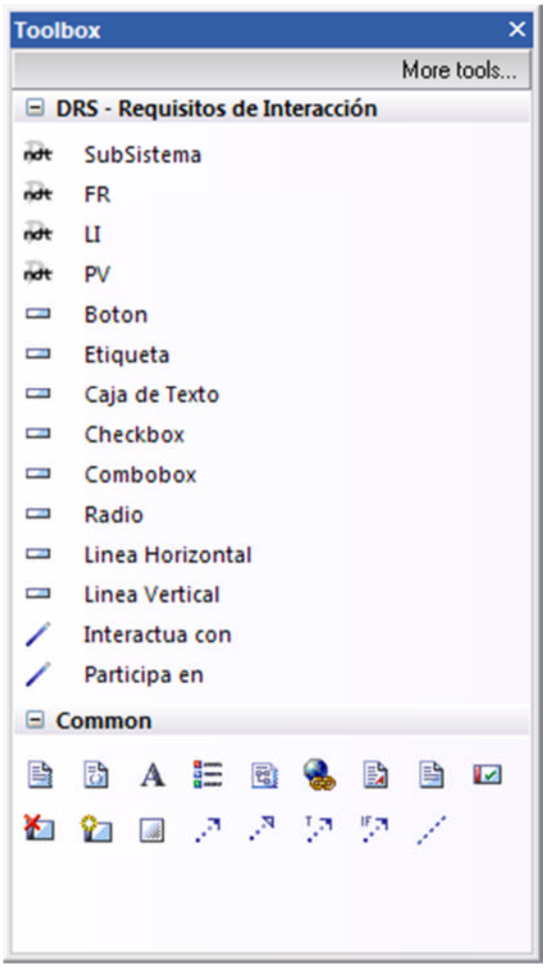

Figure 56. Toolbox to display prototypes

Figure 56 displays the toolbox for modeling and visualization prototype artifacts, PV. To create a prototype display, simply click on the toolbox in the artifact PV diagram and drag to where you want to model. To create an item uicontrol on the PV, simply click on the item and drag over PV to which he wishes to associate. The way to describe an item uicontrol has already been detailed in the previous section, in particular in Figure 52.

The following standards are observed properties of a prototype visualization, in Figure 57.

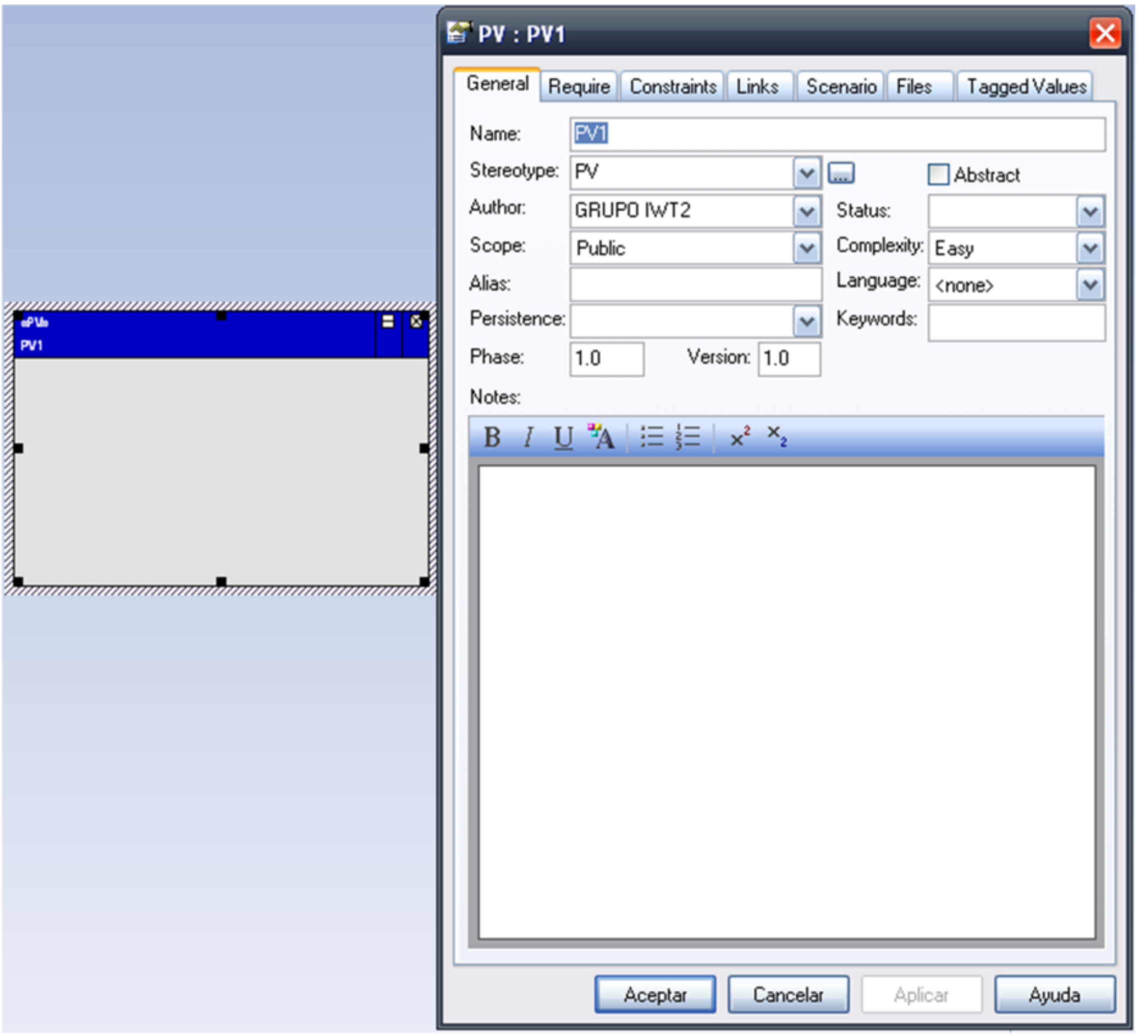

Figure 57. Properties of a prototype display

The standard properties of a prototype display are the same as those of a frase, as shown above. Its tagged values are the same as those of a target, also shown above.

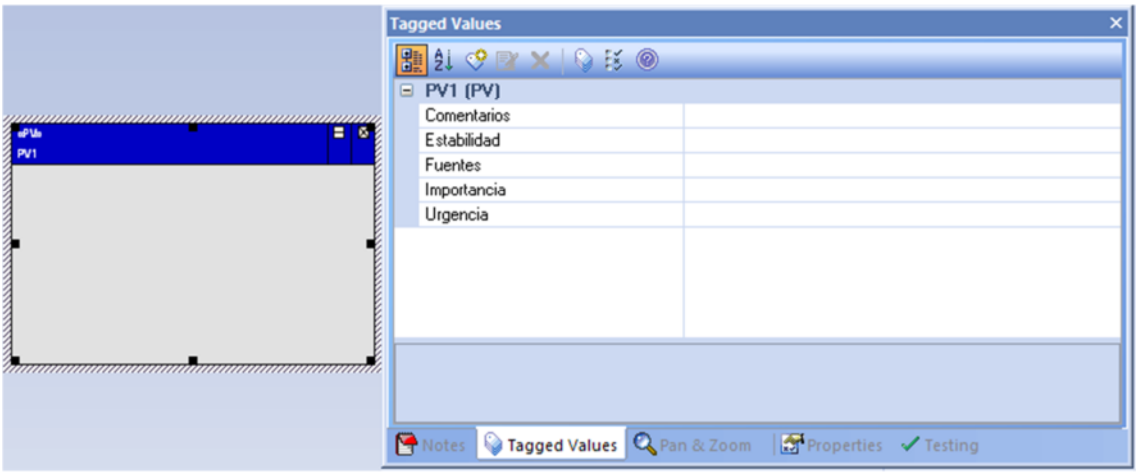

Figure 58. Tagged values of a prototype display

In the prototype view, mandatory and optional data are also the same as in the frases, so you will not be described again.

The prototype display can be linked with any other artifact with each other through interaction and **Interacts With** the connector and actors through the **Join** link, which appears in the toolbox seen in Figure 56. These connectors are explained in paragraph 10.7.1 and shown in Figure 55.

Table 22. Display Prototype Rules (PV)

Diagrams Display Prototype
You must have a diagram of type of interaction requirements
All PV should be contained in the diagram
The PV can only link to other artifacts for interaction through the link interacts with
The PV can only bind to the AC by binding Join
Prototype Definition Display
The name of the artifact is PV-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
No other PV with the same number
The attribute name is not empty
The attribute name must begin in lower case, using the CamelCase notation
The type attribute is not empty
The type of the attribute is an RA, or an attribute of this
The description attribute is filled
The name of the operations must be the name of an RF
The description of the operation is filled
Language is NDT Requisitos
The type of the attribute belongs to the language of the artifact

10.7.3.Listings

A list model the result of a search, expressing those fields that are necessary to show for later viewing access to the corresponding prototype.

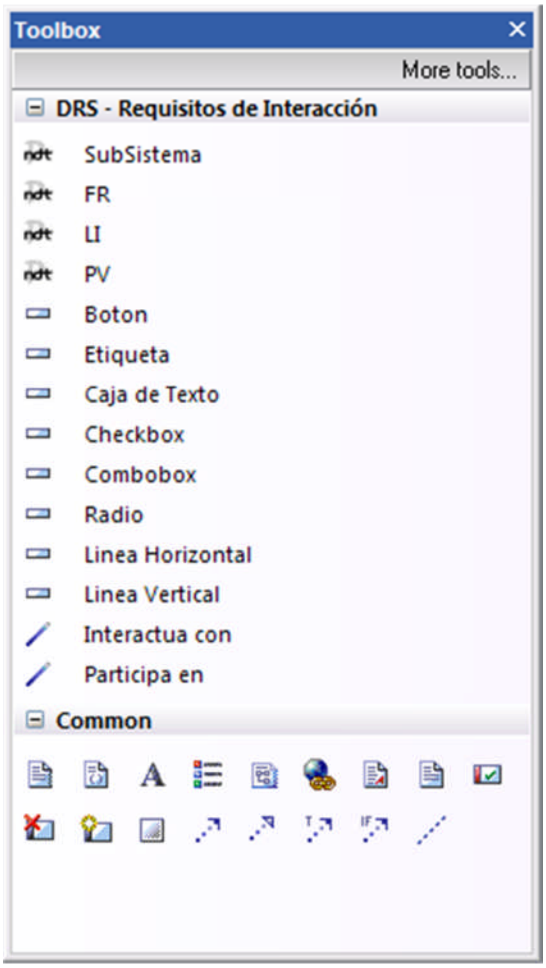

Figure 59. Toolbox for listings

Figure 56 displays the toolbox for the modeling of the artifacts listed as LI. To create a listing, simply click on the toolbox in the artifact LI and drag to the diagram where you want to model. To create a uicontrol on the LI element, simply click on the item and drag over the LI to which he wishes to associate. The way to describe an item uicontrol has already been detailed in the previous section, in particular in Figure 52.

The following standards are observed properties of a list, in Figure 57.

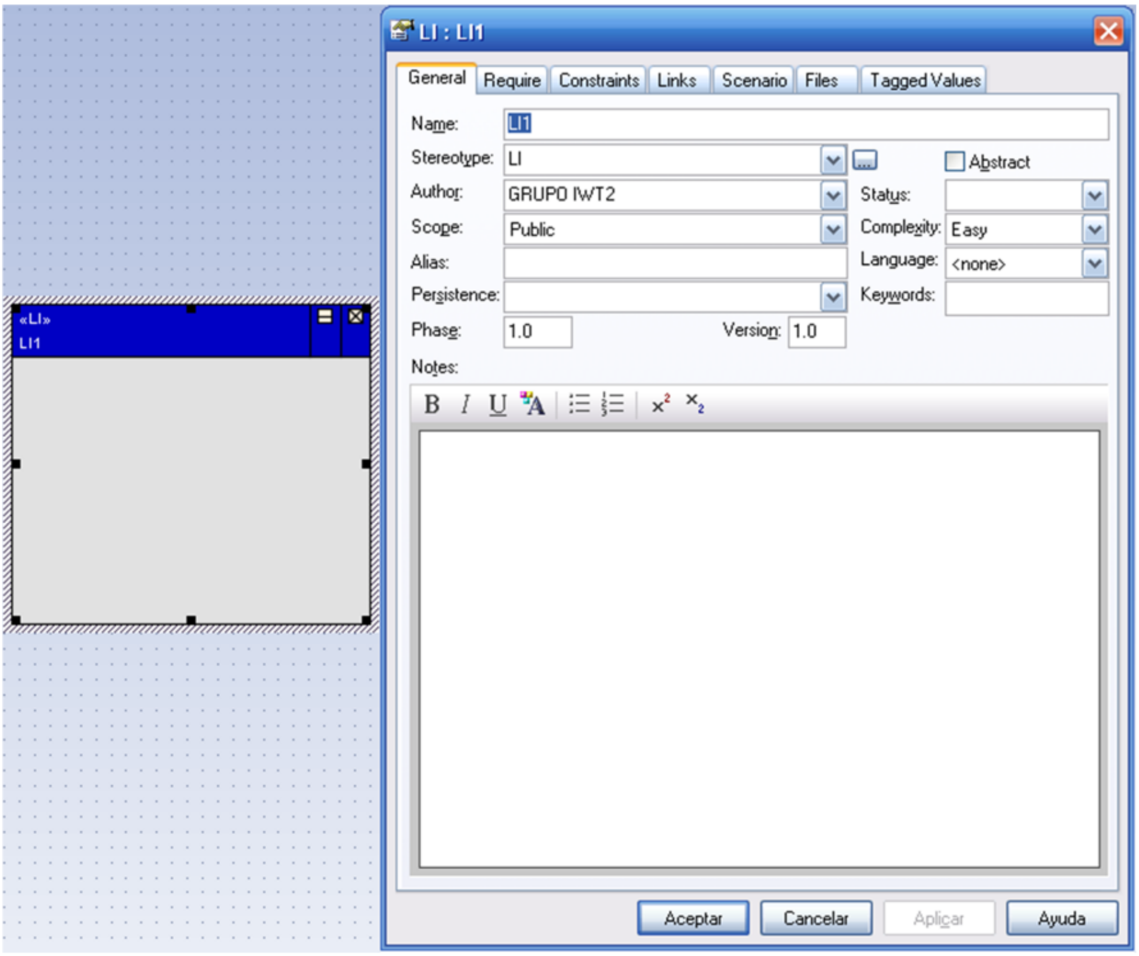

Figure 60. Properties of a list

The properties of a listing standards are the same as those of a frase, as shown above. Its tagged values are the same as those of a target, also shown above.

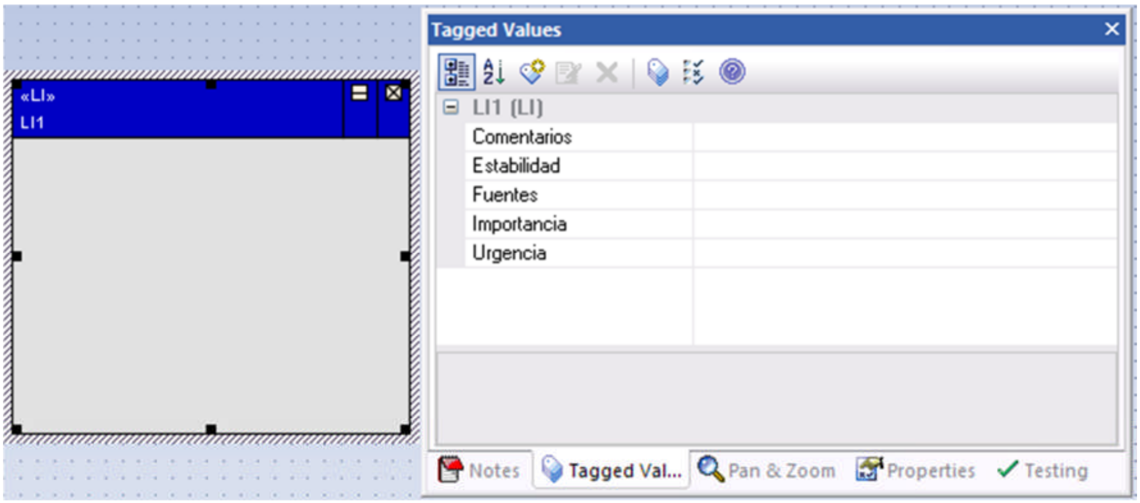

Figure 61. Tagged values of a list

In the listings, the mandatory and optional features are also the same as in the frases, so you will not be described again.

The listings can be linked with any other artifact of interaction through the connector "**Interact with**" and with actors using the link "**Join**" which appear in the toolbox seen in Figure 59. These connectors are explained in paragraph 10.7.1 and shown in Figure 55

Table 23. Listing Rules (LI)

Diagrams Listings
You must have a diagram of type of interaction requirements
All LI should be contained in diagram
The LI can only be linked to others through the link interacts with
Definition Lists
The name of the artifact is LI-XX (X) Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
There is no other LI with the same number
The attribute name is not empty
The attribute name must begin in lower case, using the CamelCase notation
The type attribute is not empty
The type of the attribute is an RA or an attribute of this
The description attribute is filled
The name of the operations must be the name of an RF
The description of the operation is filled
Language is NDT Requisitos
The type of the attribute belongs to the language of the artifact

10.8. Non-functional requirements

Non-functional requirements are requirements do not cover in any of the foregoing. A nonfunctional requirement is a requirement that specifies criteria that can be used to judge the operation of a system rather than specific behaviors.

Figure 62 shows the toolbox model defined as non-functional requirements such artifacts RNF.

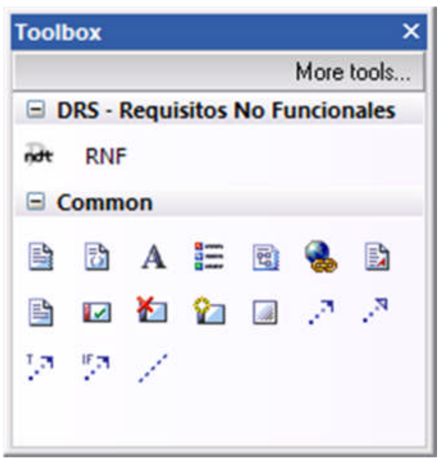

Figure 62. Toolbox for non-functional requirements

To create a nonfunctional requirement, simply click on the toolbox in the artifact RNF and drag to the diagram where you want to model.

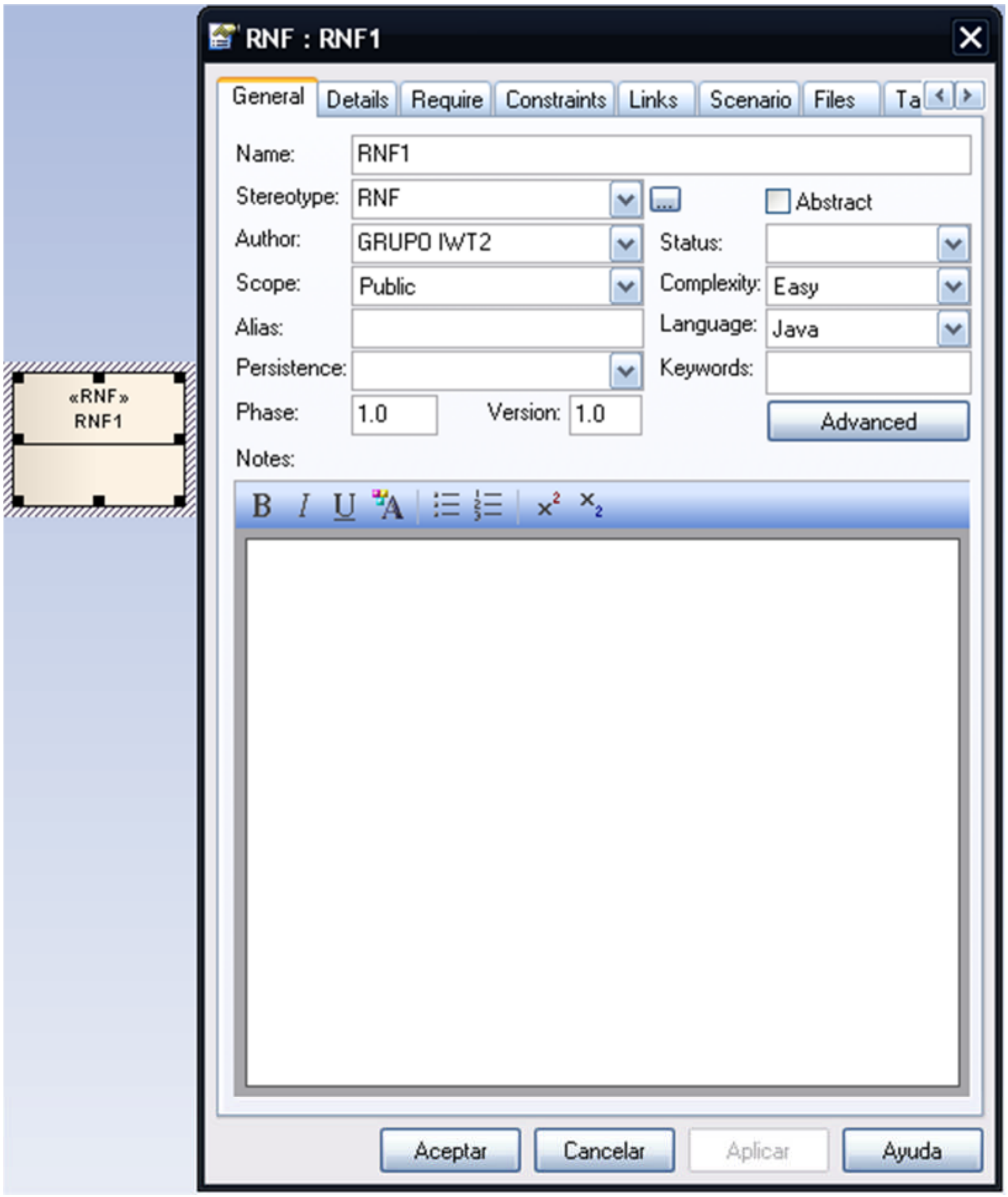

Figure 63. Properties of a non-functional requirement

In the artifact RNF there are a number of fields that are mandatory for the description of non-functional requirement is considered correct. This series of fields are:

- **Name** (Name): Every non-functional requirement to be classified with a code and a descriptive name. As shown in Table 14, the name must meet the following format RNF-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there are a large number of non-functional requirements.

- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Language** (field Language): This field specifies the language in which it is the artifact. There might NDT Requisitos.

There are also other fields are optional, but recommended that the form is completed for a better definition of non-functional requirements. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Importance** (tagged value): This value indicates the importance that the system is the concept that models the artifact to the customer. Must be chosen one of the presets.
- **Urgency** (tagged value): This value indicates the urgency of the fulfillment of the concept that models the artifact. Must be chosen one of the presets.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.
- **Stability** (tagged value): This value reflects the probability of labeling the appliance undergoes changes in its definition. Must be chosen one of the presets.
- **Sources** (tagged value): This value reports the sources of the version of the appliance.

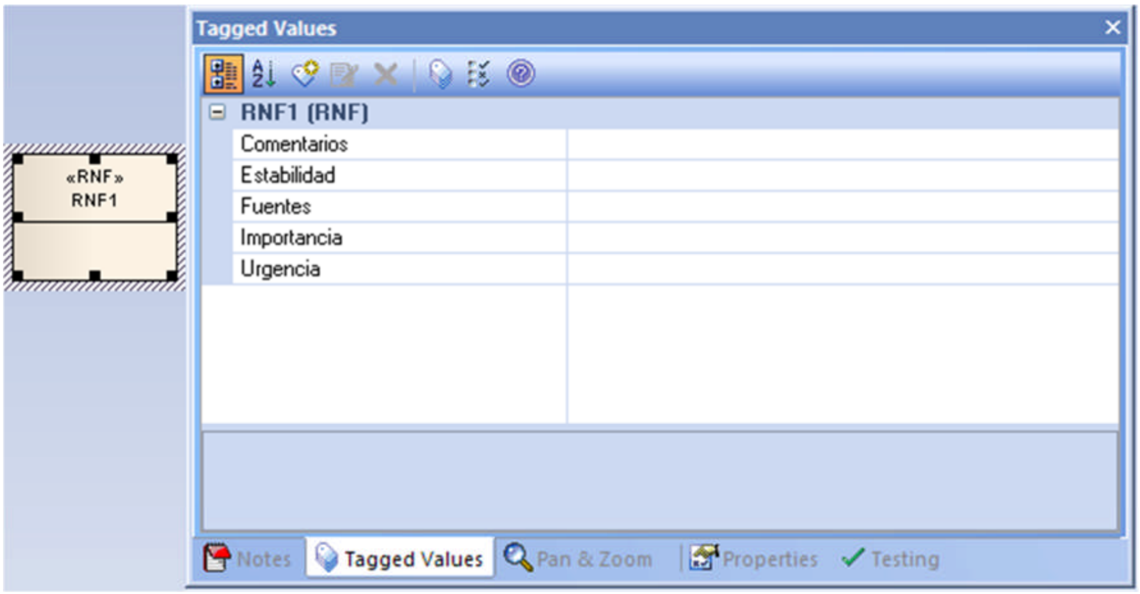

Figure 64. Tagged values of a non-functional requirement

Table 24. Rules of nonfunctional requirements (RNF)

Diagrams nonfunctional requirements
Diagram type has a non-functional requirements
All content must be RNF in the diagram
No Functional Requirements Definition
The name of the artifact is RNF-XX (X). Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
There is no other with the same number RNF
All are within the diagram RNF
Language is NDT Requisitos
The type of the attribute belongs to the language of the artifact

11. System Analysis

The analysis phase will contain the resulting products to analyze, define and structure the requirements in the previous phase.

The codes for each artifact are shown in Table 25.

Table 25. Nomenclature Analysis

Analysis	Name
Services	Service-XX. Name
Classes	CL-XX.Name
Natures New Classes	CLn-XX.Name
Process Classes	CP-XX.Name
Actor Studio	AE-XX.Name
Nodes	NO-XX.Name
Queries	QU-XX.Name
Indexes	IN-XX.Name
Menus	ME-XX.Name

The document structure analysis system and its relation to the structure of packages NDT profile shown in Figure 65.

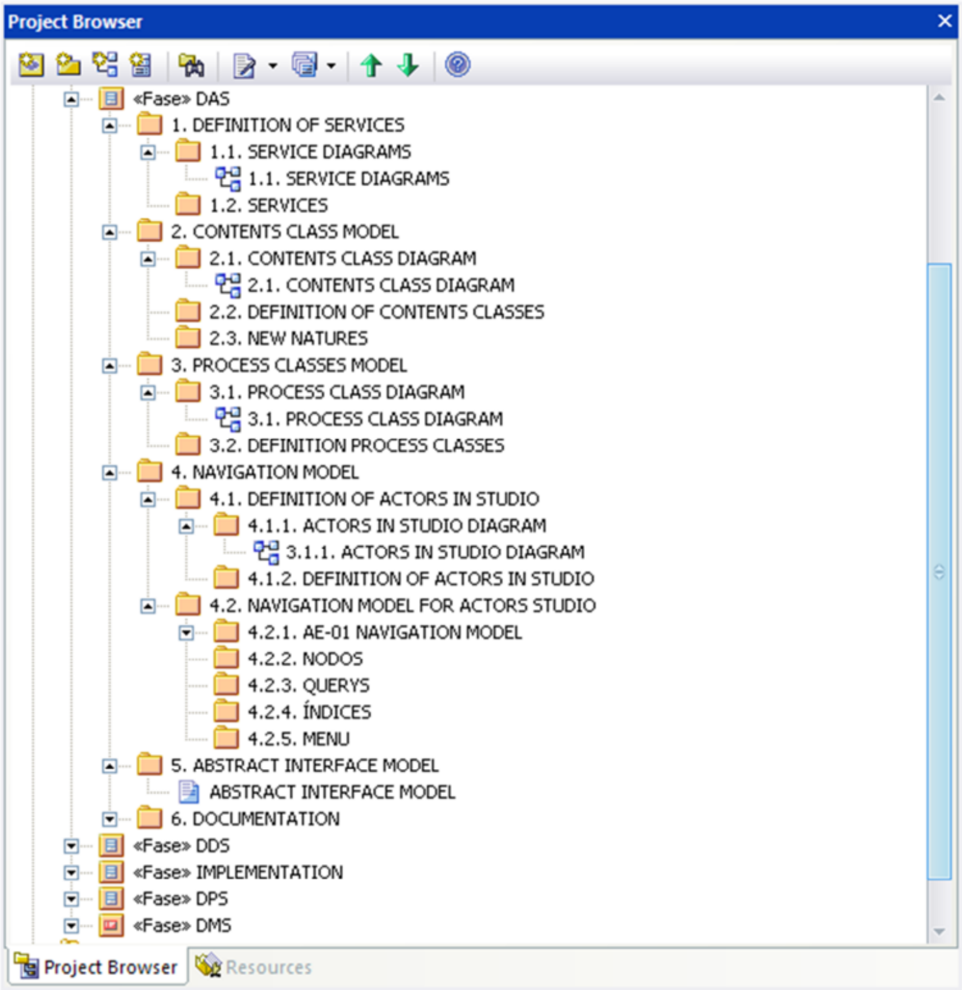

Figure 65. DAS Structure

Corresponding tools for the definition of the artifacts of analysis are shown in Figure 66.

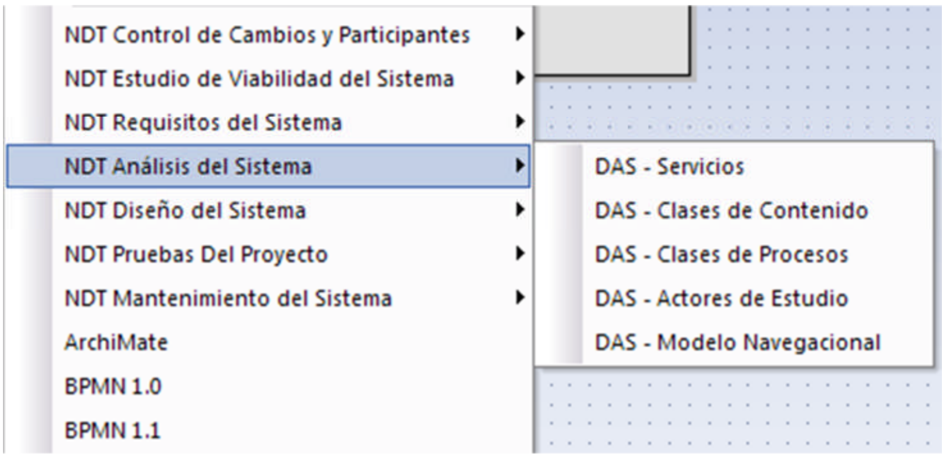

Figure 66. Toolboxes analysis

11.1. Service Definition

Having identified the services at the phase of "System Requirements" will be two types of services. Those are of the repository of the client, already defined and designed and the services we have identified for our project. It is the latter, the services that we define in this phase. The definition of services is to gather the most relevant information associated with it.

Figure 67 shows the toolbox set to artifacts such Services.

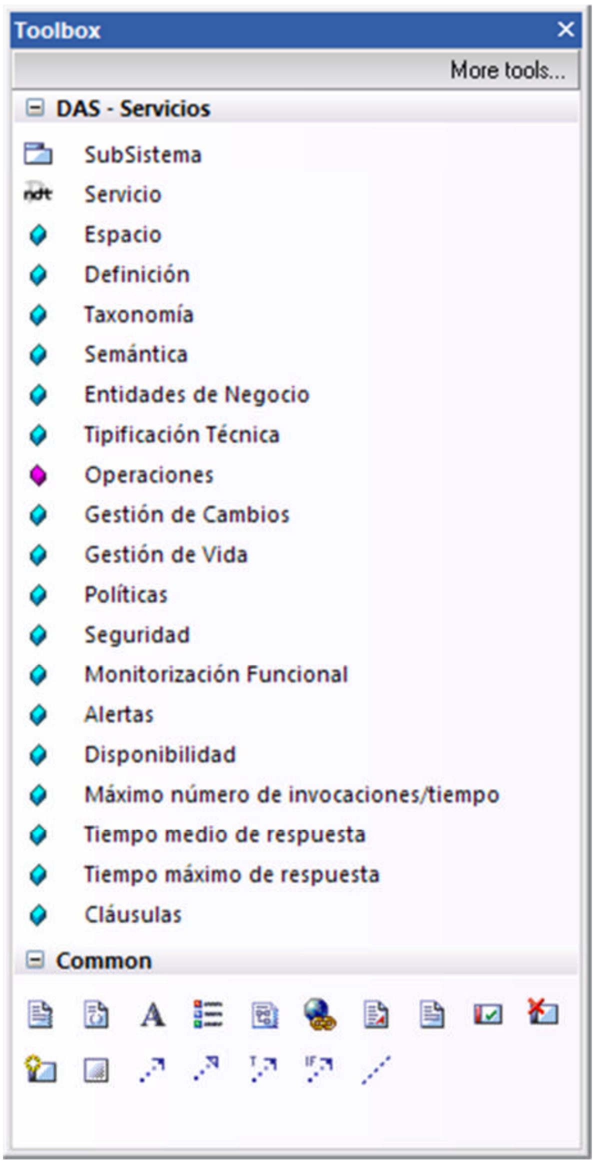

Figure 67. Analysis Services Toolbox

Figure 68 is observed standard properties of the Services.

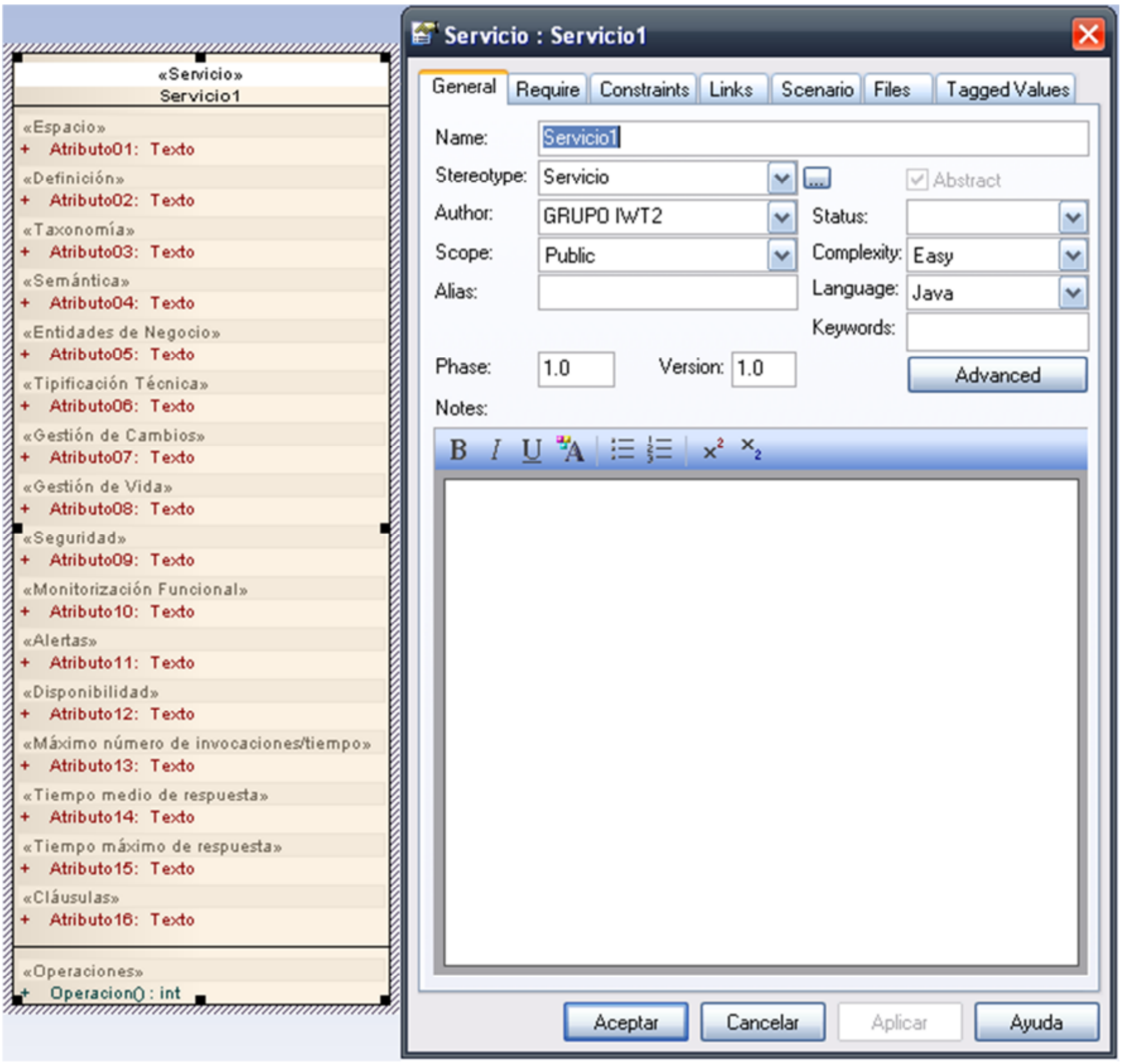

Figure 68. Properties of a Service Standards

The artifact analysis services there are a number of fields that are mandatory for the service description to be correct. In the case of service attributes, they are added to the artifact from the Toolbox DAS-Services, seen in Figure 67. To do this, click on the toolbox is the attribute to add and drag the desired service.

These mandatory fields are:

- **Name** (Name): Each goal should be classified with a code and a descriptive name. As shown in Table 25, the name must meet the following format Service -XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a high number of services.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the service.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the service that is being treated.
- **Language** (field Language): This field specifies the programming language in which it is the artifact.
- **Space** (attribute): general taxonomic grouping of service. Originates namespaces that begin with the generic branch of taxonomy, and concludes with the most specific. Is best

represented by the names of each of the nodes between them (eg junta-andalucia.ccul.empleados ...). Information provided by membership in a repository.

- **Definition** (attribute): Full description of the service. Must be indexable by search artifacts.
- **Taxonomy** (attribute) node of the taxonomy of services to which it belongs. Information provided by membership in a repository.
- **Semantics** (attribute): List of keywords to the evaluations made by search artifacts. Adjectives and nouns are often intimately related to the functionality of the service.
- **Business Entities** (attribute): List of business entities related to the service. Be taken in this field the main institutions associated with the business (in case of using auxiliary entities that fall outside the scope of the functionality will not be included).
- **Typing technique** (attribute): Any service must be established technically. Here is the tree of typing services:

Table 26: Technique Typing Service

Raíz	Tipo	Subtipo	Descripción
TYPOLOGY	PROCESS	COORDINATION	Coordinates a sequence or flow formed by several processes.
		PROCESS	Is responsible for performing logic functions of processes (decision making, timing, etc.)..
	FUNCTIONAL	BUSINESS	Encapsulates business logic atomized.
		PROXY	Provides a remote business functionality implemented by distributed applications.
		WRAPPER	Encapsulates business functionality provided by legacy applications.
	TECHNOLOGICAL	CONTROL	Provides horizontal sharing, security, session, etc. (Dispatcher, façades ...).
		UTILITY	Provides functionality outside the business (calculations, helpers ...).

- **Change management** (attribute) data are versions, branches of development, obsolete features, compatibility, date related, responsible for the changes, reasons for them, history of previous versions, etc.
- **Management of life** (attribute): Data related to the state it was found service in certain contexts (development, testing, integration, production). For example, maintains that the active versions of a service in development are x and x 1, while production is x-3. It also maintains data dependencies (should be minimal in the case of services) which may be by the membership of service to a BPM.
- **Governance** (attribute): Specification of the policies implemented by the service to be defined in the SOA governance model.
- **Security** (attribute): Specification of the security restrictions that apply to the service to be defined in the SOA governance model.
- **Monitoring function** (attribute): Functional monitoring indicators that apply to the service to be defined in the SOA governance model.
- **Alerts** (attribute): Levels and types of warning based on indicators defined above.
- **Availability** (attribute): Indicates the slot or slots and days on which the service can be used. For example, from 8:00 to 20:00 from Monday to Friday except public holidays or 24hx7días.
- **Maximum number of invocations / time** (attribute): Indicates the maximum number of invocations to a customer may perform in a defined time unit, for example, 10 invocation per second.
- **Maximum response time** (attribute): Maximum message processing.

- **Clauses** (attribute): Other clauses that may apply to the particular hiring and where appropriate, non-compliance.
- **Operations** (in Details tab, button Operations): The services must have at least one operation. How to add operations are explained in paragraph 4.2 (Figure 17). Remember that a transaction of a service has to be necessarily a name, parameters, return value and description.

There are also other fields are optional, but recommended that the form is completed for a better definition of services. These are:

- **Status** (Status Field): This field contains the situation where the service is in its development process. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the service. This field is meaningless when it comes to large systems where version management is a critical task.

Table 27. Analysis Services Rules

Definition of Services
You must have a type diagram Services
The name and description must be filled
Each service that is not the repository of services has to be at least the attributes that appear in the toolbox
If a service is in the service repository, you must have all the data and have to crawl into this diagram (not to be in the project folder browser)

The attributes must have completed compulsory, at least the name, type and stereotype. Operations, must have filled the name, parameters, return value and description.

11.2. Content classes

The content model class represents the class diagram derived from the information storage requirements. Allows for modeling how to structure the information handled in the system.

Figure 48 shows the toolbox model defined content classes such as artifacts of CL, in the case of classes that come from the RA requirements and CLn type artifacts, for classes that come from the NA of requirements.

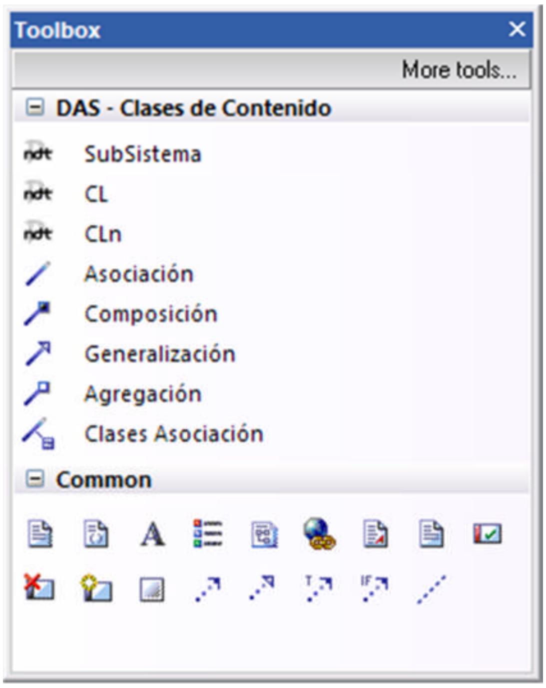

Figure 69. Content classes Toolbox

To create a class, simply click on the toolbox to the artifact CLn or CL and drag to the diagram where you want to model. To see the standard properties of a class, you will double click on the CL, showing the screen of Figure 70.

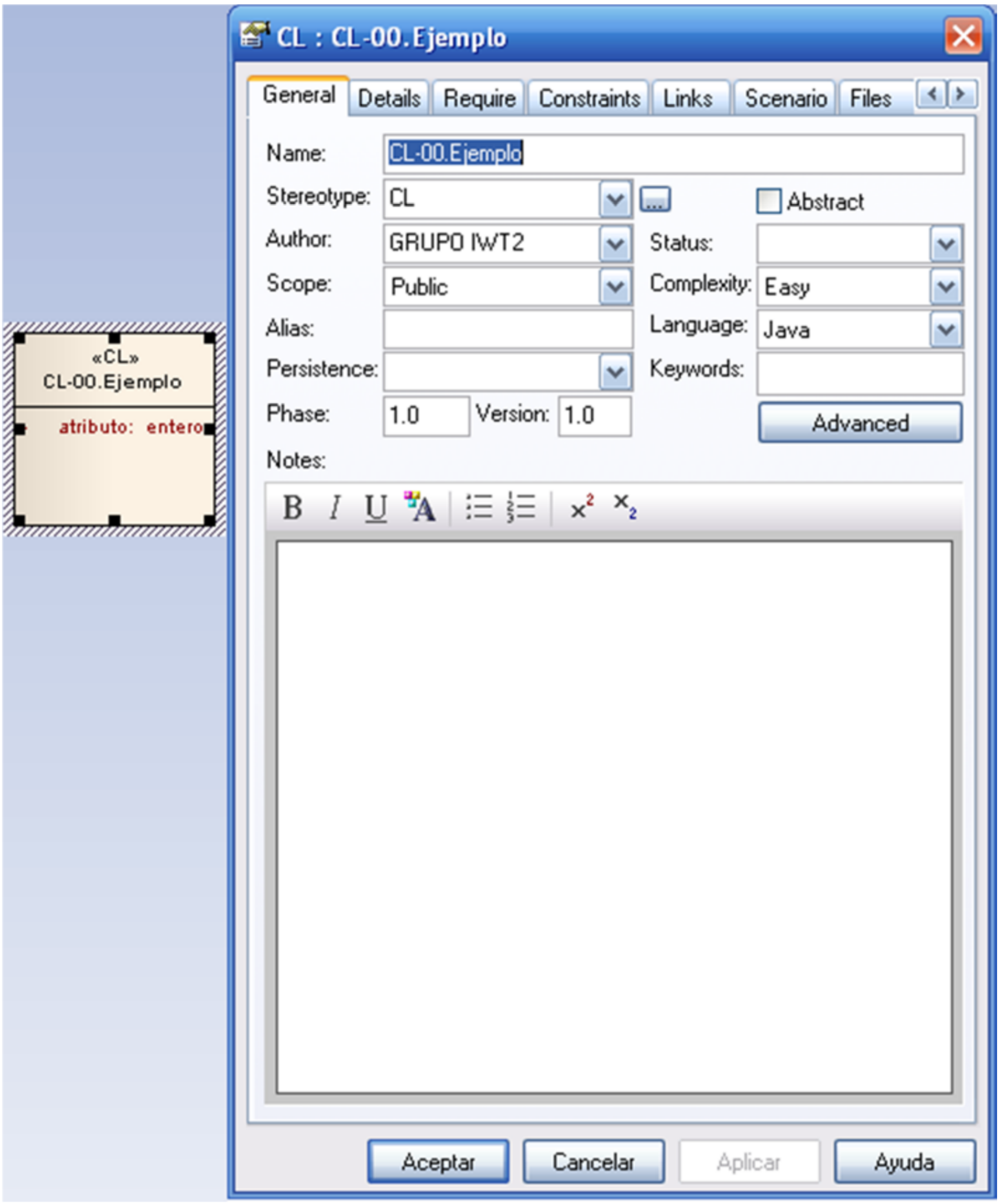

Figure 70. Standard properties of persistent classes

In CL and CLN artifacts there are a number of fields that are mandatory for the description of classes of persistence is considered correct. This series of fields are:

- **Name (Name):** the name must meet the following format CL-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of non-functional requirements . CLn-XX.Nombre in the case of the CLn.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Language** (field Language): This field specifies the programming language in which it is the artifact.

- **Attributes** (in Details tab, Attributes button): The attributes represent the set of descriptors of a CL from the point of view of users and the client. How to add attributes is explained in paragraph4.2 (Figure 15). Remember that a mandatory attribute must have a name, type, description, and cardinality.

There are also other fields are optional, but recommended that the form is completed for a better definition of the classes. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.

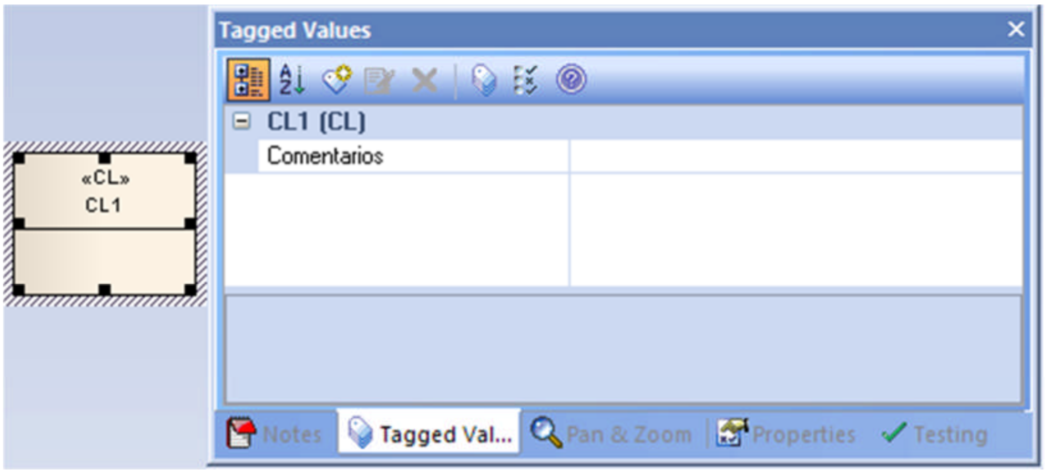

Figure 71. Tagged values of a class analysis

Persistent classes can be bound together by the connectors' **aggregation**, **association**, **composition** and **generalization** that appears in the toolbox Figure 69 Figure 35. These connectors are extensions of the connectors' aggregation, association, composition and generalization of UML. To make use of these connectors, simply click on the toolbox to select the connector, click on the class of origin and drag to the target class in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Table 28. Persistence Class Rules (CL and CLN)

Diagrams Display Prototype
You must have a diagram of type persistent classes
All CL and CLn should be contained in the diagram
Prototype Definition Display
The name of the artifact is CL-XX (X). Name or CLn-XX (X). Name
The artifact name must be unique, using the CamelCase notation and starting in uppercase
The artifact has description
There is no other RA with the same number
The attribute name is not empty

The attribute name must begin in lower case, using the CamelCase notation
The language of the appliance must be a valid programming language
The type attribute is not empty
The type attribute must be a type of language fixture
The description attribute is filled
A CL / CLn for each RA / NA
Each attribute of the RA / NA is in the CL / CLn
The type attribute of the CL / CLn is equal to the RA / NA
The cardinality of the attribute of the CL / CLn is equal to the RA / NA
If there is an attribute of type RA in the RA / NA, there is an association between CL / CLn equivalent
The cardinality of the association is the cardinality of the attribute to create the association in the RA / NA

11.3. Navigation Model

The navigation model can vary substantially depending on the actor at all times to interact with the system. So the actors are defined in the study from the actors defined in the requirements and conduct a navigation diagram for each of the actors under study.

Table 29. Navigation Class Rules

Navigation Class Diagram
You must have a navigational chart type Model
All nodes, Queries, Indexes and menus should be contained in the diagram
Artifacts can only be linked to others through the links Navigate

11.3.1. Actors studio

The actors under study are defined as groups of actors defined requirements and artifacts are represented by AE-XX in NDT-Profile. This will only be required when the computer detects that actors can be grouped to reduce the need to develop models for different actors alike. Figure 72 shows the model defined toolbox actors.

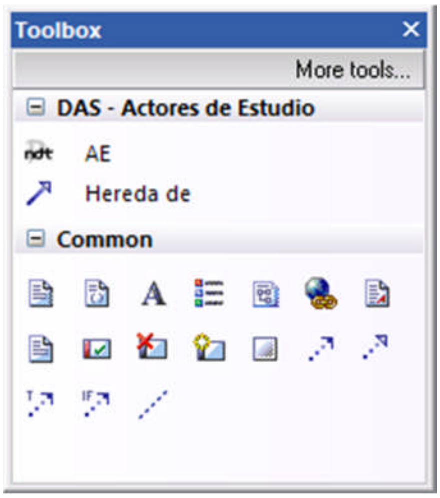

Figure 72. Study Toolbox for actors

To create an actor, just click on the toolbox in the artifact AE and drag to the diagram where you want to model. In Figure 73 and Figure 74 shows an actor to study their properties and their values labeled standards.

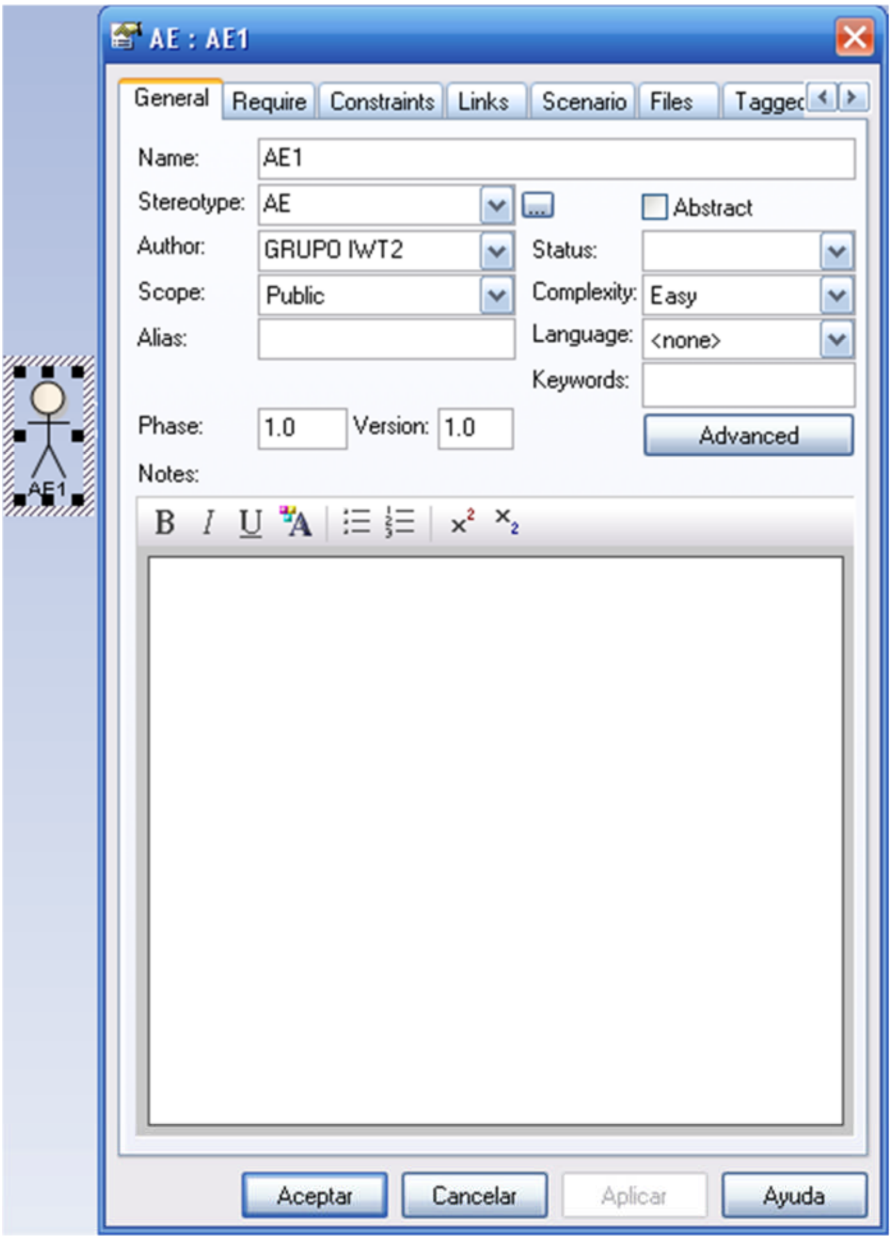

Figure 73. Standard properties of an actor to study

In the artifact AE there are a number of fields that are mandatory for the description of the actor is considered correct. This series of fields are:

- **Name** (Name): Each actor should be classified with a code and a descriptive name. As shown in Table 14, the name must meet the following format AE-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of players.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.

- **Language** (field Language): This field specifies the programming language in which it is the artifact.

There are also other fields are optional, but recommended that the form is completed for a better definition of the actors. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Collection** (Collection Classes button in the Details tab): Through this section shall indicate if the class is a collection. If it were not, would not have to fill anything.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.

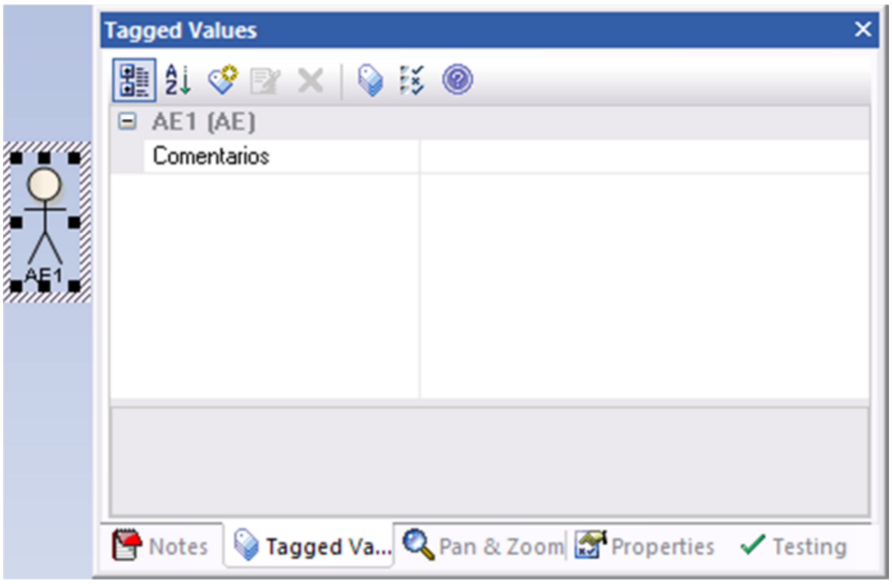

Figure 74. Tagged values of an actor to study

As for the actors of the system requirements, the study generalization between actors is permitted and is manifested with the existing connector **Inherits From** the toolbox of Figure 74. To use this connector, simply click on it in the toolbox, click on the original actor and drag to the target actor in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Table 30. Rules of Actors Studio (AE)

Diagrams Actors Studio
You must have a type diagram Actors studio
All actors must be studied in the diagram contents
You may only link by link "inherited from"
Definition of Actors Studio
The name of the artifact is AE-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase

The description is filled
No other AE with the same number
There is one AE for each AC
The language of the appliance must be a valid programming language

11.3.2. Nodes

A node is a point of navigation where the user can work with information: retrieve or modify data in the system and have the same functional possibilities. The nodes are generated from the prototype display. Basically, each prototype display generates a node, and the artifacts uicontrol than button-type prototype become uicontrol attributes and button-type artifacts become nodes operations. Then, in Figure 75, we see the toolbox to model the nodes, along with other navigational model classes, such as an artifact of NO.

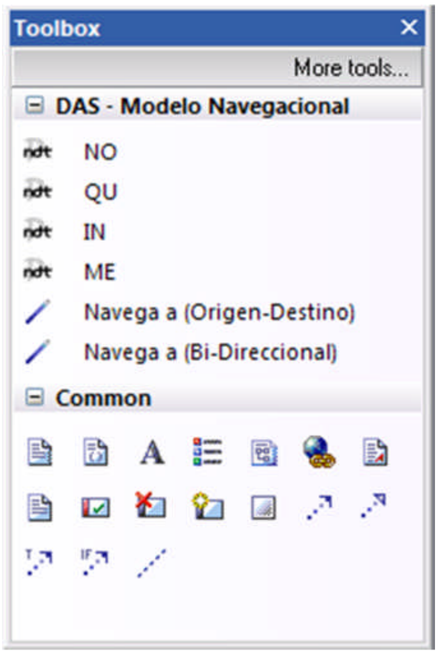

Figure 75. Toolbox for navigational model classes

In Figure 76 we see the standard properties of a node modeled as a artifact type NO.

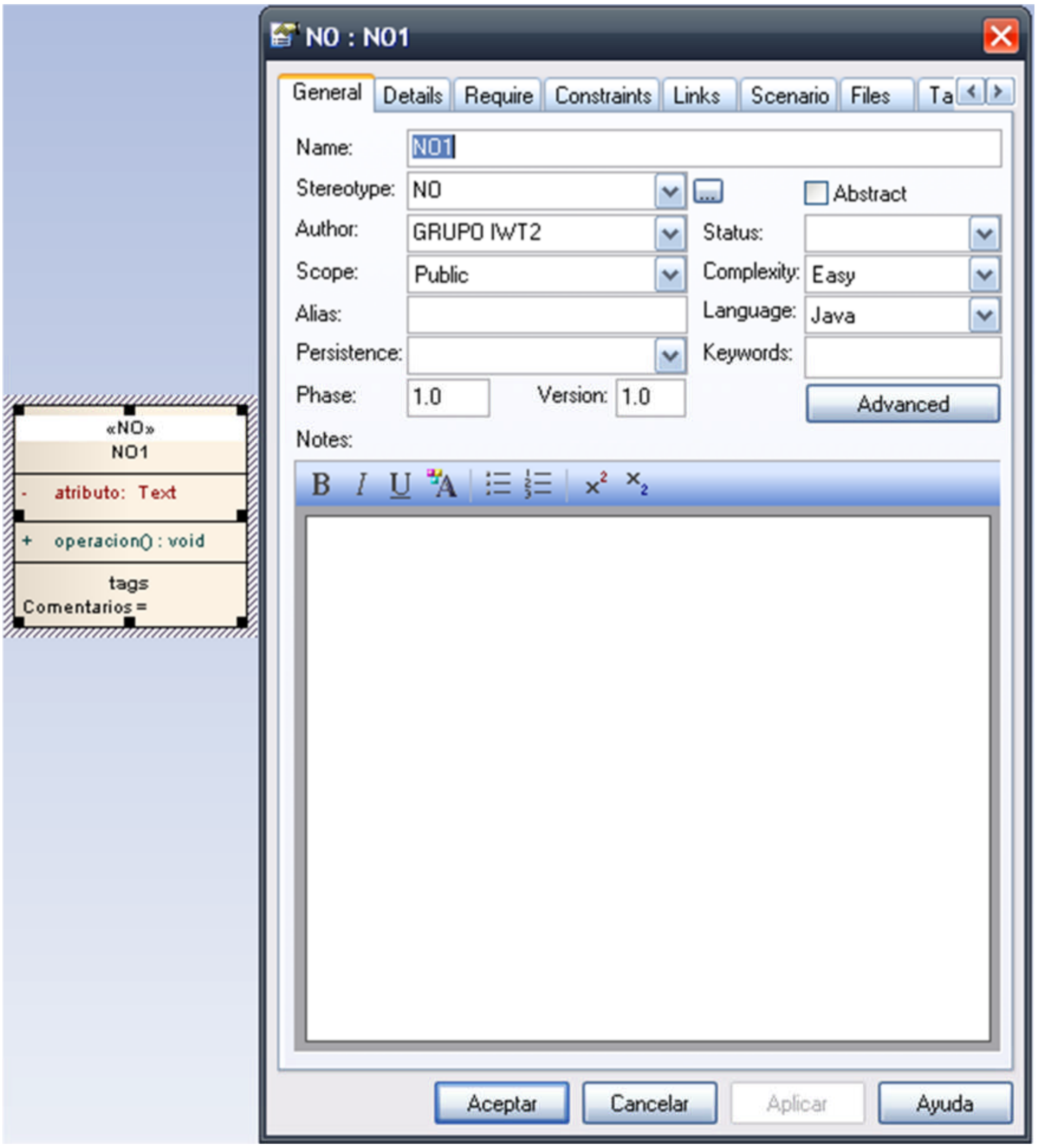

Figure 76. Standard properties and values of a node labeled

In the artifact NO there are some fields that are mandatory for the description of the node is considered correct. This series of fields are:

- **Name** (Name): Each node should be classified with a code and a descriptive name. As shown in Table 25, the name must meet the following format NO-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there are a large number of nodes.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Language** (field Language): This field specifies the programming language in which it is the artifact.

- **Artifact associated** (Alias): This field specify the artifact from which the artifact generates current, in this case, the prototype display that comes.
- **Attributes** (in Details tab, Attributes button): The attributes are created from uicontrol artifacts not of type button. How to add attributes is explained in paragraph 4.2 (Figure 15). Remember that a mandatory attribute must have a name, type, description, and a cardinality.
- **Operations** (in Details tab, button Operations): The procedures were created from the artifacts uicontrol button type. How to add operations are explained in paragraph 4.2 (Figure 17). Remember that a transaction has to be necessarily a name, alias and description.

There are also other fields are optional, but recommended that the form is completed for a better definition of the nodes. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.

Nodes can connect to any other appliance model navigation Navigate through the links, shown in Figure 75. To use this connector, simply click on it in the toolbox, click on the source node and drag to the destination artifact in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Table 31. Rules of nodes (NO)

Defining Nodes
The name of the artifact is NO-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
No other NO with the same number
The attribute name is not blank
The attribute name must begin in lower case, using the CamelCase notation
The type attribute is not empty
The type of the attribute is a CL, or an attribute of this
The description attribute is filled
The name of the operations must be the name of an RF
The description of the operation is filled
There is one NO for each PV
Each attribute of the PV is in the NO
The type attribute in NO is the CL corresponding to the RA attribute of PV
The cardinality of the attribute is not equal to the PV
The operation in NO is of the class Control of the operation of PV
If there is a single bond (not multiple) between two PV, a link must be identical between the two corresponding NO
If there is a link between PV and FR, the same should be a link between NO and corresponding QU
The NO language must be a valid programming language

11.3.3.Queries

A query represents those points where the navigation system requests user information that is essential to continue browsing. The queries are generated from the frases. Basically, every frase creates a query, and artifacts uicontrol than button-type prototype become uicontrol attributes and button-type artifacts become the operations of the queries. Figure 75, shows the toolbox to model the queries, along with other navigational model classes, such as an artifact of QU.

In Figure 77 we see the standard properties of a query modeled as an artifact of type QU.

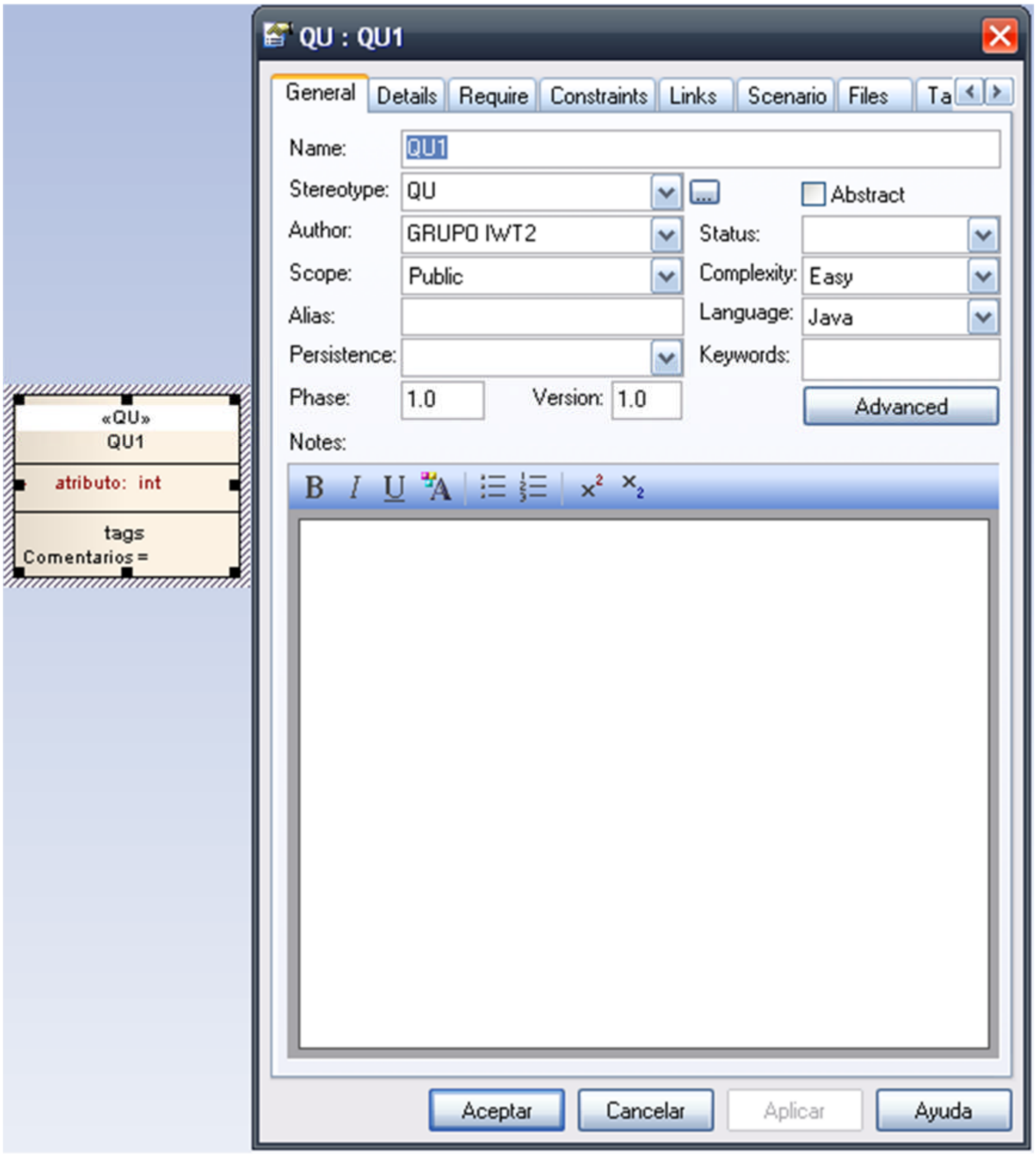

Figure 77. Standard properties and tagged values of a query

In the artifact QU there are a number of fields that are mandatory for the description of the query is considered correct. This series of fields are:

- **Name (Name):** the name must meet the following format QU-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of queries.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Language** (field Language): This field specifies the programming language in which it is the artifact.
- **Artifact associated** (Alias): This field specify the artifact from which the artifact generates current, in this case, the prototype display that comes.
- **Attributes** (in Details tab, Attributes button): The attributes are created from uicontrol artifacts not of type button. How to add attributes is explained in paragraph 4.2 (Figure 15). Remember that a mandatory attribute must have a name, type, description, and a cardinality.
- **Operations** (in Details tab, button Operations): The procedures were created from the artifacts uicontrol button type. How to add operations are explained in paragraph 4.2 (Figure 17). Remember that a transaction has to be necessarily a name, alias and description.

There are also other fields are optional, but recommended that the form is completed for a better definition of the nodes. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.

The queries can be connected to any other appliance model navigation **Navigate** through the links, shown in Figure 75. To use this connector, simply click on it in the toolbox, click on the query source and drag to the destination artifact in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Table 32. Rules Queries (QU)

Defining Queries
The name of the artifact is QU-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
There is no other with the same number QU
The attribute name is not blank
The attribute name must begin in lower case, using the CamelCase notation
The type attribute is not empty
The type of the attribute is a CL, or an attribute of this
The description attribute is filled
There is a QU per FR
Each attribute FR is in the QU
The type attribute in QU is the CL corresponding to the RA attribute of FR
The cardinality of the attribute is equal to the QU FR

If there is a link between PV and FR, the same should be a link between NO and corresponding QU
The language of QU must be a valid programming language

11.3.4. Indexes

An index represents those points where the user navigation you get a list of possible results to be displayed. All referrals to the same information. The indices are generated from a multiple bond between display prototypes. If the link between two display prototypes has cardinality multiple, between nodes generated an index is created, connecting the two. Figure 75, shows the toolbox to model rates as an artifact of type IN.

In Figure 78 we see the standard properties of an index modeled as an artifact of type IN.

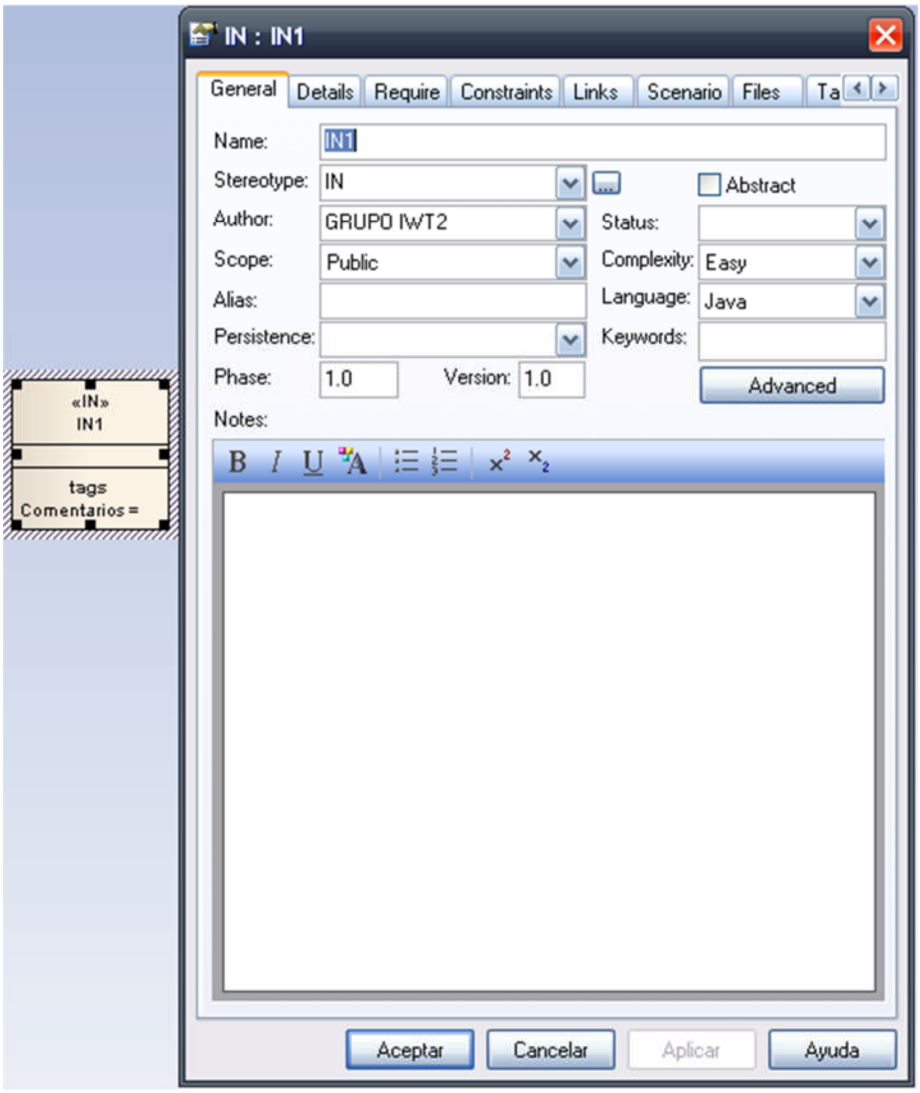

Figure 78. Properties labeled standards and values of an index

The IN artifact there are a number of fields that are mandatory for the description of the index is considered correct. This series of fields are:

- **Name** (Name): Each index should be classified with a code and a descriptive name. As shown in Table 25, the name must meet the following format IN-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of indexes.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Language** (field Language): This field specifies the programming language in which it is the artifact.

There are also other fields are optional, but recommended that the form is completed for a better definition of the indices. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.

The index can be connected to any other navigation artifact model using the links "**Navigate**", shown in Figure 75. To use this connector, simply click on it in the toolbox, click on the background rate and drag to the destination artifact in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Table 33. Rules of Indices (IN)

Defining Indexes
The name of the artifact is IN-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
IN no other with the same number
There is an IN for each multiple bond between two PV, the IN will have a link to each PV
The language of IN must be a valid programming language

11.3.5.Menus

A menu is a navigation point from which the user can go to several different options. The difference between the menu and the index is that in the index all the items listed refer to the same information. In a menu of options from which you can choose to not have to be related. The detection step is really based menu to ensure the quality of the final navigation model. The inclusion of menus in the navigation model will be aimed at ensuring that all parts of the navigational model are reachable from any other point. Figure 75 shows the toolbox to model rates as an artifact of type ME.

In Figure 79 we see the standard properties of a menu modeled as an artifact of type ME.

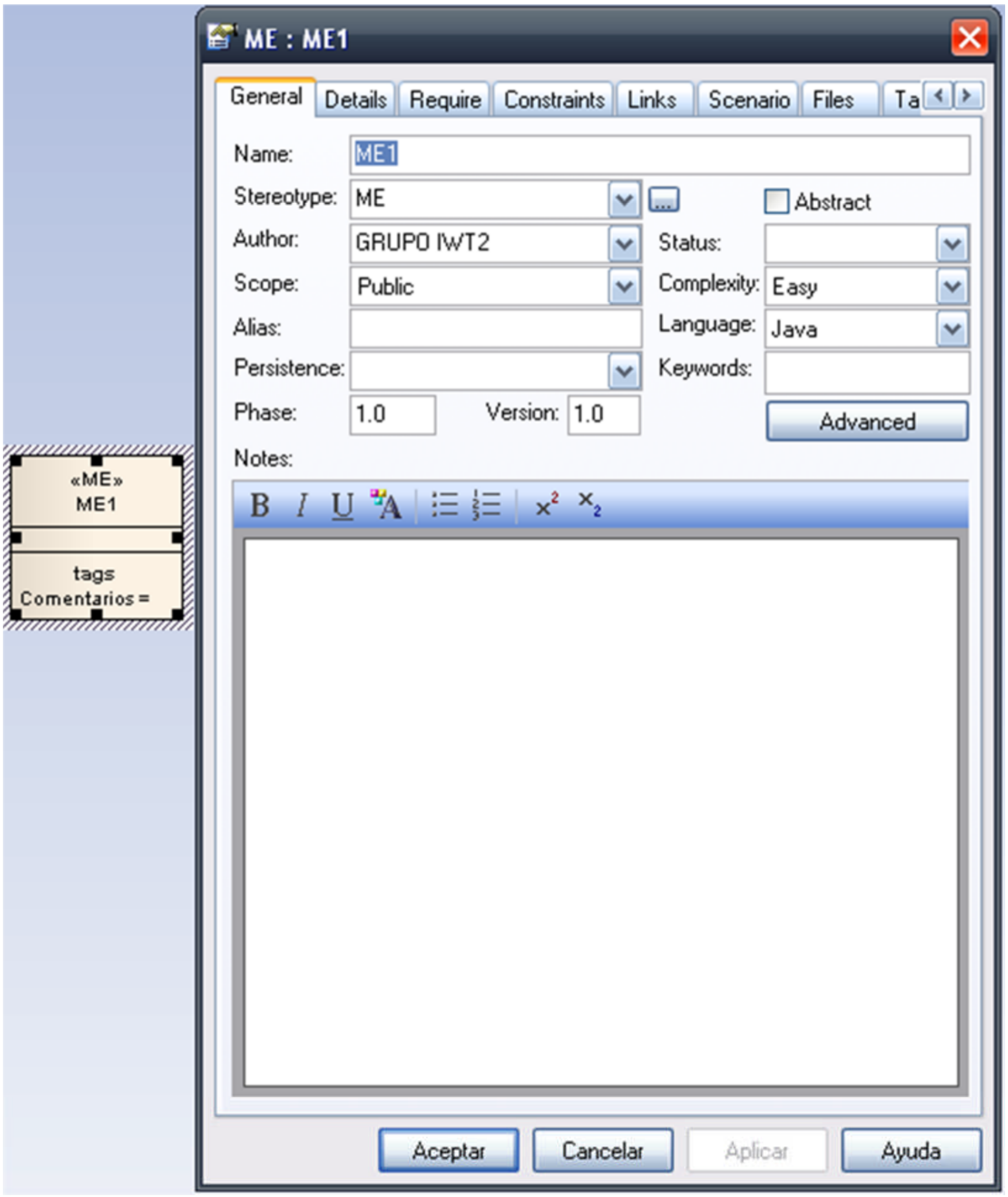

Figure 79. Properties standards and values of a menu labeled

In the ME artifact there are a number of fields that are mandatory for the description of the menu is considered correct. This series of fields are:

- **Name** (Name): Each menu should be classified with a code and a descriptive name. As shown in Table 25, the name must meet the following format ME-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of menu.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Language** (field Language): This field specifies the programming language in which it is the artifact.

There are also other fields is optional, but recommended that the form is completed for a better definition of the menus. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.

Menus can be connected to any other artifact of the navigation model using the links "**Navigate**", shown in Figure 75. To use this connector, simply click on it in the toolbox, click on the menu origin and drag to the destination artifact in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Table 34. Rules Menu (ME)

Defining Menus
The name of the artifact is ME-XX (X). Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
No other ME with the same number
The language of the ME must be a valid programming language

11.4. Abstract interface model

In general, the recommendation for the development of the abstract interface is to use the HTML prototypes generated by NDT-Prototypes. However, teams can develop a prototype interface based on the needs or model they prefer. No activity is seen as a compulsory but is recommended for use because it is very valid as validation technique with users.

It will be necessary to link the folder where are these prototypes, which should be the folder **/ docs / das / abstract interface.** The form of link is as described in Section 5.4.

12. System Design

The design phase includes the specifics of how the analysis will be implemented in the machine. It is oriented to the concrete platform with which they go to work and should match the structure of the future code.

It then describes how to model each element of the analysis using the tools provided. Each artifact has an identification code. The codes for each artifact are shown in Table 35.

Table 35. Design Nomenclature

Design	Name
Services	Service-XX.Nombre
Presentation	PR-XX.Name
Business	NE-XX.Name
Data Access	AD-XX.Name
Table	According customer

The structure of the system design document and its relationship to the structure of packages NDT profile shown in Figure 80.

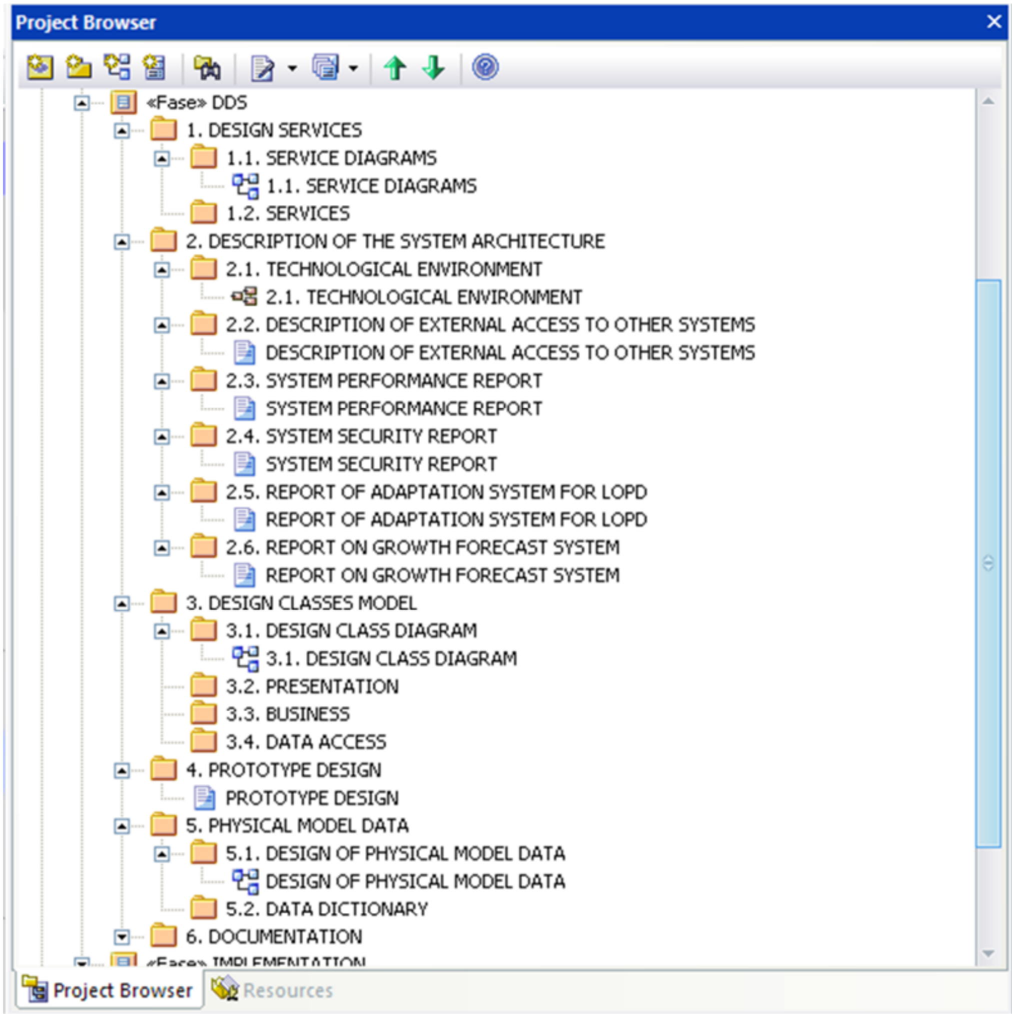

Figure 80. Structure DDS

Corresponding tools for the definition of design artifacts are shown in Figure 81.

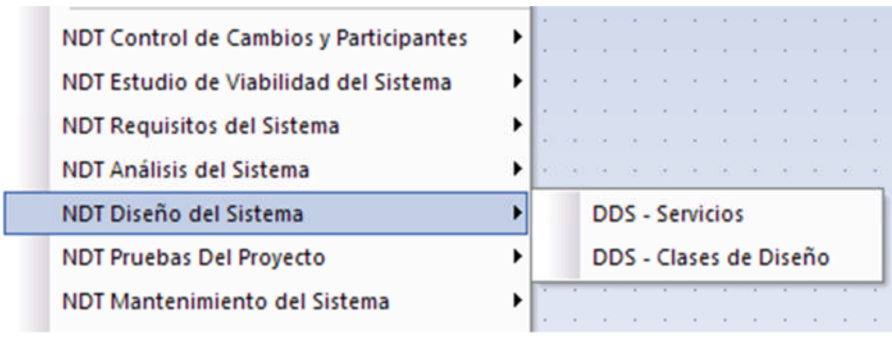

Figure 81. Design Toolboxes

12.1. Design Services

After defining the services that we encountered in the project will proceed at this phase the technical design of such services.

Figure 82 shows the toolbox identified for services such artifacts.

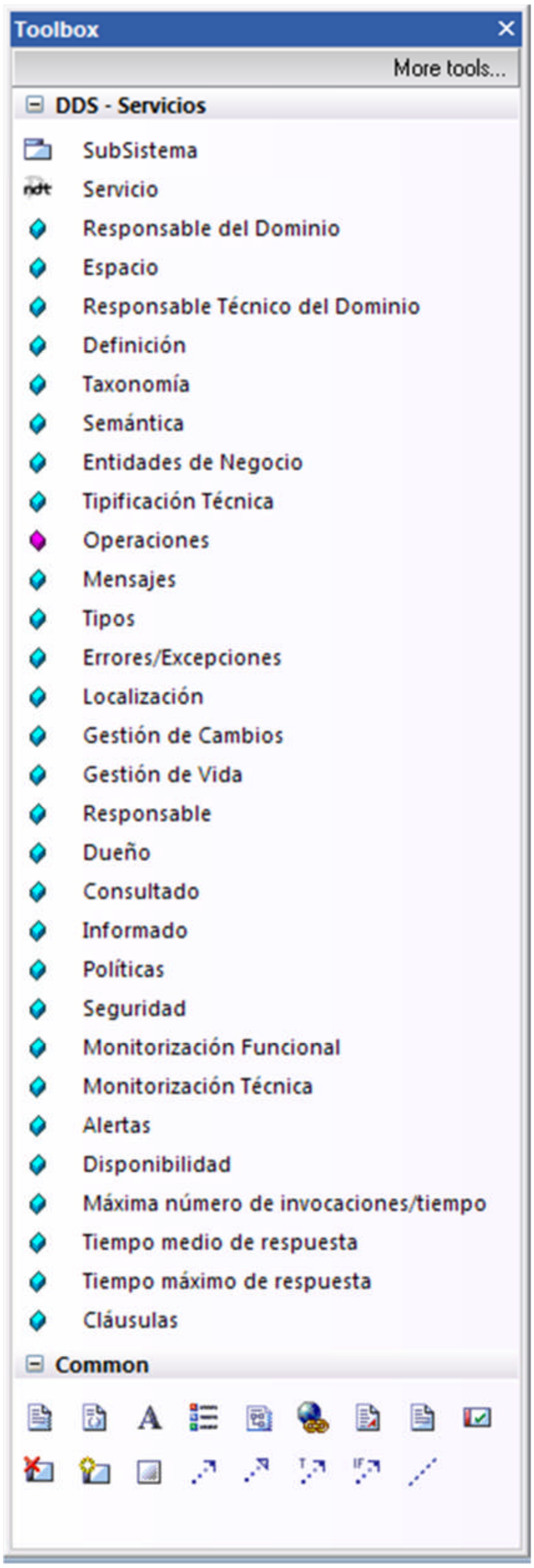

Figure 82. Toolbox design services

Figure 83 is observed standard properties of the Services. Required fields are the name of the artifact, the author, description and attributes required.

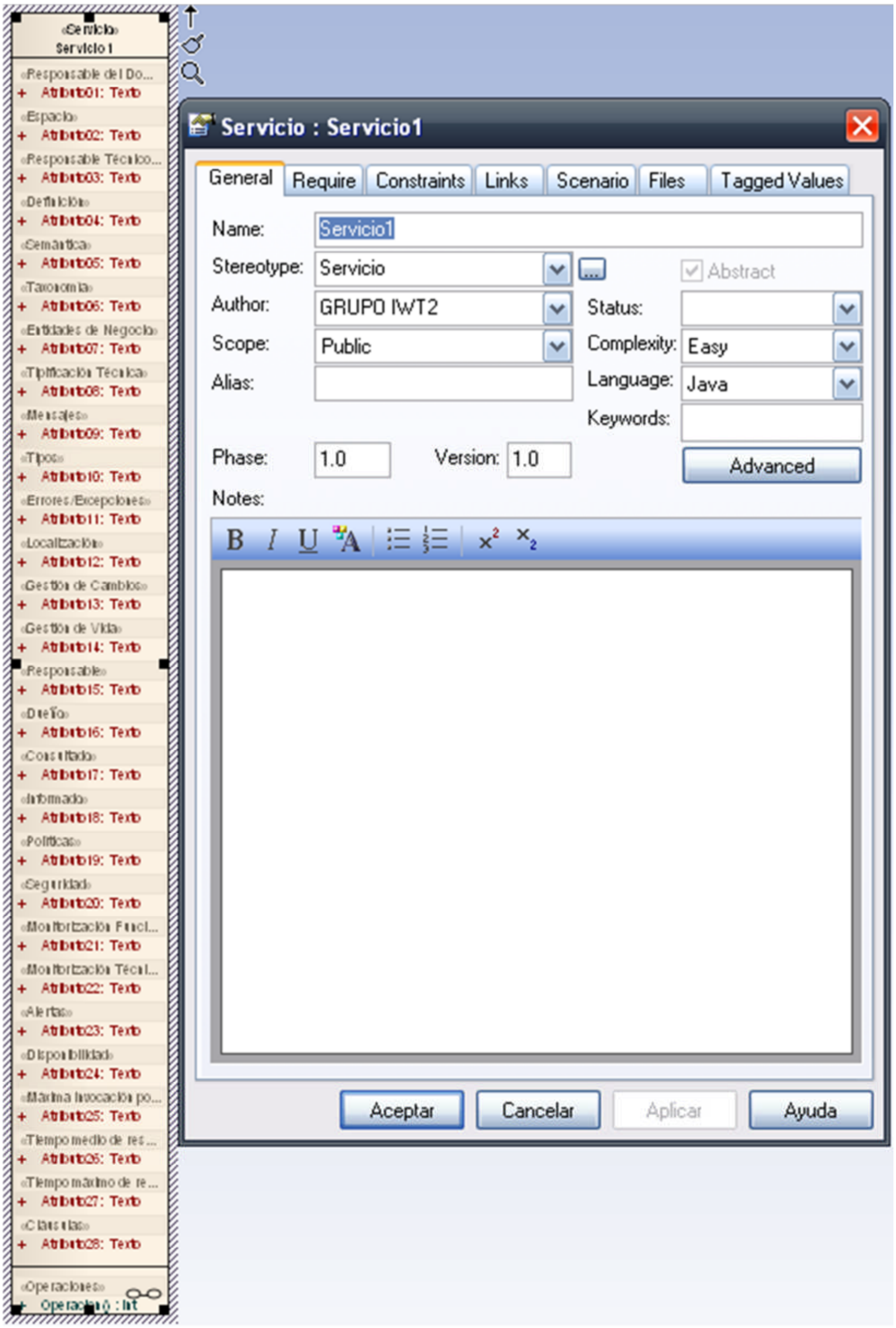

Figure 83. Standard properties of a design service

In the artifact design services there are a number of fields that are mandatory for the service description to be correct. In the case of service attributes, they are added to the artifact from the Toolbox DDS-Services, seen in Figure 67. To do this, click on the toolbox is the attribute to add and drag the desired service.

These mandatory fields are:

- **Name (Name):** As shown in Table 25, the name must meet the following format Service-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a high number of services.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the service.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the service that is being treated.
- **Responsible Domain** (attribute): This field indicates the person with ultimate responsibility of the domain.
- **Technical Manager Domain** (attribute): This field indicates the person responsible for the domain.
- **Space** (attribute): general taxonomic grouping of service. Originates namespaces that begin with the generic branch of taxonomy, and concludes with the most specific. Is best represented by the names of each of the nodes between them (eg joint-andalucia.ccul.empleados ...). Information provided by membership in a repository.
- **Definition** (attribute): Full description of the service. Must be indexable by search artifacts.
- **Taxonomy** (attribute) node of the taxonomy of services to which it belongs. Information provided by membership in a repository.
- **Semantics** (attribute): List of keywords to the evaluations made by search artifacts. Adjectives and nouns are often intimately related to the functionality of the service.
- **Business Entities** (attribute): List of business entities related to the service. Be taken in this field, the principals associated with the business (in case of using auxiliary entities that fall outside the scope of the functionality will not be included).
- **Technical Typing** (attribute): Any service must be established technically. Here is the tree of typing services:

Table 36: Technical Typing Services

Raíz	Tipo	Subtipo	Descripción
TYPOLOGY	PROCESS	COORDINATION	Coordinates a sequence or flow formed by several processes.
		PROCESS	Is responsible for performing logic functions of processes (decision making, timing, etc.)..
	FUNCTIONAL	BUSINESS	Encapsulates business logic atomized.
		PROXY	Provides a remote business functionality implemented by distributed applications.
		WRAPPER	Encapsulates business functionality provided by legacy applications.
	TECHNOLOGICAL	CONTROL	Provides horizontal sharing, security, session, etc. (Dispatcher, façades ...).
		UTILITY	Provides functionality outside the business (calculations, helpers ...).

- **Messages** (attribute): This is the information they require operations to vary their behavior. The message is return. Usually represented by input and output parameters.

- **Type** (attribute): Constraints of form and content of messages. It is recommended that type definitions are standard and serializable.
- **Errors / Exceptions** (attribute): Possible malfunctions of operations.
- **Location** (attribute): Point of service invocation.
- **Change management** (attribute) data are versions, branches of development, obsolete features, compatibility, date related, responsible for the changes, reasons for them, history of previous versions, etc.
- **Life Management** (attribute): Data related to the state it was found service in certain contexts (development, testing, integration, production). For example, maintains that the active versions of a service in development are x and x 1, while production is x-3. It also maintains data dependencies (should be minimal in the case of services) which may be by the membership of service to a BPM.
- **Responsible** (attribute): Responsible for the deliverables of the contract / service.
- **Owner** (attribute): Maximum level of escalation in terms of the contract / service.
- **Retrieved** (attribute): Who should be consulted before taking any action on this contract / service.
- **Informed** (attribute): Who should be informed of any decision or action to be taken on this contract / service.
- **Politics** (attribute): Specification of the policies implemented by the service to be defined in the SOA governance model.
- **Segurity** (attribute): Specification of the security restrictions that apply to the service to be defined in the SOA governance model.
- **Functional Monitoring** (attribute): Functional monitoring indicators that apply to the service to be defined in the SOA governance model.
- **Technical Monitoring** (attribute).
- **Alerts** (attribute): Levels and types of warning based on indicators defined above.
- **Availability** (attribute): Indicates the slot or slots and days on which the service can be used. For example, from 8:00 to 20:00 from Monday to Friday except public holidays or 24hx7días.
- Maximum number of invocations / time (attribute): Indicates the maximum number of invocations that a client can perform in a defined time unit, for example, 10 invocations per second.
- **Average response time** (attribute).
- **Maximum response time** (attribute): Maximum message processing.
- **Clauses** (attribute): Other clauses that may apply to the particular hiring and where appropriate, non-compliance.
- **Operations** (in Details tab, button Operations): The services must have at least one operation. How to add operations are explained in paragraph 4.2 (Figure 17). Remember that a transaction of a service has to be necessarily a name, parameters, return value and description.

There are also other fields are optional, but recommended that the form is completed for a better definition of services. These are:

- **Status** (Status Field): This field contains the situation where the service is in its development process. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the service. This field is meaningless when it comes to large systems where version management is a critical task.

And the service has to be associated with two diagrams. One of them is associated by default, which is the WSDL diagram. Toolbox is complete with predefined WSDL brings Enterprise Architect 7.5. Another diagram is required SOAML Component diagram, which should be created. This diagram is predefined in Enterprise Architect 7.5. Additionally, it is advisable to make a sequence diagram of the service. For this we will use the diagram that comes default Sequence SOAML Enterprise Architect 7.5.

Table 37. Design Services Rules

Definition of Design Services
You must have a type diagram Services
The name and description must be filled
Each service that is not the repository of services has to be at least the attributes that appear in the toolbox
If a service is in the service repository, you must have all the data and have to crawl into this diagram (not to be in the project folder browser)
Every service must have an associated WSDL diagram
Every service must have an associated SoaML Component diagram

12.2. Architectural Overview

To begin to define the design of any system it is necessary to define the technology architecture system. This part is divided into a diagram to define the technological environment and a series of documents.

In the diagram of this folder should be described the technological environment, the programming language, programming environment, etc. Should be discussed with a level of detail that does not lead to errors. For this, use the diagram of components that are created by default in NDT-Profile.

The remaining documents must describe how the system accesses to external systems, the performance expected of the system, security mechanisms, adaptation to the LOPD or the anticipation of future growth. These documents should be attached to the eap file as described in Section 5.4.

12.3. Model design classes

In the class model will be the evolution of the types of analysis including the necessary design patterns, classes and methods needed for proper implementation.

Figure 84 shows the defined toolbox for design class, shared among the business classes, presentation and data access.

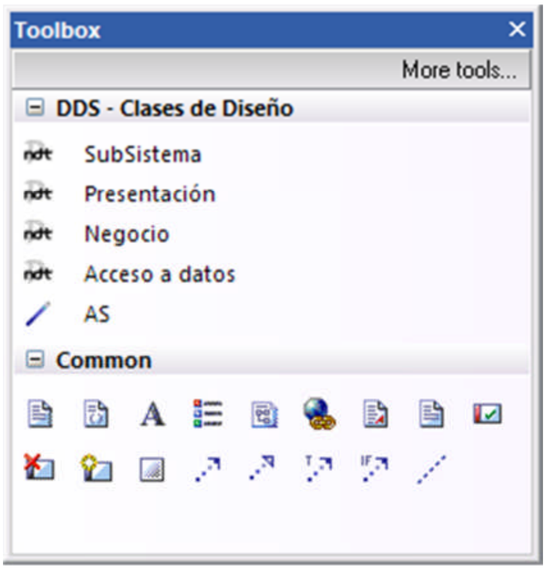

Figure 84. Toolbox design classes

12.3.1. Data Access

A data access artifact is the evolution of the artifacts of the content model of analysis. Since each type of content (CL, CLn) generates a data access class with the same attributes.

In Figure 85 we see the standard properties of an artifact access to data modeled as an artifact of type AD.

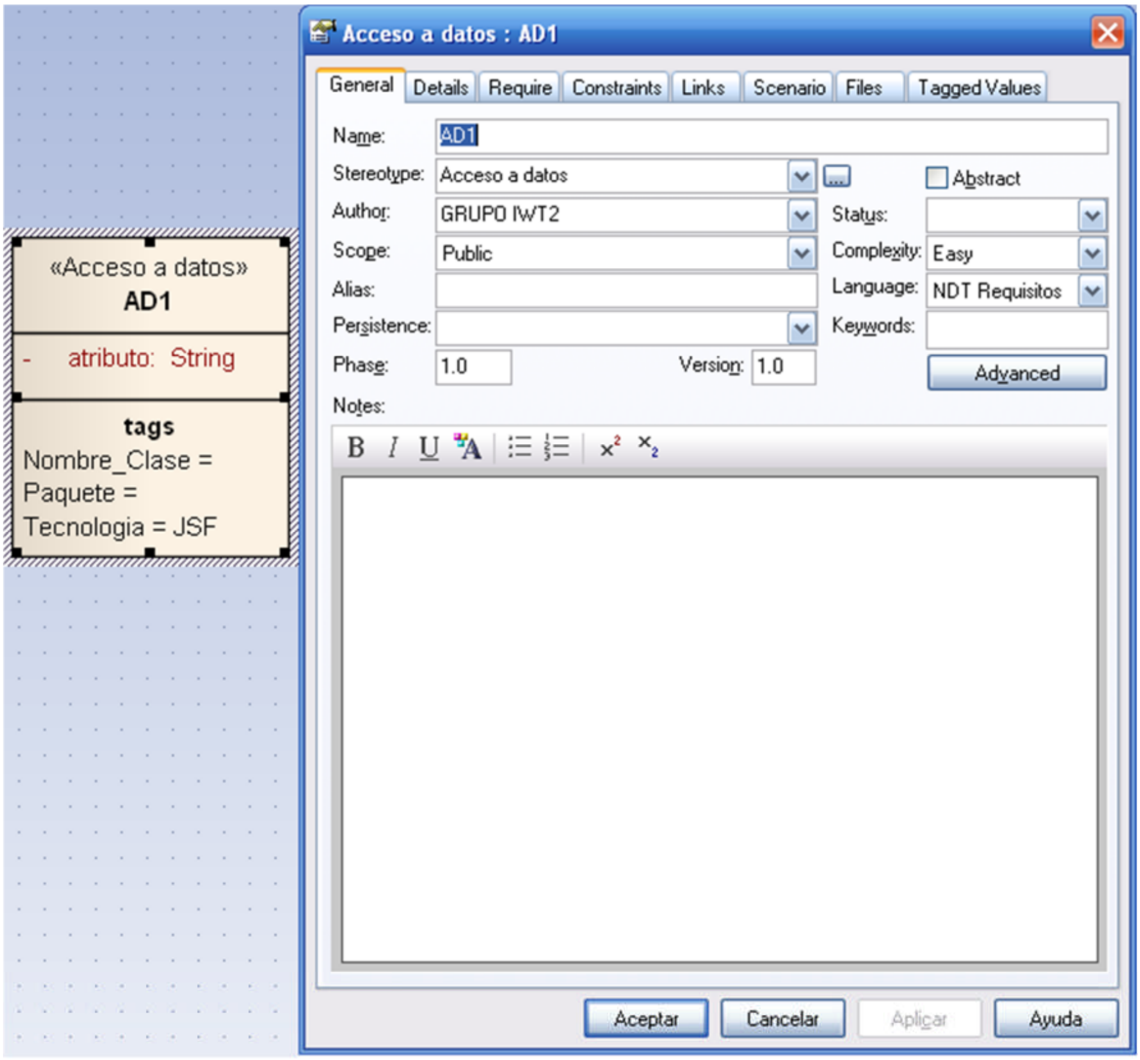

Figure 85. Standard properties and tagged values of a data access class.

In the AD artifact there are a number of fields that are mandatory for the description of the kind of access to data is considered correct. This series of fields are:

- **Name (Name):** Every kind of presentation should be classified with a code and a descriptive name. As shown in Table 35, the name must meet the following format AD-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of types of presentation.

- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.

- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.

- **Language** (field Language): This field specifies the programming language in which it is the artifact.

- **Artifact associated** (Alias): This field shall specify the artifact from which the artifact generates current, in this case, the kind of content that is appropriate.

- **Attributes** (in Details tab, Attributes button): The way to add attributes is explained in paragraph 4.2 (Figure 15). Remember that a mandatory attribute must have a name, type, description, and cardinality.

There are also other fields are optional, but recommended that the form is completed for a better definition of the nodes. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Classname** (tagged value): This field shows the exact name of the class that is implemented.
- **Package** (tagged value): This field shows the exact name of the package containing the class implements.
- **Technology** (tagged value): This field is indicated by a drop-down technology in the classroom.

The design classes can connect to any other appliance model design classes through the links "**AS" (Association),** shown in Figure 84. To use this connector, simply click on it in the toolbox, click on the home appliance and drag to the target artifact in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide

Table 38. Data Access Rules (AD)

Definition of Data Access Classes
The name of the artifact is AD-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
No other AD with the same number
The attribute name is not empty
The attribute name must begin in lower case, using the CamelCase notation
The type attribute is not empty
The type of the attribute is an AD, or an attribute of this
The description attribute is filled
There is a AD for each type of content
Each attribute of a CL is in the AD
The cardinality of the attribute of AD is equal to the CL
The language of AD must be a valid programming language

12.3.2. Business

A business artifact is the evolution of the process classes. From each CP is generated an artifact of business with the same operations.

In Figure 86 we see the standard properties of an artifact of modeling business as an artifact of type NE.

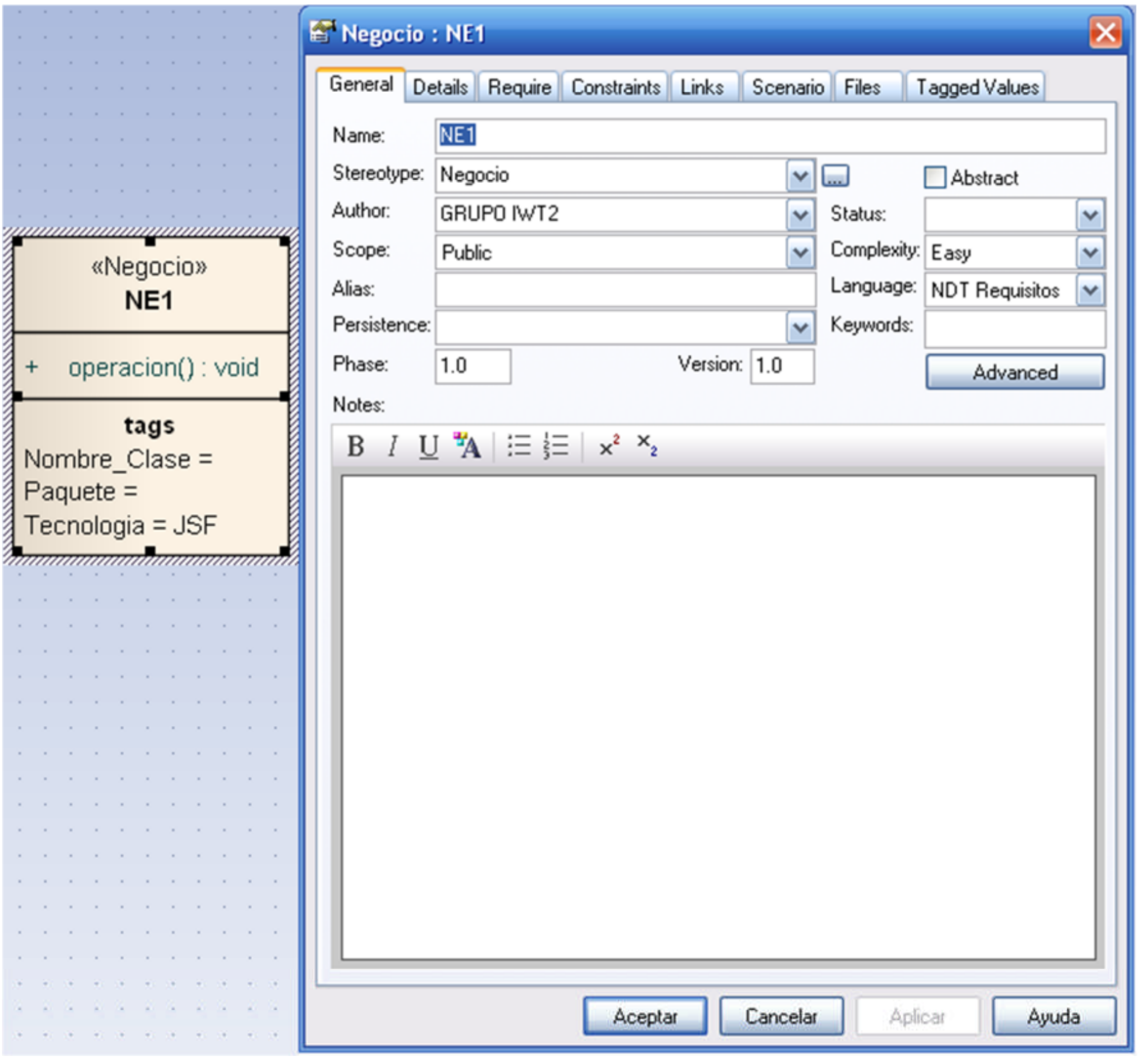

Figure 86. Standard properties and tagged values of a class of business

NE in the artifact there are a number of fields that are mandatory for the description of the kind of business is considered correct. This series of fields are:

- **Name** (Name): Every business class should be classified with a code and a descriptive name. As shown in Table 35, the name must meet the following format NE -XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of business classes.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Language** (field Language): This field specifies the programming language in which it is the artifact.
- **Artifact associated** (Alias): This field shall specify the artifact from which the artifact generates current, in this case, the prototype display that comes.

- **Operations** (in Details tab, button Operations): The way to add operations are explained in paragraph 4.2 (Figure 17). Remember that a transaction has to be necessarily a name, nickname and a description.

There are also other fields are optional, but recommended that the form is completed for a better definition of the nodes. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Classname** (tagged value): This field shows the exact name of the class that is implemented.
- **Package** (tagged value): This field shows the exact name of the package containing the class implements.
- **Technology** (tagged value): This field is indicated by a drop-down technology in the classroom.

The design classes can connect to any other appliance model design classes through the links **AS (Association),** shown in Figure 84. To use this connector, simply click on it in the toolbox, click on the home appliance and drag to the target artifact in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Table 39. Business Rules (NE)

Business Class Definition
The name of the artifact is NE-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
No other NE with the same number
The name of the operations should begin in lowercase, using the CamelCase notation
The operation description is filled
There is a NE for each artifact CP
The language of PR must be a valid programming language

12.3.3. Presentation

A presentation artifact is the evolution of artifacts from the navigation model (NO, QU e IN). Since each artifact model navigation artifact generates a presentation on the design model classes with the same attributes and / or operations, if any.

In Figure 87 we see the standard properties of an artifact of presentation modeled as an artifact of PR.

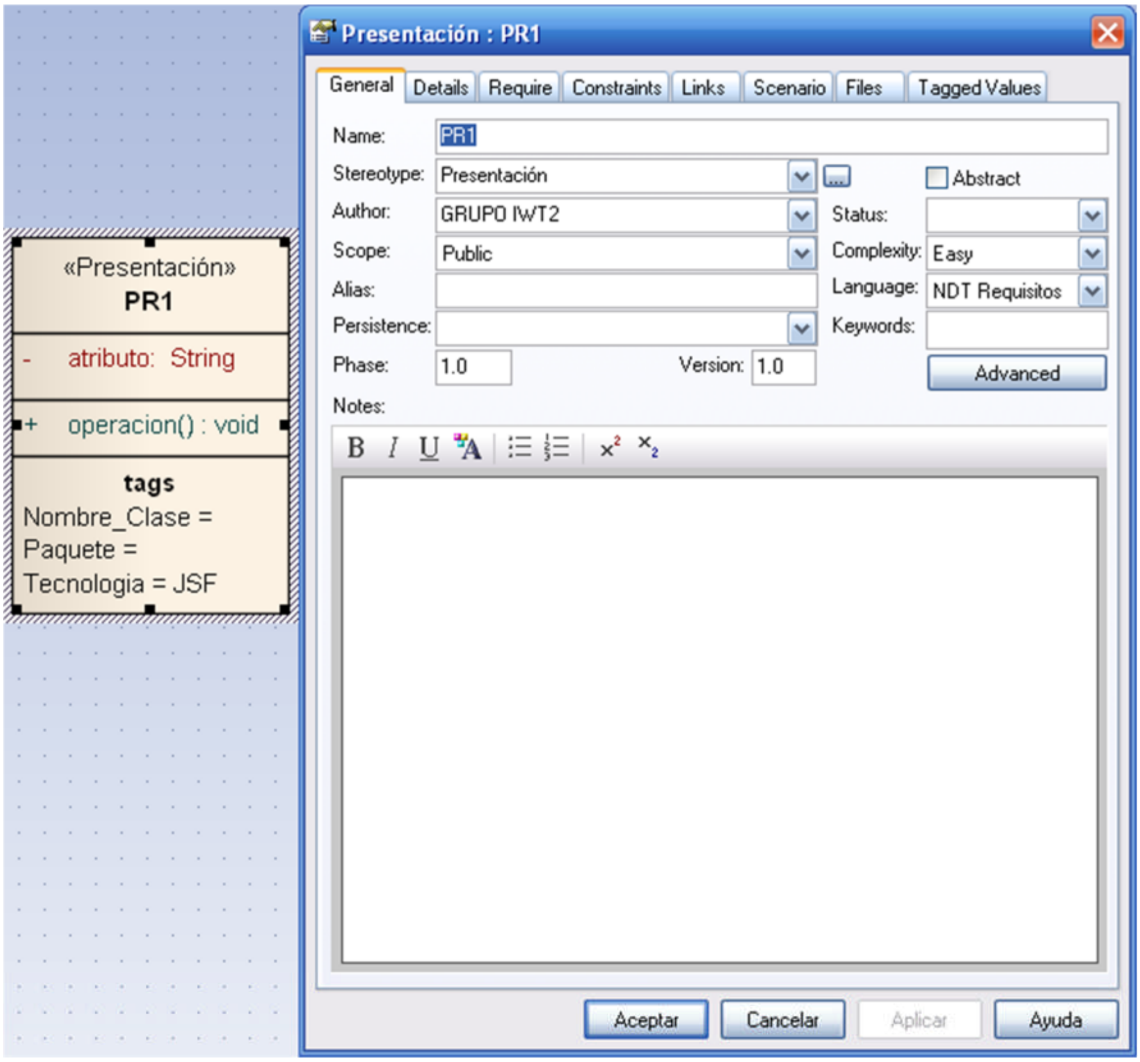

Figure 87. Standard properties and tagged values of a class presentation

In the PR artifact there are a number of fields that are mandatory for the description of the kind of presentation is considered correct. This series of fields are:

- **Name** (Name): Every presentation class should be classified with a code and a descriptive name. As shown in Table 35, the name must meet the following format PR-XX.Nombre, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of types of presentation.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Language** (field Language): This field specifies the programming language in which it is the artifact.
- **Artifact associated** (Alias): This field shall specify the artifact from which the artifact generates current, in this case, the prototype display that comes.

- **Attributes** (in Details tab, Attributes button): The way to add attributes is explained in paragraph 4.2 (Figure 15). Remember that a mandatory attribute must have a name, type, description, and a cardinality.
- **Operations** (in Details tab, button Operations): The way to add operations are explained in paragraph 4.2 (Figure 17). Remember that a transaction has to be necessarily a name, nickname and a description.

There are also other fields are optional, but recommended that the form is completed for a better definition of the nodes. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Classname** (tagged value): This field shows the exact name of the class that is implemented.
- **Package** (tagged value): This field shows the exact name of the package containing the class implements.
- **Technology** (tagged value): This field is indicated by a drop-down technology in the classroom.

The design classes can connect to any other appliance model design classes through the links **AS (Association)**, shown in Figure 84. To use this connector, simply click on it in the toolbox, click on the home appliance and drag to the target artifact in the diagram. Once you create a line that models the connector, if you want to edit its properties, would have to double click on the connector going to the properties screen that is described in paragraph 4.3 of this guide.

Table 40. Class Rules presentation (PR)

Class Definition display
The name of the artifact is PR-XX (X). Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
No other PR with the same number
The attribute name is not empty
The attribute name must begin in lower case, using the CamelCase notation
The type attribute is not empty
The type of the attribute is an AD, or an attribute of this
The description attribute is filled
The name of the operations should begin in lowercase, using the CamelCase notation
The operation description is filled
There is a PR for each navigation artifact, except ME
The language of PR must be a valid programming language

12.4. Prototype Design

If necessary because there has been reached with the definition of the abstract interface in the design phase can address the development of a prototype implementation to validate the user responsible for the area.

It is advisable from the prototypes generated by NDT-Prototypes and modify them by adding functionality, requirements and needs of the user area.

12.5. Physical data model

For systems that are based on relational databases, it will be necessary to develop the entity relationship model. The entity-relationship model is defined by the entity relationship diagram and data dictionary describing each of the elements that appear in the diagram.

Table 41. Physical Model Rules Data

Design Physical Data Model
You must have a diagram of type Data Modeling, Data Modeling with the toolbox
All Tables should be contained in the diagram
Data Dictionary
The name of the artifact must meet the specific nomenclature marked by the client
The table alias is the name of CL from which
The type of artifact is table
The description is filled
The field name is not empty
The type field is not empty
The alias of the fields is the name of the attribute from which (CL)
The description of the fields is filled

13. Implementation

The implementation phase consists simply of products resulting from the implementation of the conceptual framework defined in previous phases. The structure of packages NDT profile shown in Figure 88.

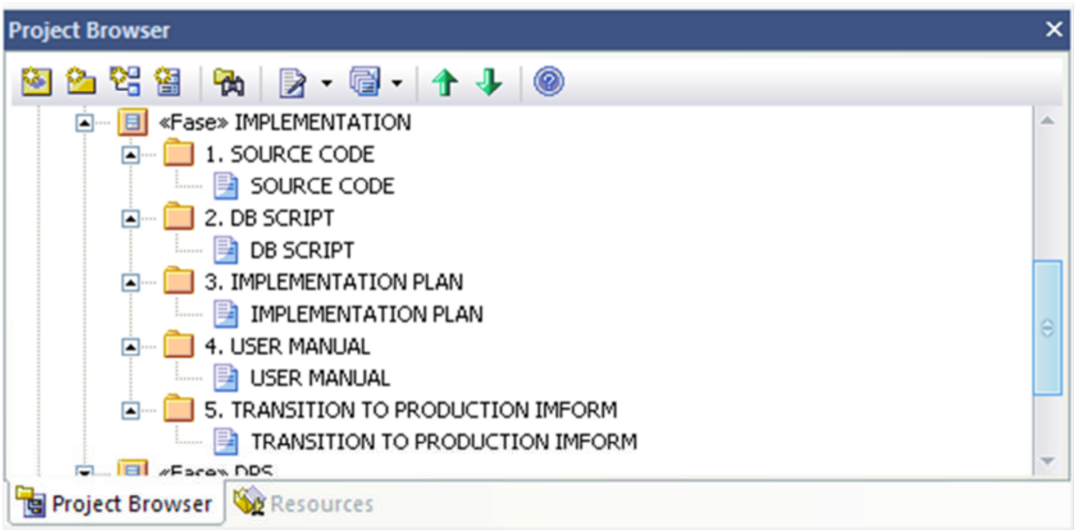

Figure 88. Structure of the construction phase

The implementation phase is divided basically in a series of documents. These documents should be attached to the eap file as described in Section 5.4. Source documents and DB script need not be filled, but must contain a link, or to subversion, or the corresponding file.

14. System Tests

The test phase, in contrast to other phases, is performed in parallel with other life cycle phases. The structure of packages NDT profile shown in Figure 89.

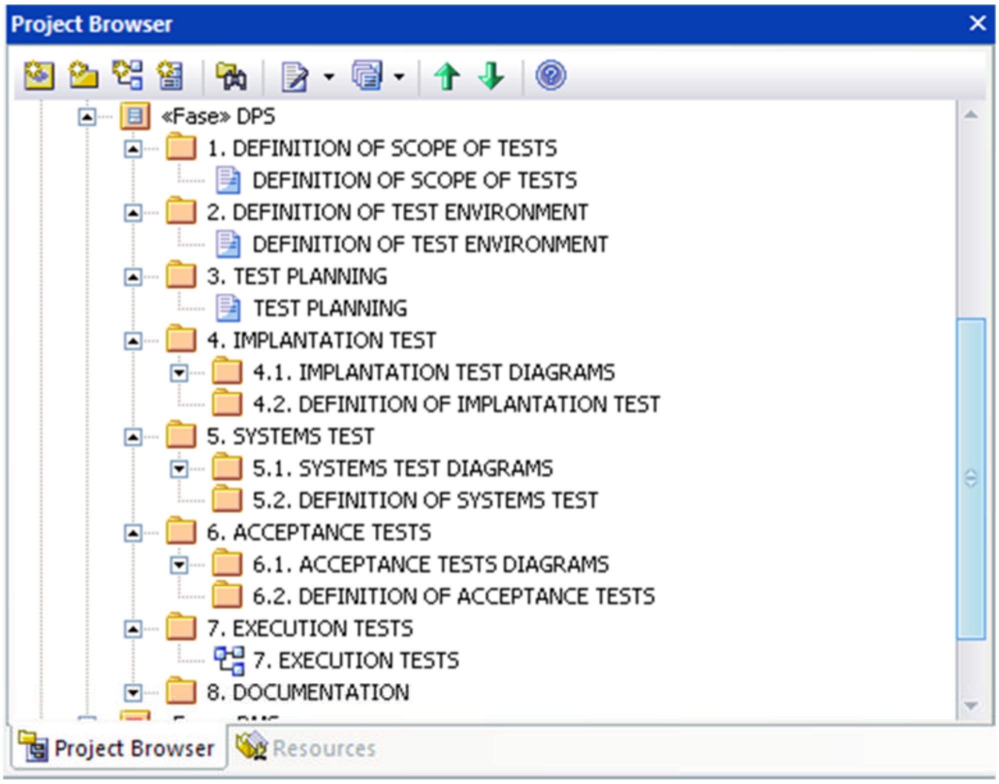

Figure 89. DPS Structure

The test phase is divided basically into a series of documents and the definition of test artifacts. The documents have to describe the depth and scope of testing, the environmental requirements necessary for the execution of the tests, and the timing of the tests. These documents should be attached to the PAD file as described in Section 5.4. In the second part, we must define three types of test artifacts. Each artifact has an identification code. The codes for each artifact are shown in Table 42.

Table 42. Test Nomenclature

Testing	Name
Implantation Tests	PI-XX.Name
System Tests	PS-XX.Name
Acceptance Tests	PA-XX.Name

The set of tools for defining tests are shown below in Figure 90.

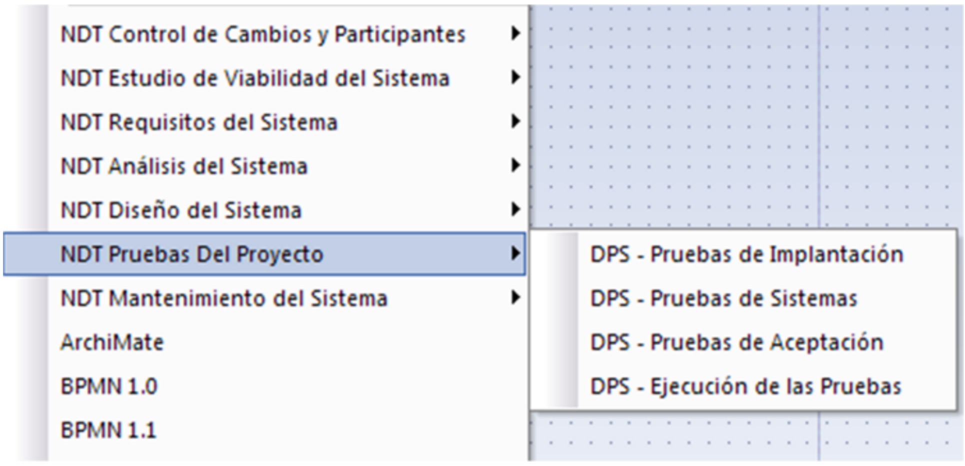

Figure 90. Test Artifacts

14.1. Implantation Tests

Implantation tests are directed to the Department of Systems. Define the tests to be performed once implemented the system to verify implementation.

Figure 48 shows the toolbox to model definite evidence of implementation artifacts such as IP.

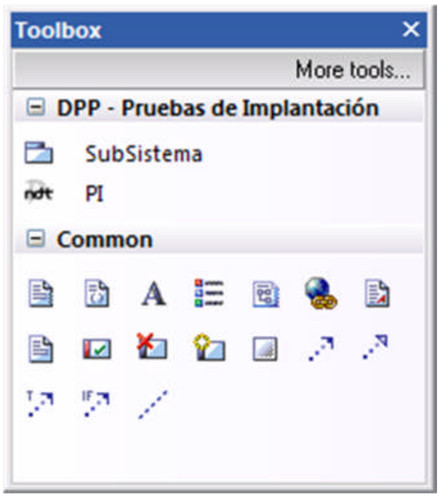

Figure 91. Toolbox Implantation tests

To create an Implantation tests, simply click on the toolbox to the PI artifact and drag to the diagram where you want to model. For more information about working with use case diagrams, please read section 4.1.1 of this guide, which explains how to be these diagrams.

Figure 92 shows an Implantation tests modeled as an artifact of type PI and property standards.

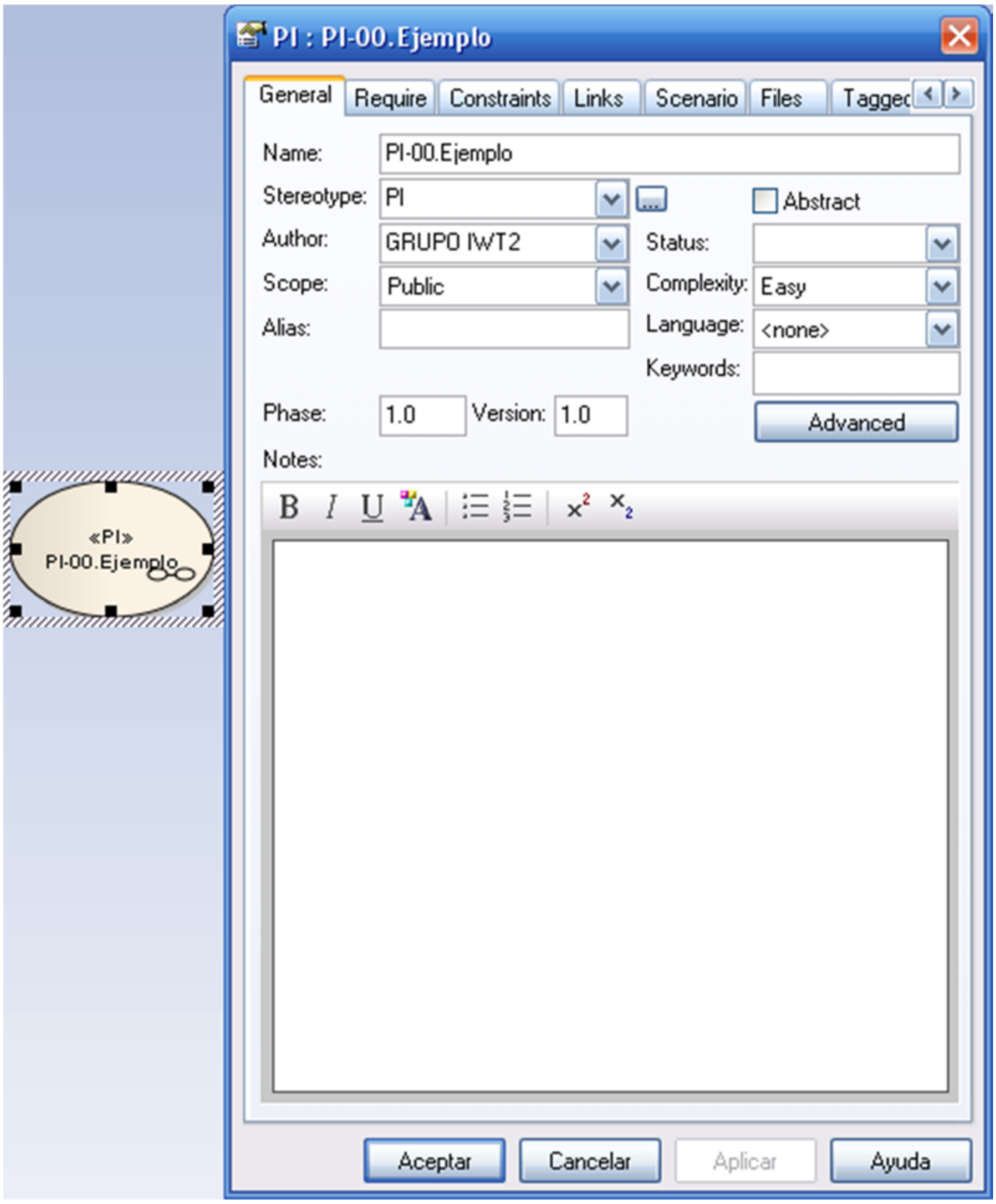

Figure 92. Standard properties of an Implantation tests

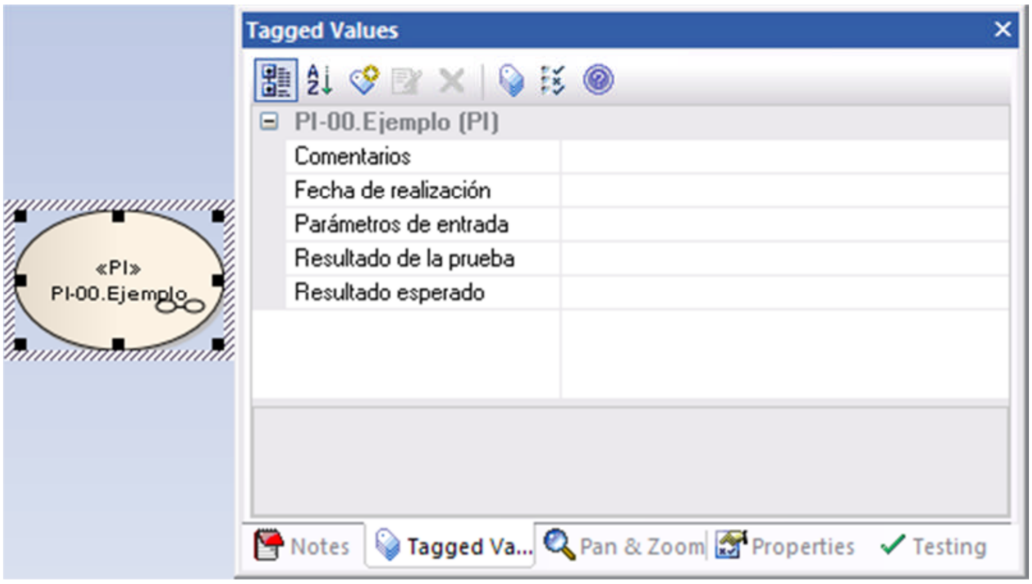

Figure 93. Tagged values of an Implantation tests

In the PI artifact there are a number of fields that are mandatory for the description of the test of implementation is considered correct. This series of fields are:

- **Name** (Name): Each Implantation tests to be classified with a code and a descriptive name. As shown in Table 42, the name must meet the following format PI-XX.Name, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of tests deployment .
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Constraints** (Constraints tab): In a functional requirement must be defined mandatory restrictions. These need to be detailed in the Constraints tab (as described in paragraph 4.2 of this guide, Figure 18) and the rates may have are pre-condition and post-conditions. If the functionality does not have any restrictions, both as pre-condition post-condition will make clear that does not apply.
- **Diagram of activities or scenarios:** The tests must necessarily be described by scenarios or an activity diagram. Both options are available, the first by Scenarios tab seen in Figure 92 and the second by double clicking on the test. If you want the functionality that will be used to describe a complex activity diagrams mandatory, which are described further in Section 4.1.2.
- **Input Parameters** (tagged value): Indicate those input parameters necessary to run the test.
- **Expected results** (tagged value): Indicate the output result should be obtained for the test to be considered satisfactory.

There are also other fields are optional, but recommended that the form is completed for a better definition of functional requirements. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.

- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.
- **Date** (tagged value): This field may indicate the date on which the test was carried out modeling artifact.
- **Test result** (tagged value): This field indicates the output result obtained in the test run.

14.2. System Tests

The system tests are derived from the use cases defined during the requirements phase. These represent the functional tests performed on the system.

Figure 94 shows the toolbox set to model the system testing artifacts such as PS.

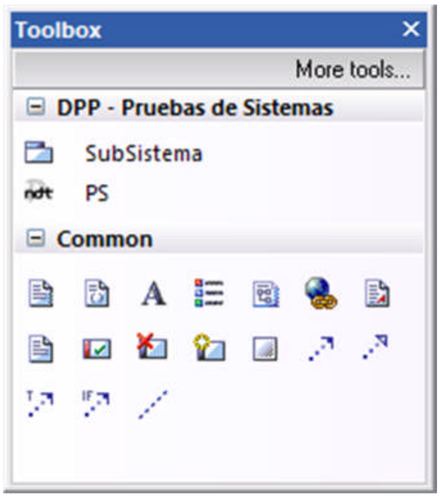

Figure 94. Toolbox for system tests

To create a system test, simply click on the toolbox to the PS artifact and drag to the diagram where you want to model. For more information about working with use case diagrams, please read section 4.1.1 of this guide, which explains how to be these diagrams.

Figure 95 shows a system test modeled as an artifact of type PS and property standards.

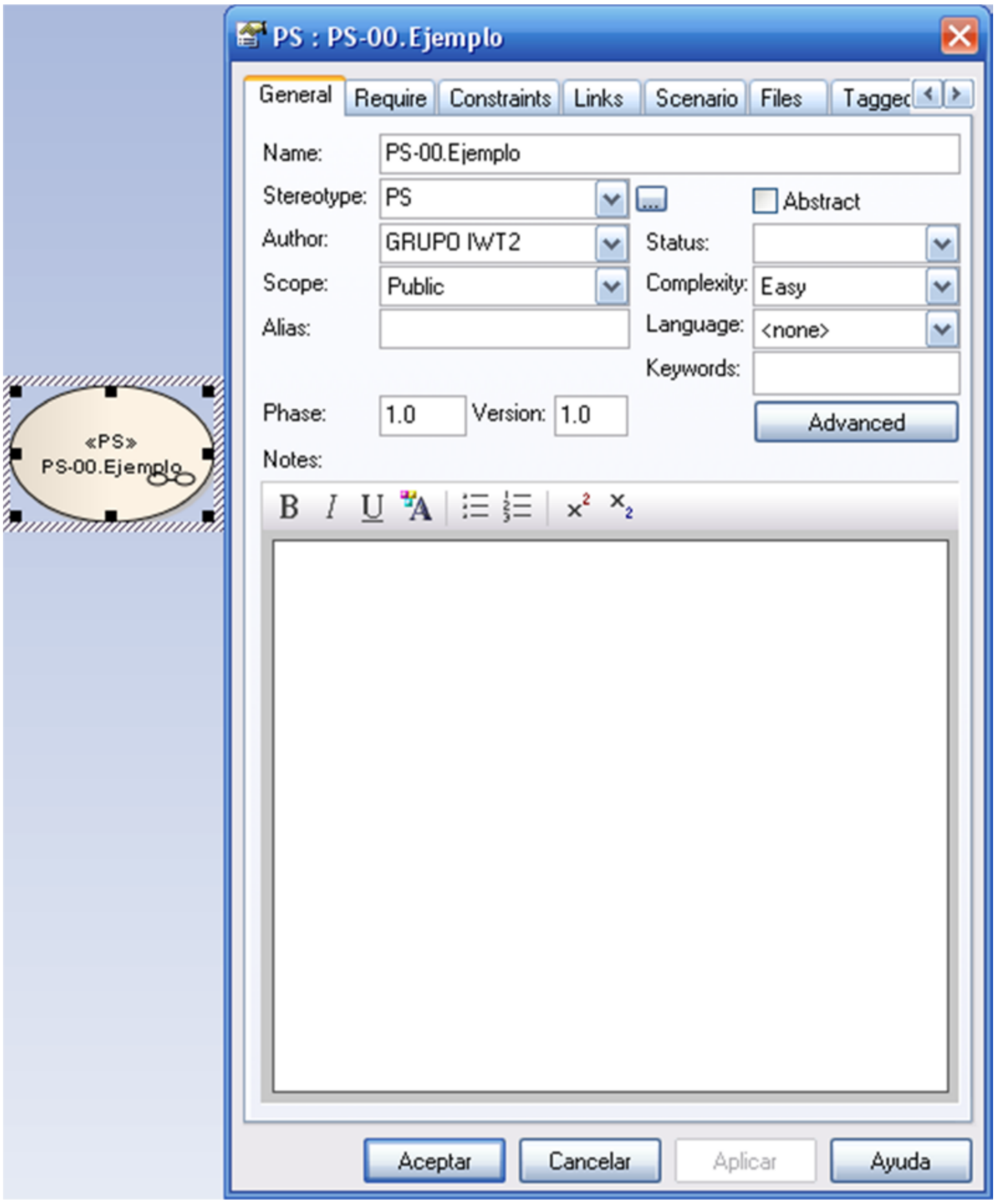

Figure 95. Standard properties of a system test

The tagged values of all tests are identical, so are the same as shown in Figure 93.

In the PS artifact there are a number of fields that are mandatory for the description of the system test is considered correct. This series of fields are:

- **Name** (Name): Each system test to be classified with a code and a descriptive name. As shown in Table 42, the name must meet the following format PS-XX.Name, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of system tests.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.

- **Constraints** (Constraints tab): In a functional requirement must be defined mandatory restrictions. These need to be detailed in the Constraints tab (as described in paragraph 4.2 of this guide, Figure 18) and the rates may have are pre-condition and post-conditions. If the functionality does not have any restrictions, both as pre-condition post-condition will make clear that does not apply.
- **Diagram of activities or scenarios**: The tests must necessarily be described by scenarios or an activity diagram. Both options are available, the first by Scenarios tab seen in Figure 92 and the second by double clicking on the test. If you want the functionality that will be used to describe a complex activity diagrams mandatory, which are described further in Section 4.1.2.
- **Input Parameters** (tagged value): Indicate those input parameters necessary to run the test.
- **Expected results** (tagged value): Indicate the output result should be obtained for the test to be considered satisfactory.

There are also other fields are optional, but recommended that the form is completed for a better definition of functional requirements. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.
- **Date** (tagged value): This field may indicate the date on which the test was carried out modeling artifact.
- **Test result** (tagged value): This field indicates the output result obtained in the test run.

The system tests should define traceability relationships between elements of the system test and use cases.

14.3. Acceptance Tests

The acceptance tests are a subset of system tests and are intended to be held by end users.

Figure 96 shows the toolbox to model definite acceptance testing artifacts such as PA.

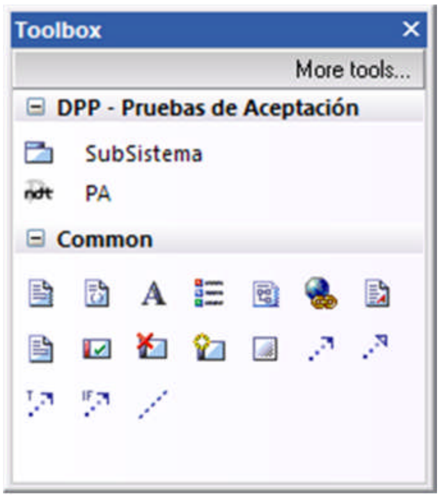

Figure 96. Toolbox for acceptance tests

To create an acceptance test, simply click on the toolbox in the PA artifact and drag to the diagram where you want to model. For more information about working with use case diagrams, please read section 4.1.1 of this guide, which explains how to be these diagrams.

Figure 97 shows a system test modeled as an artifact of type PA and property standards.

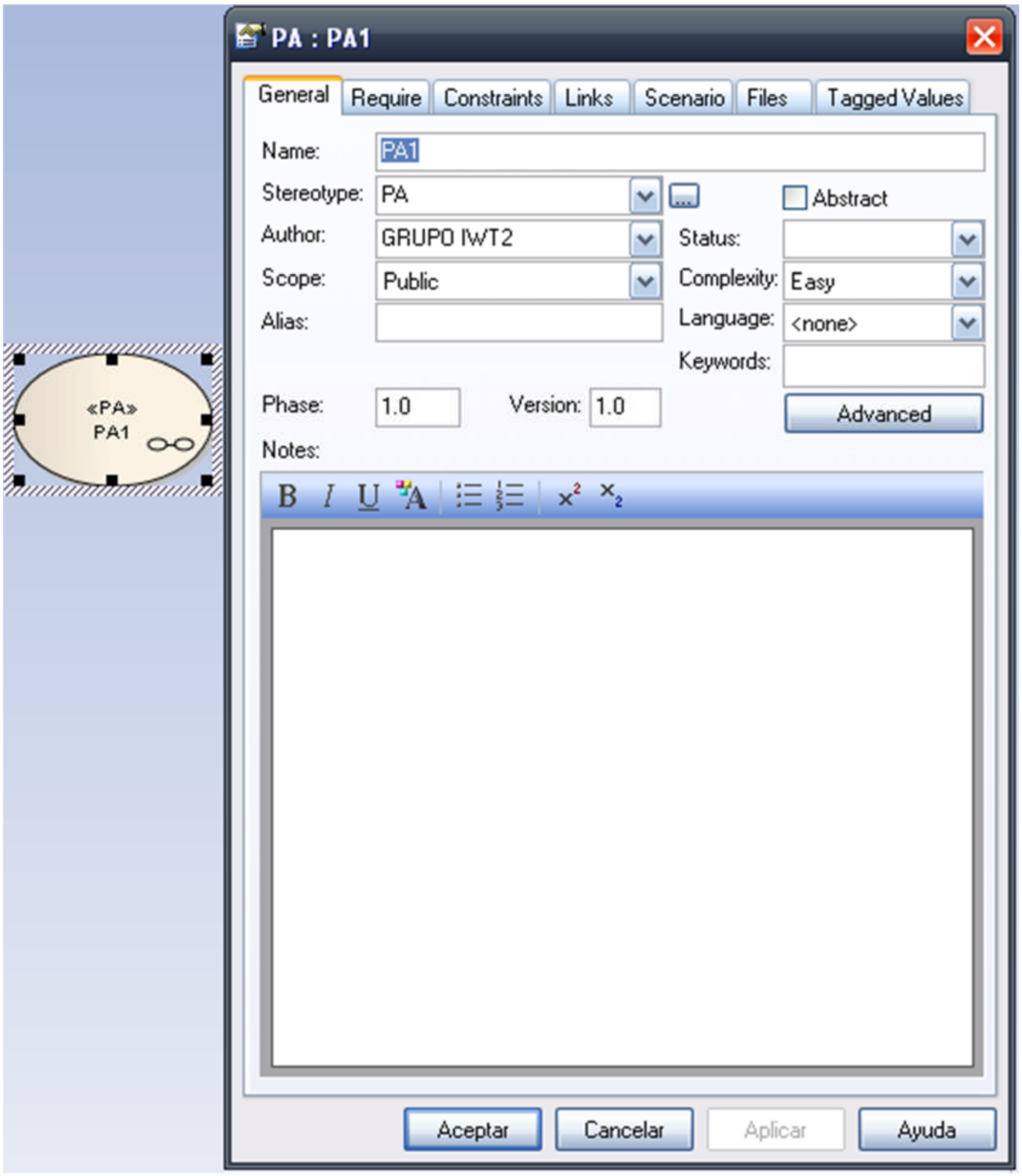

Figure 97. Standard properties of an acceptance tests

The tagged values of all tests are identical, so are the same as shown in Figure 93.

The PA artifact there are a number of fields that are mandatory for the description of the acceptance test is considered correct. This series of fields are:

- **Name** (Name): Each acceptance test to be classified with a code and a descriptive name. As shown in Table 42, the name must meet the following form PA-XX.Name, where XX is a two-digit number, or exceptionally, three figures, if there is a large number of acceptance tests .
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.

- **Constraints** (Constraints tab): In a functional requirement must be defined mandatory restrictions. These need to be detailed in the Constraints tab (as described in paragraph 4.2 of this guide, Figure 18) and the rates may have are pre-condition and post-conditions. If the functionality does not have any restrictions, both as pre-condition post-condition will make clear that does not apply.
- **Diagram of activities or scenarios**: The tests must necessarily be described by scenarios or an activity diagram. Both options are available, the first by Scenarios tab seen in Figure 92 and the second by double clicking on the test. If you want the functionality that will be used to describe a complex activity diagrams mandatory, which are described further in Section 4.1.2.
- **Input Parameters** (tagged value): Indicate those input parameters necessary to run the test.
- **Expected results** (tagged value): Indicate the output result should be obtained for the test to be considered satisfactory.

There are also other fields are optional, but recommended that the form is completed for a better definition of functional requirements. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.
- **Comments** (tagged value): This field allows the author to the target indicate any other information it deems appropriate.
- **Date** (tagged value): This field may indicate the date on which the test was carried out modeling artifact.
- **Test result** (tagged value): This field indicates the output result obtained in the test run.

Table 43. Rules of Evidence (PI, PS, PA)

Test Diagram
Each type of test should have a diagram of type Implementation Tests
All PI, PS and PA should be contained in their respective diagrams
Definition Test
The name of the artifact is PI-XX (X).Name, PS-XX (X).Name or PA-XX (X).Name
The name of PI, PS or PA must be in the infinitive
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
There is no other test with the same numbering
The test is an activity diagram, or scenarios
The scenarios should have a name and description
The scenarios must be numbered sequentially
The activity diagram must have a start node, end node and at least one activity
All objects in the activity diagram has to be named
The activities must have an outgoing link, and at least one inbound link
The start node must have an outgoing link
The end node must have at least one inbound link
If the test constraints, must have a name and description
In the diagram, all tests have an associated AC is either part of other testing

14.4. Execution Tests

This package will present the implementation details of the evidence. The test kits are a set of tests (taken from the previous sections) made by a participant in the project.

In Figure 98 show defined toolbox for modeling the outcome of the execution test.

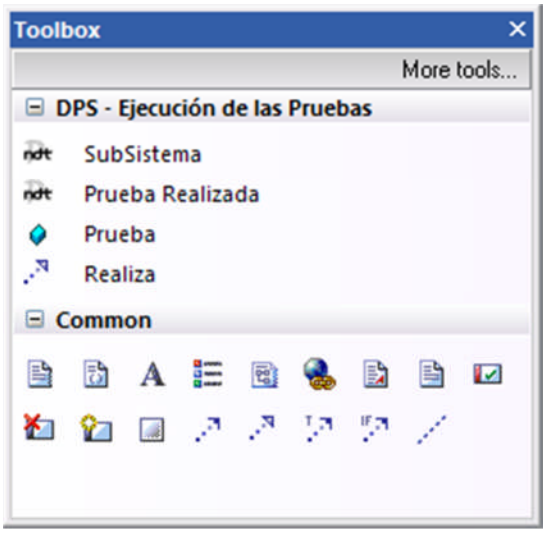

Figure 98. Toolbox execution Test

To create a battery of tests, simply click on the toolbox in the engine tests and drag to the diagram where you want to model.

Figure 95 shows a battery of tests modeled as an artifact of type Tests Performed and their property standards.

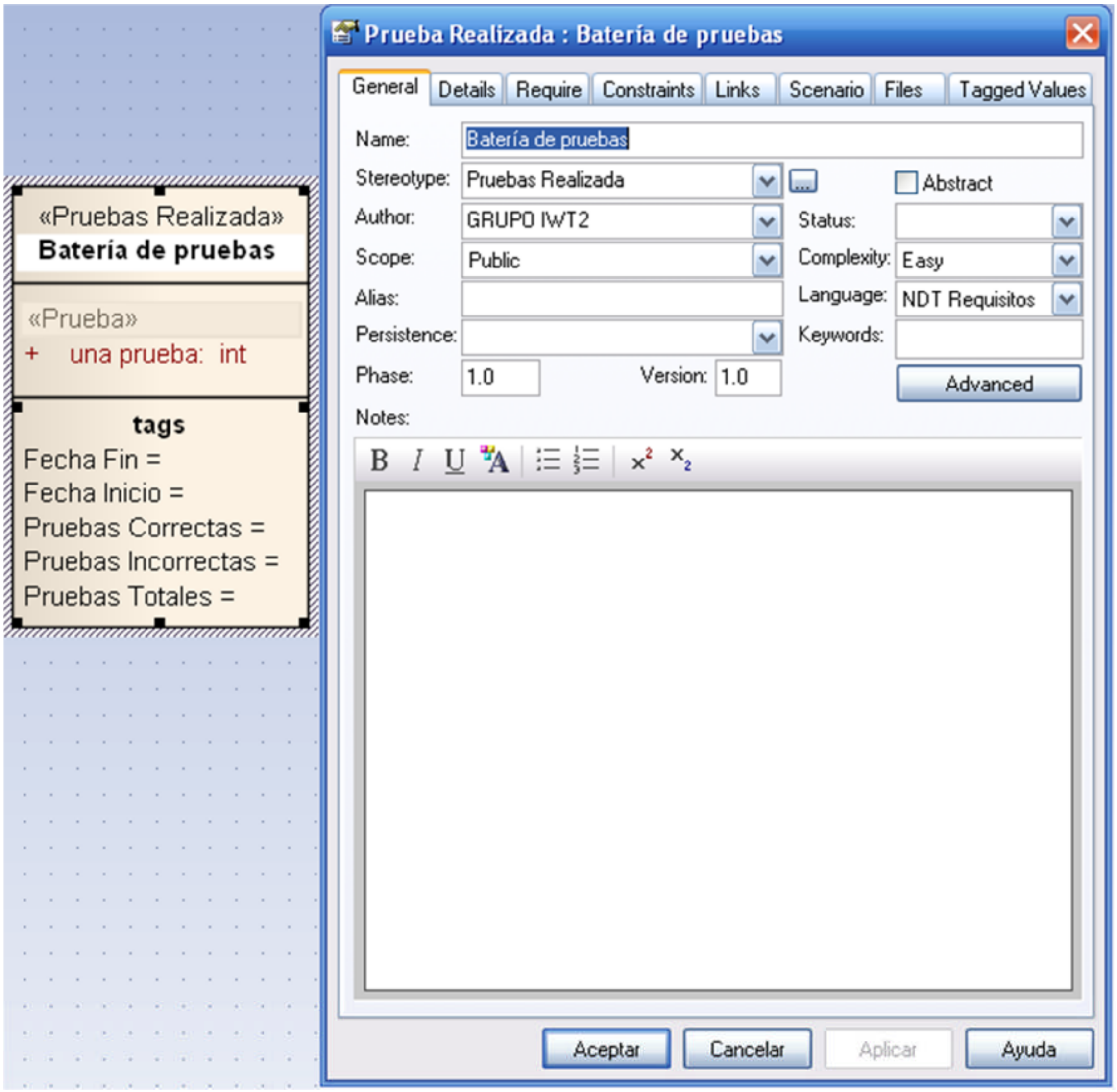

Figure 99. Standard properties of a battery of tests

Tagged values of all tests are identical, so are the same as shown in Figure 99.

Test Performed in the artifact there are a number of fields that are mandatory for the description of the battery of tests is considered correct.

- **Name** (campo Name): Each test system must have a descriptive name.
- **Author** (campo Author): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (campo Notes): This field describes the depth of detail that the author deems appropriate, the artifact being treated.
- **Start Date** (Tagged value): Enter the date at which begins to run the battery of tests.
- **End Date** (tagged value): Indicate the date that ends the battery of tests.
- **Total Tests** (Tagged value): Indicate the number of tests have been completed successfully.
- **Incorrect Tests** (Tagged value): Indicate the number of tests that have not been completed successfully.
- **Total Tests** (Tagged value): Total number of tests that have been implemented in the battery of tests.

- **Attributes** (Within the Details tab, Attributes button): The way to add attributes is similar to how you do on the Services. From the toolbox drag a test attribute to the artifact. In the name shall indicate the name of the test. In addition, each attribute has two more tagged values described below.
- **Date** (Tagged value of Attributes)
- **Result** (Tagged value of Attributes)

There are also other fields are optional, but recommended that the form is completed for a better definition of functional requirements. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field manage the different versions of the appliance. This field is meaningless when it comes to large systems where version management is a critical task.

In addition, batteries of tests must be linked to the project participants that made by the **Realize** link (shown in the toolbox of Figure 98), taking as origin the participant (to be dragged from the package of participants) and target the battery of tests performed. After creating a line that models the connector if you want to edit its properties, would have to double click on the connector, through the display properties described in paragraph 4.3 of this guide.

15. Maintenance of the system

The maintenance process, in contrast to other processes running in development is a process that begins when the project moved into production and ends when the system falls into disuse. The structure of system maintenance document shown in Figure 100.

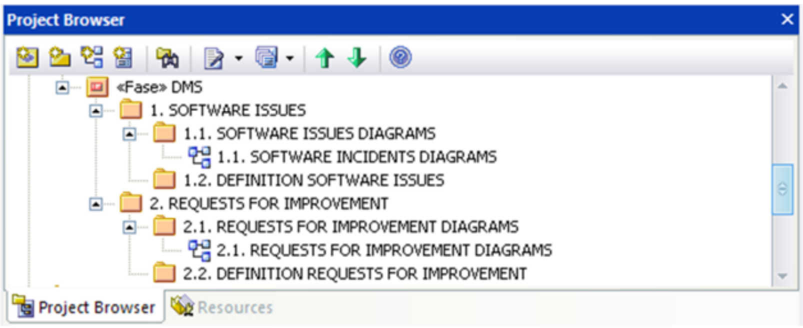

Figure 100. Structure of DMS

This phase consists of defining two types of artifacts, software issues and requests for improvement.

Table 44. Maintenance Nomenclature

	Name
Software Issues	IS-XX.Name
Requests For Improvement	PM-XX. Name

The *Software Issues* are software errors detected during the execution of the system and involving the detection of an error that contradicts the set of requirements defined for the system.

The *Requests For Improvement* are proposals that the user can take to improve the system in terms of interface, navigation, functionality, etc…

The set of tools for the definition of maintenance artifacts shown below in Figure 101.

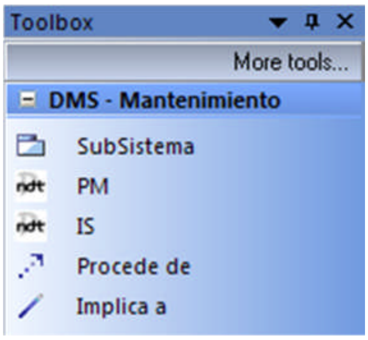

Figure 101. Toolbox

The standard properties are the same for both types of artifacts, which are observed in the Figure 102.

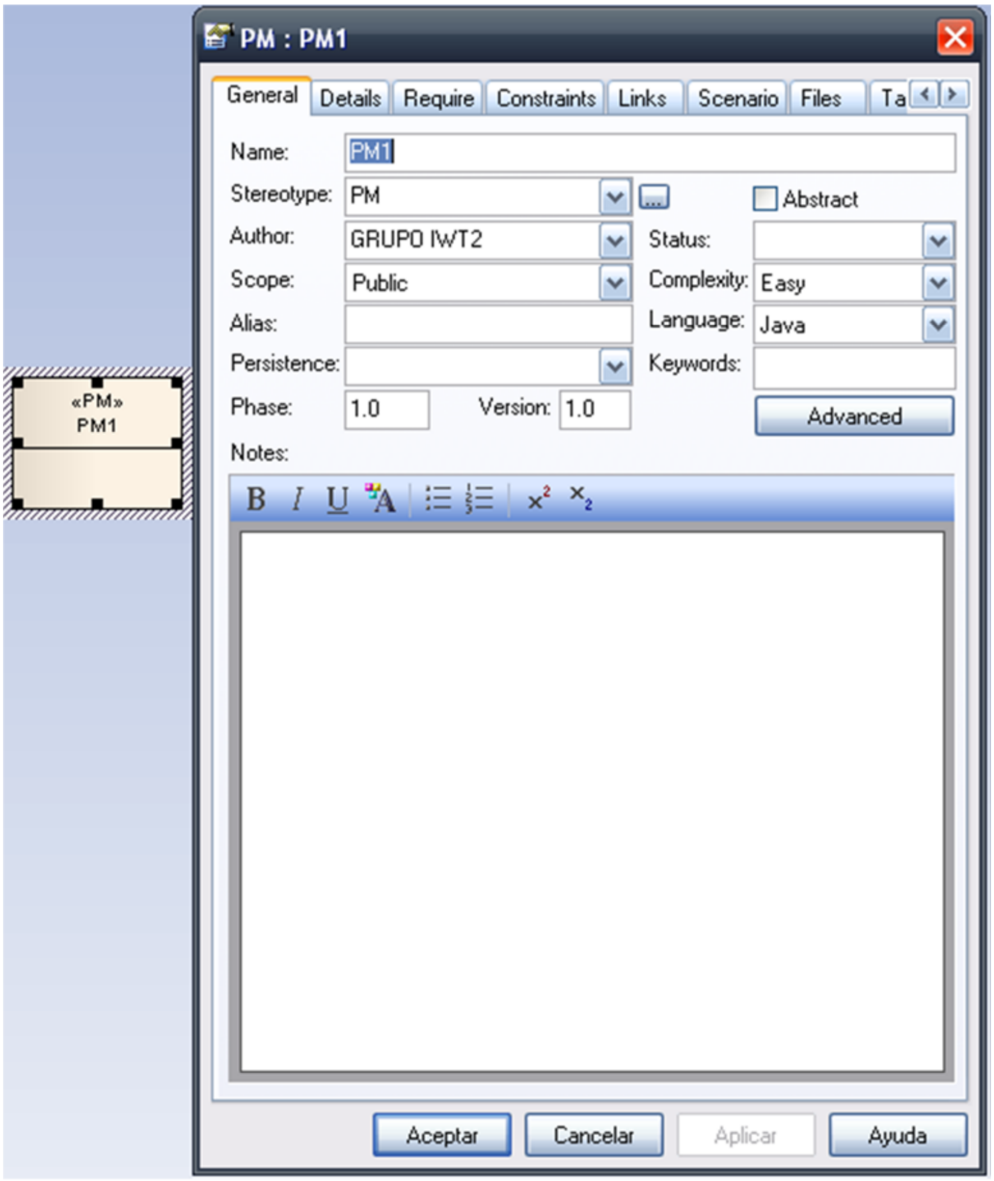

Figure 102. Standar properties

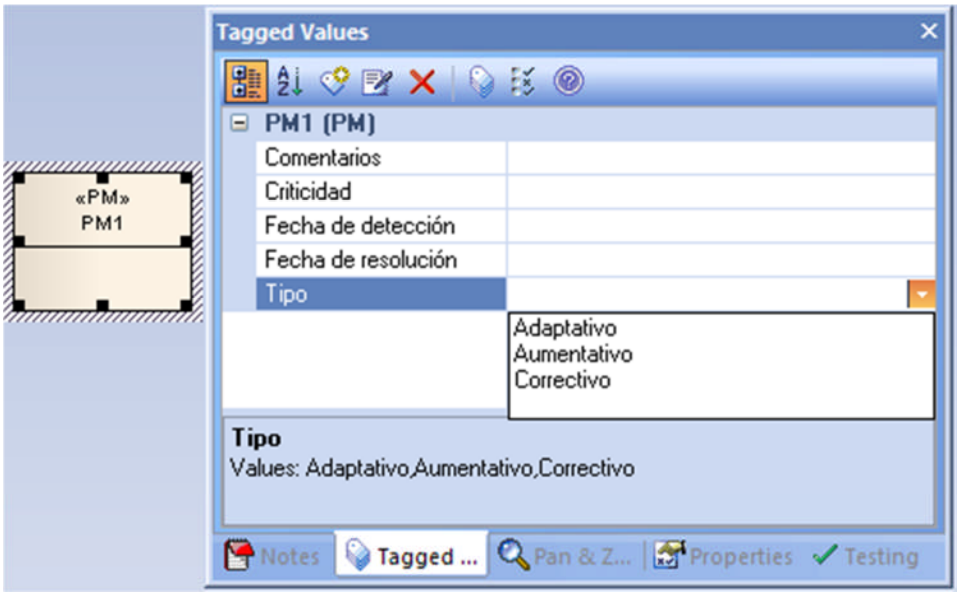

Figure 103. Tagged Values

In PM and IS artifacts there are a number of fields that are mandatory for the description of requests and incidents are considered correct. This series of fields are:

- **Name** (Name): Each menu should be classified with a code and a descriptive name. As shown in Table 44, the name of the enhancement requests must meet the following format PM-XX.Name, where XX is a two-digit number, or exceptionally, three figures, if there is a number high demands for improvement. In the case of software impact the name must meet the following format IS-XX.Name, where XX is a two-digit number, or exceptionally, three figures, if there is a high number of incidences software.
- **Author** (Author field): This field contains the name of the company or person in the company responsible for defining the artifact.
- **Description** (Field Notes): This field is described, with the depth of detail that the author deems appropriate, the artifact being treated.
- **Type** (tagged value): This field indicates the type of incident is being treated. In the case of software incidents, the type must necessarily be corrective. In the case of requests to improve the rate must be adjusted or increased.
- **Criticality** (tagged value): This value indicates the degree of criticality that has the resolution of the effect that models the artifact within the system. Must be chosen one of the predefined values.

There are also other fields are optional, but recommended that the form is completed to better define the menus. These are:

- **Status** (Status Field): This field contains the situation where the artifact is in the process of development. The value must be selected among the possible predefined shown.
- **Version** (Field Version): Through this field are managed by different versions of artifact. This field is meaningless when it comes to large systems where version management is a critical task.
- **Comments** (tagged value): This field allows the author intended to indicate any other information it deems appropriate.

- **Date of resolution** (tagged value): This field may indicate the date on which the incidence has been determined that the engine model.

All software issues or *Requests For Improvement* to define traceability relationships with the affected artifacts. It is up to the project coordinator to define whether such artifacts comply with the requirements, analysis or system design. To this end, the artifact can be connected to any other artifact using the links project **Implies to**, shown in Figure 101. To use this connector, simply click on it in the toolbox, click on the incidence of origin and drag to the destination artifact in the diagram. After creating a line that models the connector if you want to edit its properties, would have to double click on the connector, through the display properties described in paragraph 4.3 of this guide.

In addition, software issues or *Requests For Improvement* can be linked between them by links **Comes from**, shown in Figure 101, to indicate that an software issues is created to complement other software issues. To use this connector, simply click on it in the toolbox, click on the incidence of origin and drag to the incidence of destination in the diagram. After creating a line that models the connector if you want to edit its properties, would have to double click on the connector, through the display properties described in paragraph 4.3 of this guide.

Table 45. Rules Maintenance (IS, PM)

Maintenance Chart
You must have a maintenance-type diagram
All IS and PM must be contained in their respective diagrams
May link only comes from or involves
Maintenance Definition
The name of the artifact is IS-XX (X).Name or PM-XX (X).Name
The name of the artifact must use the CamelCase notation, starting with uppercase
The description is filled
No other IS or PM with the same number

16. Generate documentation

To facilitate the generation of a document for each of the main stages of the life cycle of a software project, NDT-Profile 2.X provides a set of templates designed in Enterprise Architect.

The life cycle stages for which we have this functionality are the requirements phase, analysis phase, design phase and testing of the system phase. For each of these phases, in NDT-Profile exists a package called DOCUMENTATION which is designed the master document.

Thanks to this set of templates, Enterprise Architect lets you completely automatically generate a structured document with all information collected at each stage of the life cycle mentioned above.

Below is shown the procedure to obtain the document associated with the requirements phase. For the remaining phases, the procedure is entirely analogous.

1 - Once you open the project created on the basis of NDT-Profile 2.X tool, open the logic diagram located in the package / DRS / 4. DOCUMENTATION /:

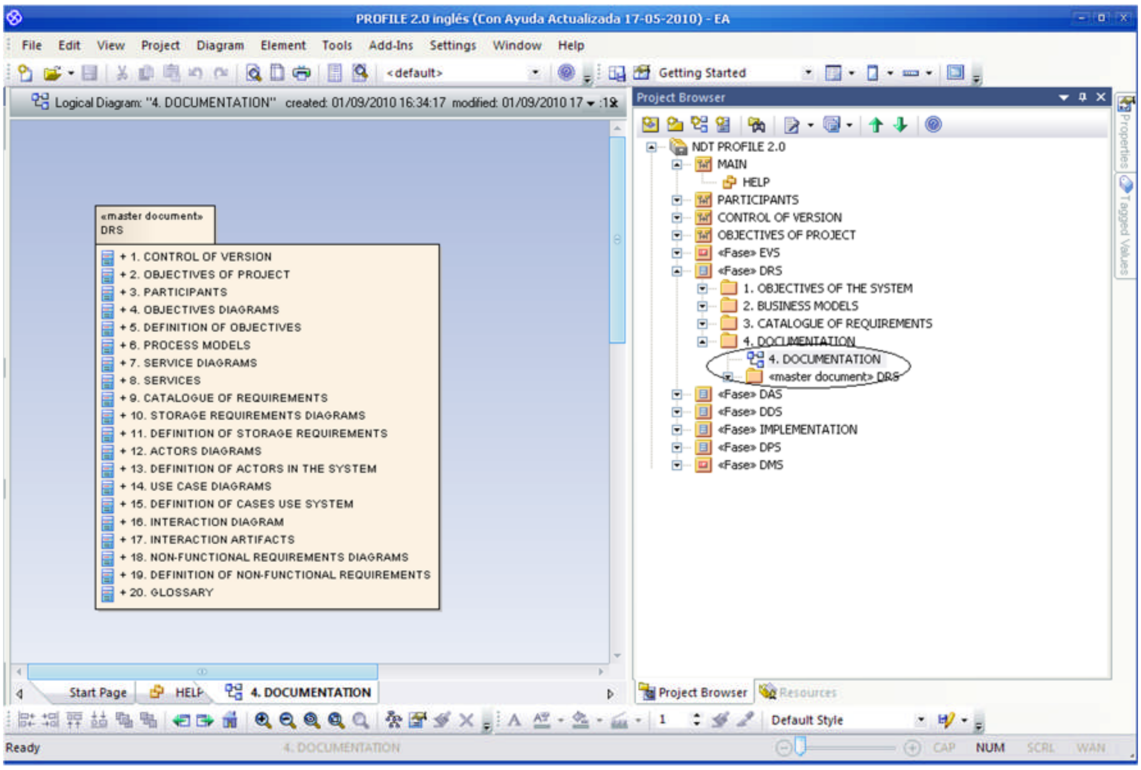

Figure 104: Master Document DRS

2 - Then, select the Master Document **Documentation DRS** displayed in the diagram above.

3 - We go to **Project> Documentation> Ritch Text Format (RTF) Report.** (Alternatively, you can press F8) to open the options window:

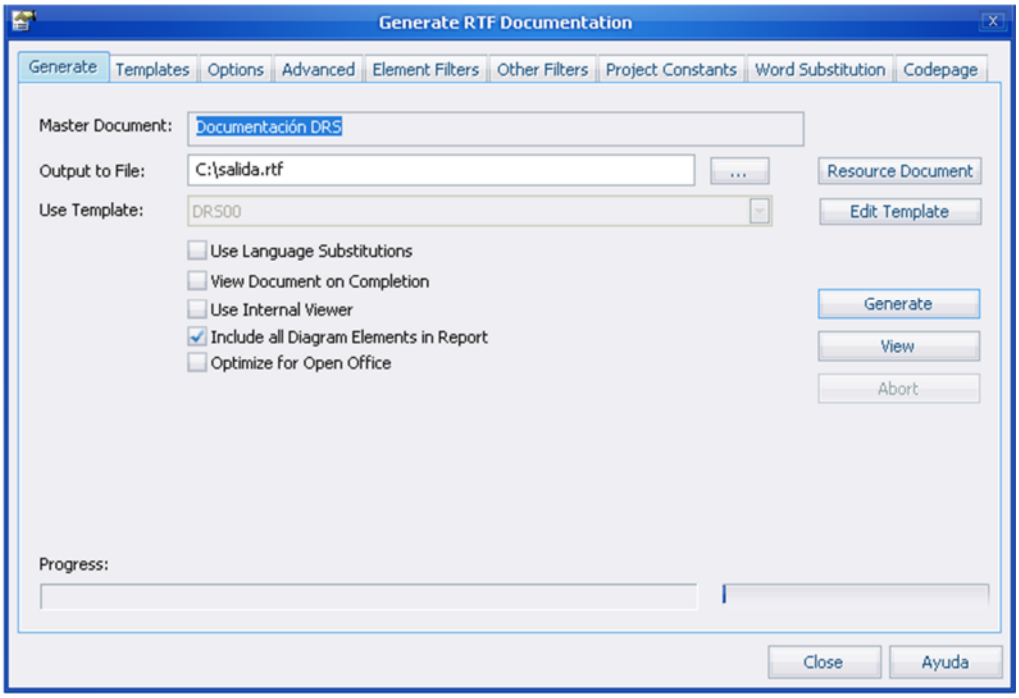

Figure 105. Window options in EA

4 - Indicate where to save the output document and click the **Generate** button.

17. Repair and Compact the EAP file

If you are using the EAP file and your project is not working on a data server, you need to take several steps to make our project is as safe as possible.

Because when we are working with the Enterprise Architect is generated much junk information that makes our file it is large, we have to regularly doing two operations that would:

- Repair .EAP File.
- Compact .EAP File

Both operations are in the menu Tools -> Manage. EAP File, as shown in Figure 106.

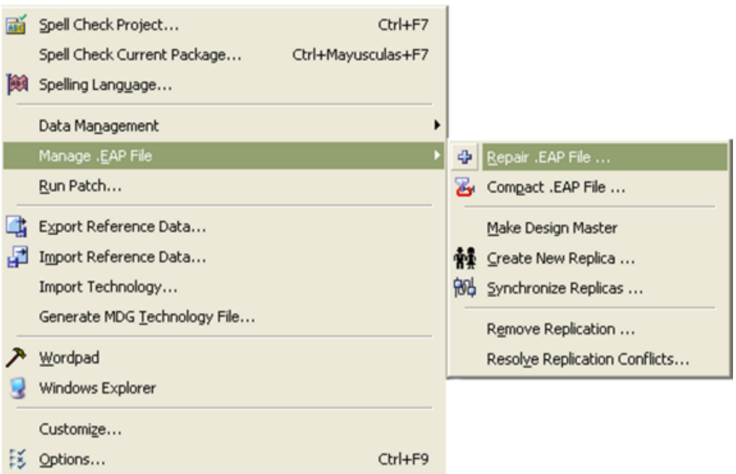

Figure 106. Compact EAP files

18. Glossary

The following are the terms used in this document.

Table 46. Glossary

Term	Description
Enterprise Architect	Enterprise Architect
CASE	Computer Aided Software Artifactering
NDT	Navigational Development Techniques

19. Bibliography and references

The following are the relevant references.

Table 47. Bibliography

Reference	Title	Código
NDT	http://www.iwt2.org	No