Spago4Q Manual

Authors Spago4QTeam

Index

Version

Version n°: **Release n°:**	0.1 1.0.0-RC1	**Data Version:**	January,18th 2008
Update description:	First version		

1 Document Goal

The document aim is to introduce the reader to the Spago4Q concepts by means of a full example based on the "Spago4Q demo".

The demo is available accessing from the http://www.spago4q.org/

Software and documentation are freely downloadable from the OW2 forge

(http://forge.objectweb.org/project/showfiles.php?group_id=301)

The document includes the following main chapters :

•**Conceptual and architectural overview**. Introduction to the core concepts of the Spago4Q free open source platform.

•**Functionalities overview**. An explanation of the Spago4Q portlets and the main activities of the various users typologies, considering the Spago4Q demo as an example.

•**In more depth**. How to use and implement Spago4Q to build a "Portal for Quality" in more detail.

1.1 References

For further information about Spago4Q platform refer to the documentation, available on the project site (http://www.spago4q.org/)

1.2 Help for lecture

Follows a short description of the most common views in Spago4Q.

1.2.1 Stylistic conventions

LITTLE CAPITALS	The LITTLE CAPITALS reference to the icon in a mask.
italics	The *italics* refer to fields of the masks.
<ITALIC CAPITALS>	In *<ITALIC CAPITALS>* the logical variables are suitable.
boldface	In **boldface** the main concepts.

1.2.2 SPECIAL SECTIONS

	Note
	Example
	Reference to other section or documents.
	In revision phase
	Future implementation. To be done.
	Advice for the reading of the section

2 Spago4Q overview

2.1 Conceptual overview

Spago4Q – SpagoBI for Quality – (www.spago4q.org) is a Free Open Source Software platform, released under GNU LGPL license, for maturity assessment, effectiveness of development software process and quality inspection of the released software: this goal is achieved by evaluating data and measures collected from the project management and development tools with non-invasive techniques.

Spago4Q supports companies and organizations both in the certification process and, more in general, in monitoring a formalized development process.

2.2 Architectural overview

Spago4Q architecture, obtained as a verticalization of SpagoBI (the Business Intelligence Free Platform www.spagobi.org) is designed in order to be easily adapted to complex organizational contexts. It integrates an advanced meta-model which makes Spago4Q fully independent from the adopted software development processes, infrastructure tools, measurement and assessment frameworks.

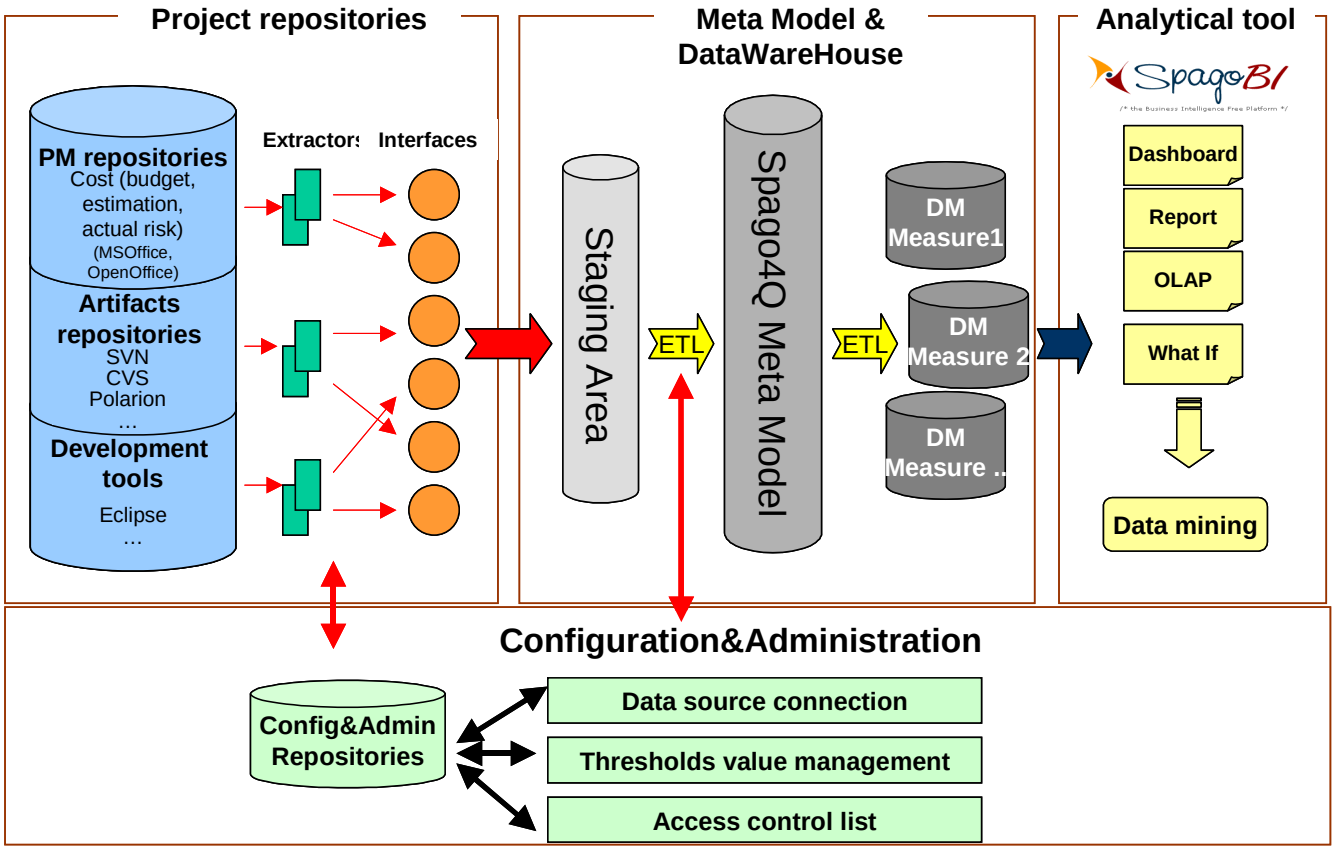

Figure 1 – Architectural overview

Figure depicts out Spago4Q architectural overview, the main components are following described.

Specialized extractors or *ETL* (Extract Transform Load) procedures extract data from the infrastructure tools.

Extractors are specialized components to collect data from different data sources.
They have the following characteristics:

- collect data from infrastructure tools and load "Interfaces" components;
- can load one or more "Interfaces" components;
- can be developed with different technologies ;
- XML is adopted to exchange data with "Interfaces" components;
- more extractors can exist to collect data from a specific tools;
- can apply rules to filter or transform data .

Release 1.0.0-RC1 includes extractors to collect data from Jira, SVN, MicrosoftExcel, Database.

Interfaces.. are components defining the format input data for ETL procedures to load datawarehouse. They create a decoupling between extractors and ETL procedures.

They have the following characteristics:

- for every area of measure (i.e. requirements, bugs, test) only one interface is defined in order to standardize format input data collected from different projects or tools.

Release 1.0.0-RC1 includes "Interfaces" for requirements, bugs, test and development measure areas.

ETL procedures load data into the *DWH-Spago4Q* datawarehouse.

A load procedure is developed for each "Interface" , it applies rules to filter or transform data.

Release 1.0.0-RC1 includes "ETL procedures" for requirements, bugs, test.

DWH-Spago4Q is the repository of all data collected from projects developed by an Organization.

Refer to chapter 2.3 for more details.

***SpagoBI* portal and analytical tools** analyze data and represent KPIs. In order to complete the assessment process, data inserted in the datawarehouse has to be analyzed by the Spago4Q analytical component. This module has been implemented as a verticalization of SpagoBI, in order to cover and satisfy the whole range of BI requirements, both in terms of analysis and data management, administration and security. Using SpagoBI platform to implement the Spago4Q analytical components makes easy to represent every KPI, metrics and the related thresholds as an instance of a particular analytical document type offered by SpagoBI itself (report, OLAP, dashboard, data mining, free inquiry, geo-referenced analysis). Spago4Q includes a configurable set of dashboards, reports and OLAP documents to monitor processes like: requirements management, bugs tracking, tests.

Configuration & Administration modules allow system configuration.
All the components described above can be properly configured through the *Configuration & Administration modules* that provide the following characteristics:

- extractors configuration allows the definition of connections to repositories and tools;
- access control list;
- thresholds values management;
- "library of measurements" management.

Note : the Spago4Q modular architecture and meta-model design guarantees extensibility towards others infrastructure tools and to further sets of activity measure areas.

2.3 DATAWAREHOUSE

Spago4Q can support Organizations that have to guarantee uniform levels of process and product quality across all their projects. In order to achieve the following goals:

- high adaptability to various organizational contexts;
- support for a complex system of evaluation;
- measurement process not bound to the adopted software development process and tools;
- automatic data collection from a set of tools;

The DWH_Spago4Q is designed on a meta-model definition followings the Meta-Object Facility (MOF) approach proposed by the Object Management Group (OMG).

2.3.1 META-MODEL

The Meta-model, defined by the contribution of the University of Milan - Department of InformationTechnology - SESAR (Software Engineering Software Architecture Research Lab) http://sesar.dti.unimi.it/ , represents the process(es) to be monitored, the measurements to be taken and the assessment frameworks to be used.

These three components have been designed independently, allowing to apply the same measurement framework to different processes, or to measure the same process according to different measurement frameworks. The interface between the process and the measurement modules is realized by the specification of measurable attributes of process activities and work products, needed by metrics. The interface between the measurement and the metrics modules is defined as a set of Key Performance Indicators (KPIs) based on measurements.

Computing process KPIs required the implementation of a series of specific extractors, that know which attributes to extract and where these attributes could be found in process and product definitions. Finally, all process metrics to be included in process reports are defined in terms of KPIs.

The meta-model is depicted in Fig. 2

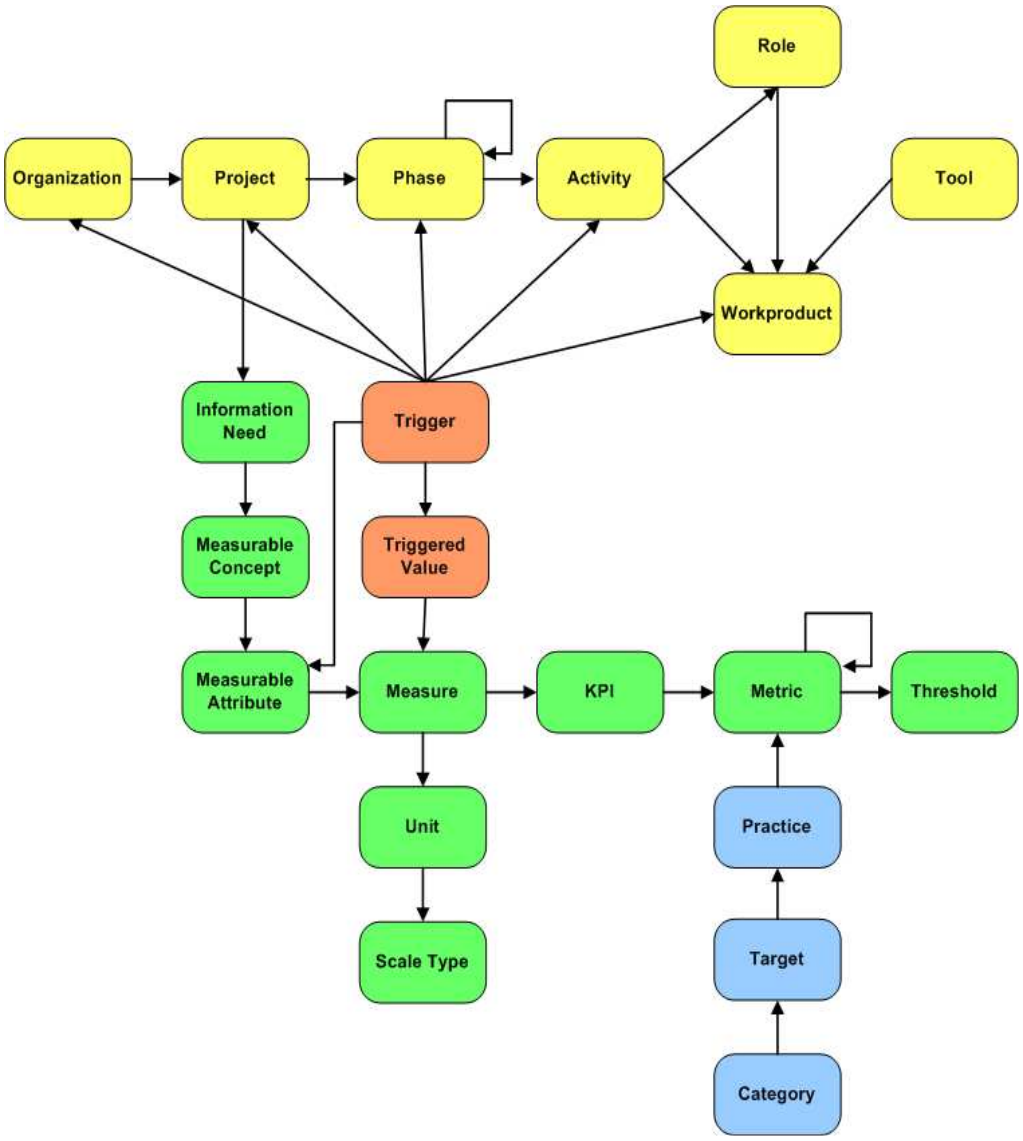

Figure 2 – Spago4Q Meta-model

We briefly describe the major meta-model modules: Process, Measurement, and Assessment.

Process module.
The top element is the node Organization, which describes the whole organization and allows enterprise-wide measurements. It gathers all the Project nodes, which indicate a single project of the organization. Each project is associated to a specific development process model (waterfall, Agile, ..).

Measurement module.
It is based on our reverse GQM (Goal-Question-Metric) approach and defines a generic framework exploitable to get measures from any development process.
The Information Need node is the container node that defines the need that drives all the measurement actions. For each information need there is a set of Measurable Concept nodes. These concepts drive the definition of the Measurable Attributes, which indicates the attributes to be measured in order to accomplish the analysis goals.

The Measure node defines the structure of data retrieved during a measurement campaign. This node is directly connected to attributes and supplies raw data to KPI and Metric. Each node is specified by the nodes Scale Type and Unit, defining, respectively, the unit of measurement used and the type of scale adopted (nominal, ordinal, and so forth).
The monitoring indicators are defined in terms of KPIs (Key Process Indicator) and Metrics. Finally, the Metric class is in relation with the Threshold entity specifying the threshold values for each metric when needed for qualitative evaluations.

Assessment Module
The module simply defines a general assessment framework as CMMI or ISO 9001:2000.
This module allows a simple classification in terms of Category, Target, and Practice. The Practice node is connected with the Metric class in the Measurement module, highlighting the concept that the evaluation of a set of metrics could assess the real appraisal of the practices.

2.3.2 DWH_Spago4Q

DWH_Spago4Q is an implementation of the meta-model.

The figure shows the datawarehouse high level schema.

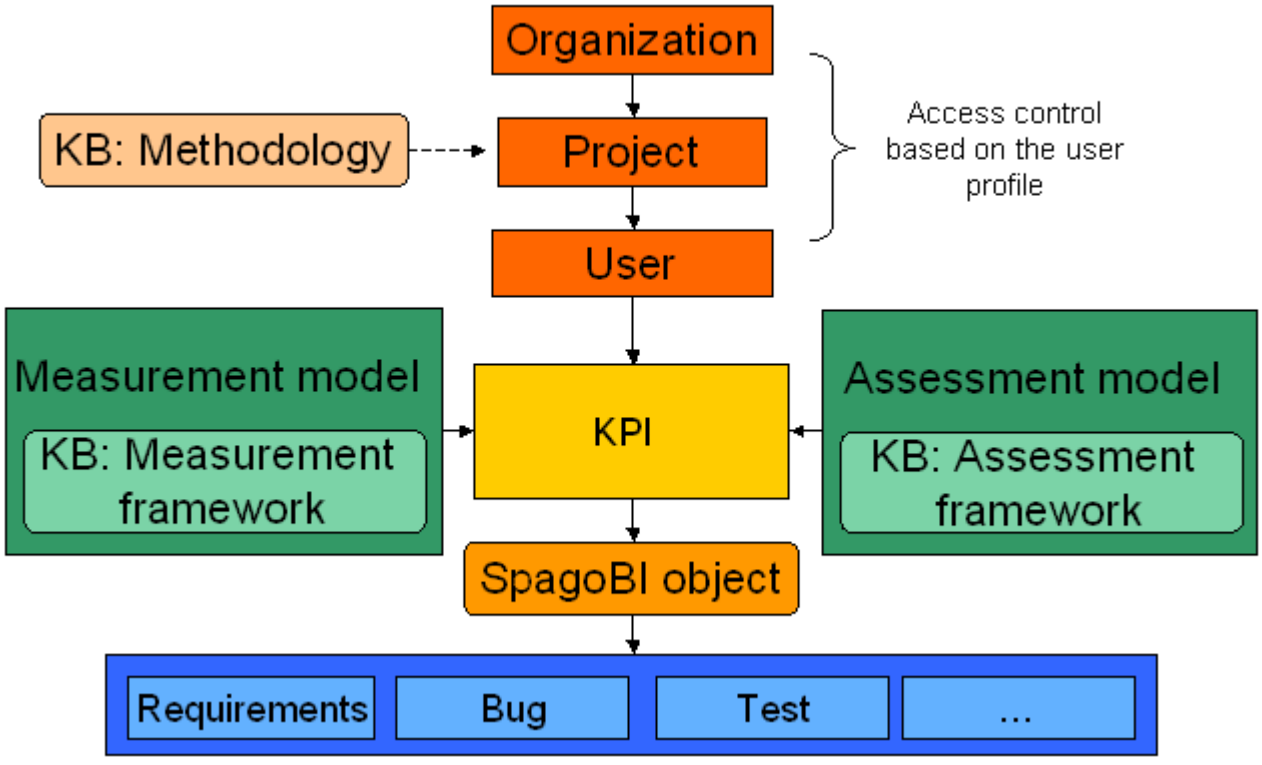

Figure 3 – Dwh-Spago4Q

The KPIs are linked to:

- Organizational structure (user/project/organization's areas). You can manage the user profile by role based criteria. KPIs and projects are assigned to the user by role in the Organization.
- measurement model;
- assessment model;
- SpagoBI analytical document that implements the algorithm defined for the KPI. The fact tables "Requirements, bugs and so on" contain the data collected from infrastructure tools.

Datawarehouse structure

The datawarehouse is modeled on the dimensional model concept.

According to this theory record data is stored in de-normalized tables named "Dimension tables"
Transactional data is stored in "Fact tables". This structure is named "star schema" .

Spago4Q implements the dimensional model theory applying the "snowflake" technique. This method requires to manage level of normalization on dimension tables. This normalization increases the number of Joins and introduces a potential performance worsening, but the configuration operations are more useful.

Fact Tables implemented in release 1.0.0-RC1 :Area Requirements, Area Bugs, Area Test.

Refer to chapter 4.6 "How to create a new area in DWH" and chapter 4.7 "How to load a scratch DWH"

3 Functionalities overview

Functionalities are described for user and administrator.

3.1 Recurring themes

The graphical interface adopts conventions described in this chapter.

3.1.1 Portlet Layout

Every user portlet points out some common characteristics:

- On the top, there is the title identifying the portlet meaning.
- On the right side of the title, some icons allow the access to the general functions acting on the portlet's content. The main functions are (where admitted):

	going back to the previous page without saving changes;	Every portlet
	creating a new element;	Every portlet
	switching from the list view to the tree view;	Document config.
	switching from the tree view to the list view;	Document config.
	saving information;	Details pages
	saving information and going back to the previous page;	Details pages
	testing before saving.	LOV details
	Return to main page	Details pages

- The '*' character identifies the required fields.

3.1.2 List and detailed View

One of the most common views in Spago4Q is a simple table showing a list of elements.

Common characteristics are:

- On the top, the title identifying the table meaning.
- The first row shows a label for each column displayed.
- The list can be divided into pages that can be turn over using the two arrows on the bottom row.

•The current page and the total number of pages are displayed in the middle of the bottom row.
•Every list has a detailed page showing and allowing to modify all the data about a single element.

Every list is alphabetically ordered on the first column's content (the label) and each row shows the essential data of an element, always identified by a unique label or title.

On the right side of every row, some icons drive the operativeness on the single element (row) of the list. The main possible functions are (where admitted):

• accessing the details page for the selected element (row).	Every list
• deleting the corresponding element (row);	Every list
• executing the corresponding element (row);	Analytical Doc. list only
• Selecting all.	Tree management
• Selecting element (row)	Every list
• View parameters	Every list
• View list values	Threshold list

A standard view of a list and detailed page follows.

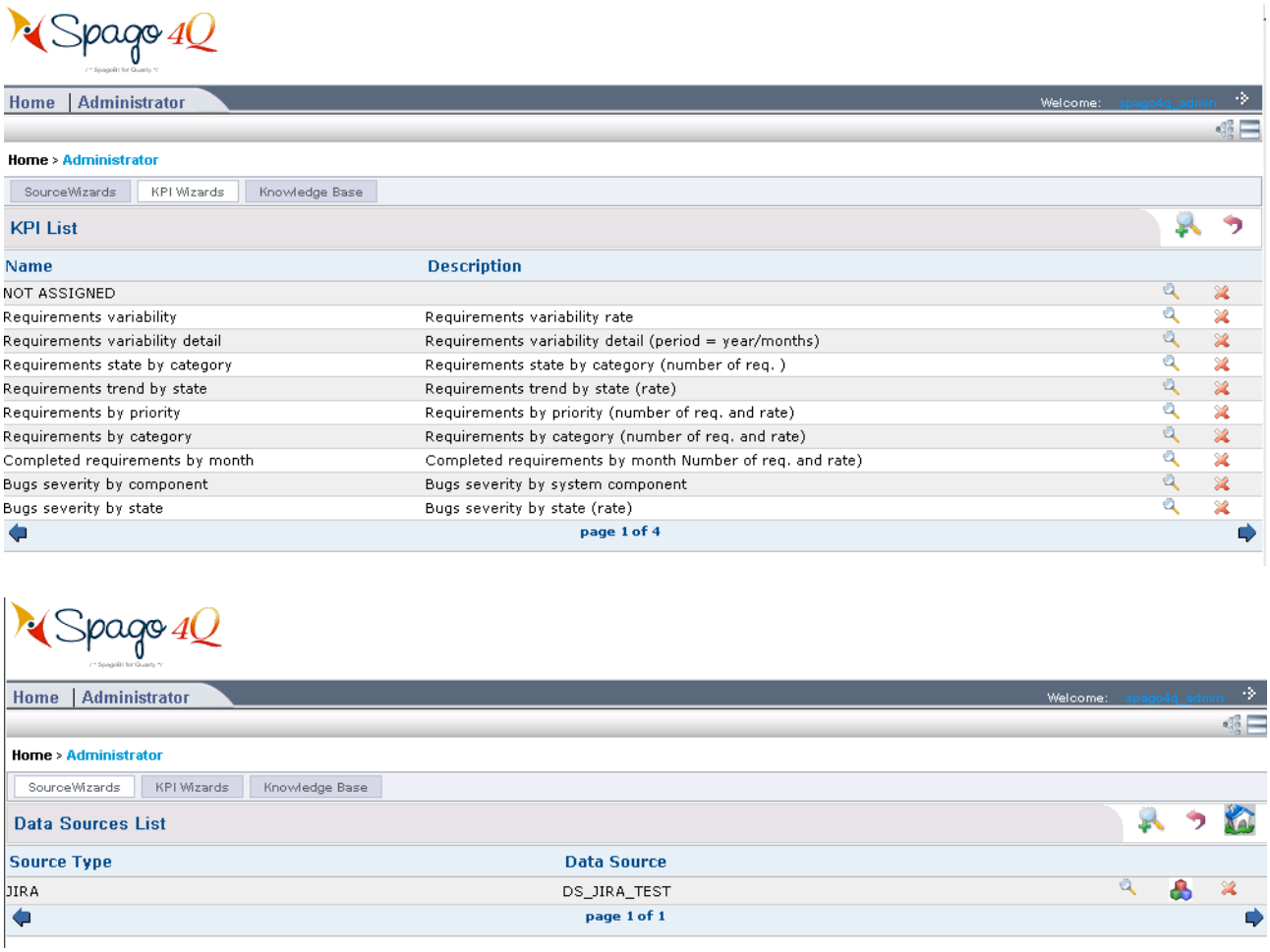

Figure 4 – List-detail examples

3.2 EXO PORTAL ADMINISTRATOR.

For further information about settings of ExoPortal platform refer to the documentation, available on the project site (http://forge.objectweb.org/projects/exoplatform)

3.3 SPAGOBI USERS

In this section are summarized the functionalities available to SpagoBI users: Administrator, Developer, Tester.

For further information about SpagoBI platform refer to the documentation, available on the project site www.spagobi.org).

3.3.1 BI ADMINISTRATORS

Settings

In SpagoBI the administrator (biadmin/biadmin user) has some portlets; one of them is the SbiSettings portlet which permits him to:

- register and configure each analytical engine inside the platform;
- synchronize SpagoBI roles with the portal roles;
- import/export documents.

Documents and tree management

The SpagoBI administrator (biadmin/biadmin user) has also the functionalities and documents administration portlet which permits him to perform the following operations:

- Managing the functional structure that classifies the analytical documents and configuring permissions on the functionalities;
- Maintaining the registered analytical documents;
- Schedule documents execution.

Notice that the administrator manages the **structural configuration** of the platform

3.3.2 DEVELOPER

The developer (bidev/bidev user) main tasks are:

- to define the possible presentation and the preloading way (LOV – list of values) for the parameters;
- to define the validation rules (CHECK) for the input value;
- to create the parameters (PARAMETER) and to set up their behaviour rules associating LOV and CHECK to the user's roles;
- to register and to configure analytical documents, referring to the used parameters.

3.3.3 TESTER

The tester (bitest/bitest user) main tasks are:

- validating the produced Analytical Document to simulate all its predefined roles;
- updating the Document state to release the documents that becomes available for the end-user.

3.4 SPAGO4Q ADMINISTRATOR

The Spago4Q administrator (spago4qadmin/spago4qadmin user) has the wizards which let him perform the following operations:

- Managing the KPI thresholds;
- Managing library of measurements to create or modify assessment framework models, measurement models, KPIs;
- Managing library of measurements to associate KPIs at a specific assessment framework model or measurement model;
- Managing set up data sources and extraction processes.

For further information about the steps to manage the library of measurements refer to the chapter 4.2

In the following it is explained how to use Spago 4Q wizards:

- KPI wizards;
- Source wizards;
- Knowledge base.

3.4.1 KPI WIZARDS

KPI Wizards functionalities are the following:

- KPI management
- Threshold management
- Associate KPIs to measurement model
- Associate KPIs to assessment model.
- Associate a threshold defined by a user to a specific KPI.

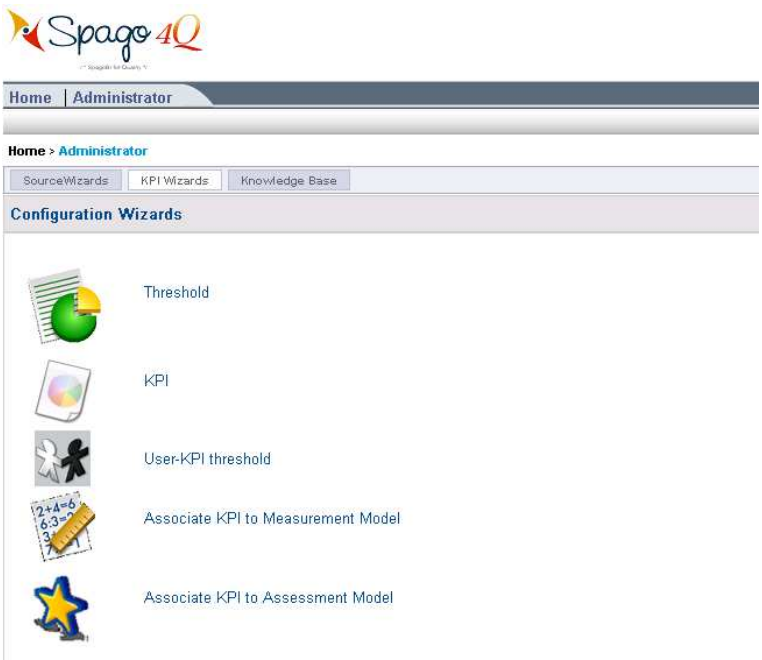

Figure 5 – KPI wizards

3.4.1.1 Thresholds

The system shows a list of all thresholds and lets the user insert new ones or update or delete the current ones. Thresholds that are referred from at least one KPI cannot be deleted. Each threshold is described by a name and a description field.
The user can choose a threshold and view the values associated, add new values and update or delete the current ones. A threshold value is characterised by a minimum and a maximum value that bound its range interval, a label and a colour.

The values associated to a specific threshold are an ordered list, in particular they are treated as a LIFO list, so that the user can delete only the last value inserted; this choice has been made to preserve a strict ordination among values.

Figure 6.a. shows the list view on all thresholds. Figure 6.b shows instead the detail of a particular value associated to a specific threshold.

You can associate a threshold to different KPIs

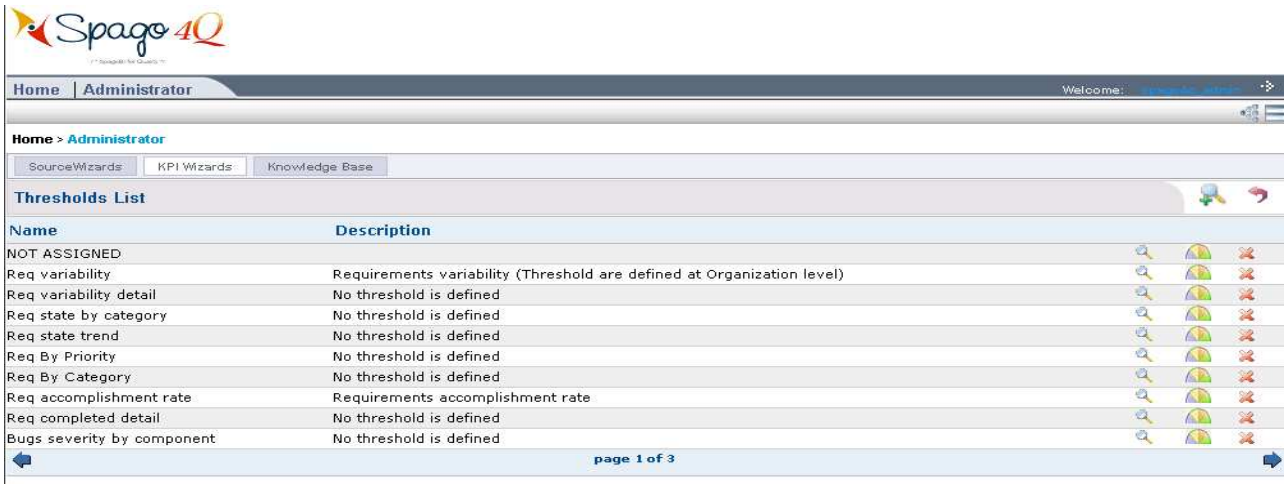

Figure 6.a – List-thresholds examples

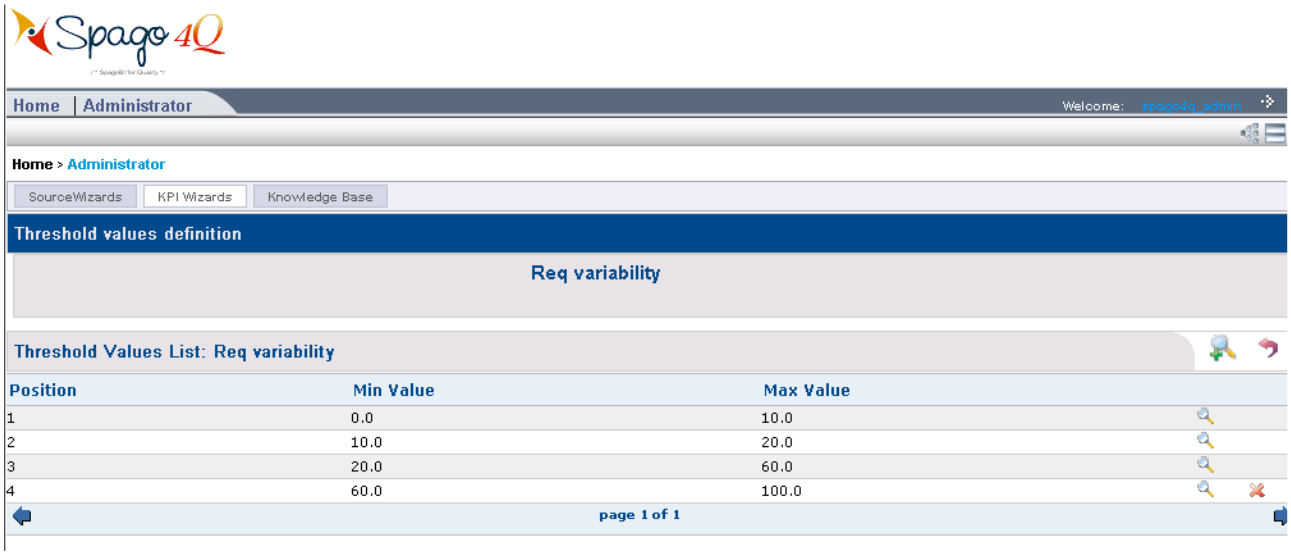

Figure 6.b – List-threshold values examples

3.4.1.2 KPIs

The system shows a list of all KPIs and lets the user insert new ones and update or delete the current ones. KPIs which are already associated to an assessment or a measurement model cannot be deleted. A KPI is characterised mainly by a name and a description; by viewing the detail it is possible also to view and set up a code and a metric. The user can also associate to the KPI a SpagoBI analytical document, from the ones available in the SpagoBI document database, and a threshold.

KPI wizard lets you define a metric as text description. Usually the SpagoBI analytical document implements the algorithm defined for the KPI. In some cases etl procedures implement the algorithm and store the result in an aggregate metric table. This values are used by the SpagoBI analytical document to display the KPI trend in a period.

In the datawarehouse metrics or aggregate metrics are managed similarly.

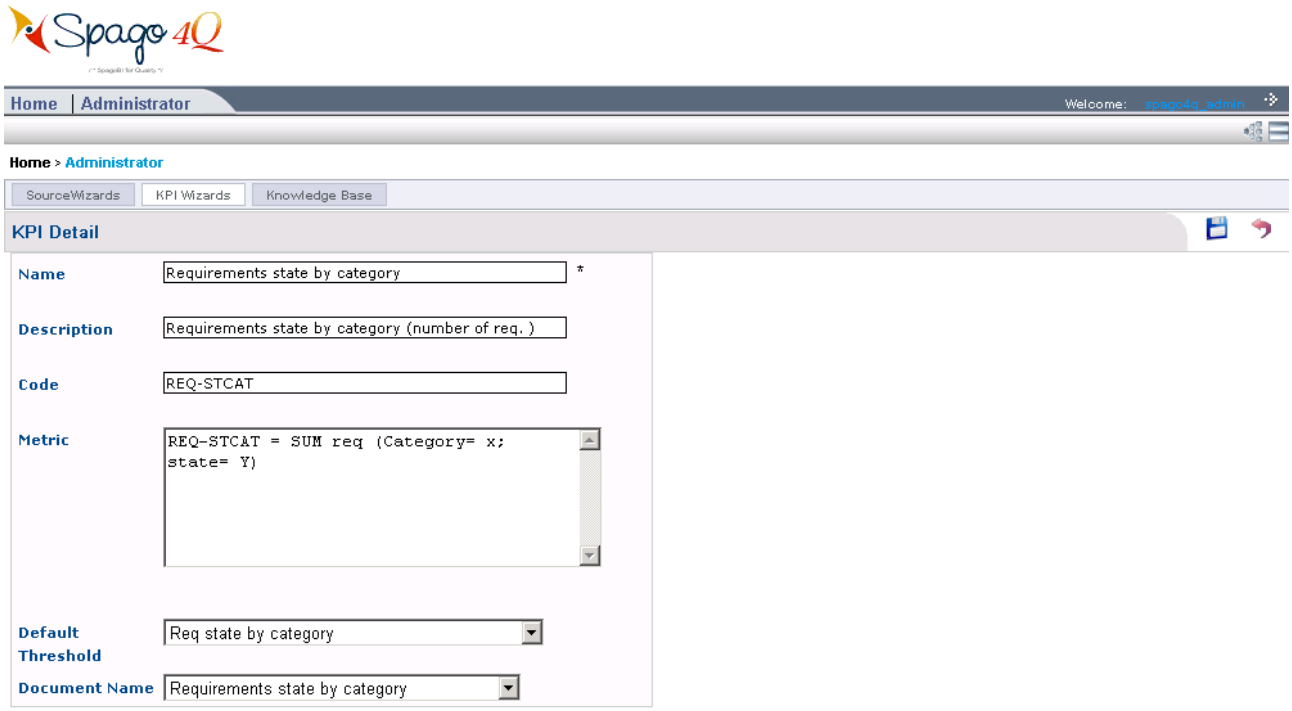

Figure 7 – KPI wizard examples

3.4.1.3 Associate User to KPI threshold

Spago4Q Admin can create specific thresholds for every user, then he can associate the user to KPIs.

The system shows a list of all the pairs <user-KPI> with a threshold assigned and lets the admin-user choose a different one.

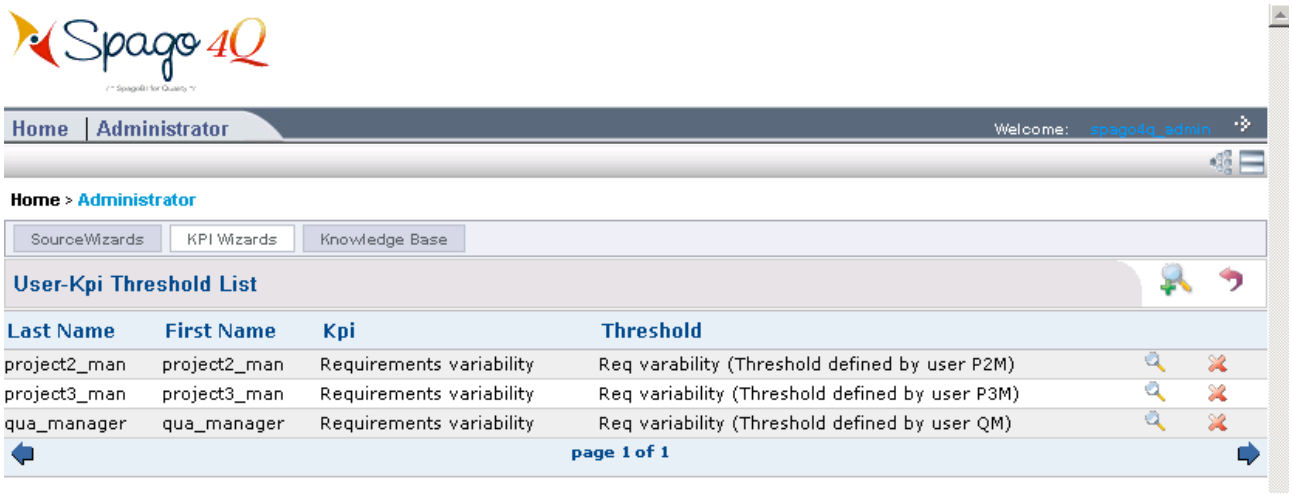

Figure 8.a – List user's thresholds examples

By Analysis by Role or Model Navigator functionalities the user executes the analytical document choosing the threshold type to apply.
DEFAULT value is the value defined at Organization level
USER value is the value defined by the user.

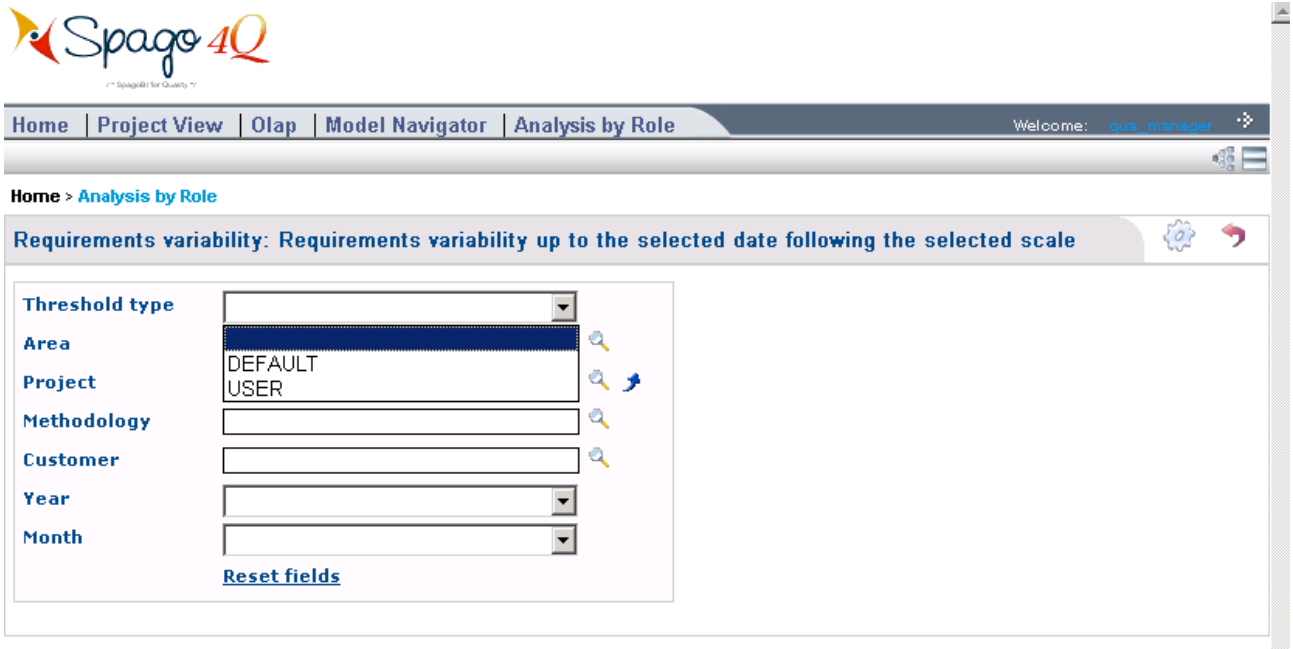

Figure 8.b – Choose threshold type

3.4.1.4 Associate KPI to Measurement Model

The system shows a list of all Measurement Models and lets the user insert new ones or update or delete the current ones. By selecting a specific measurement model the user can navigate with a tree view its structure, and change it. A measurement model is in fact composed by a hierarchy of levels and by elements associated to each level: an element defined at an i level is

associated to only one element at level i-1 and zero or more elements at the i+1 level. Figure 9 shows the levels involved in a measurement model.

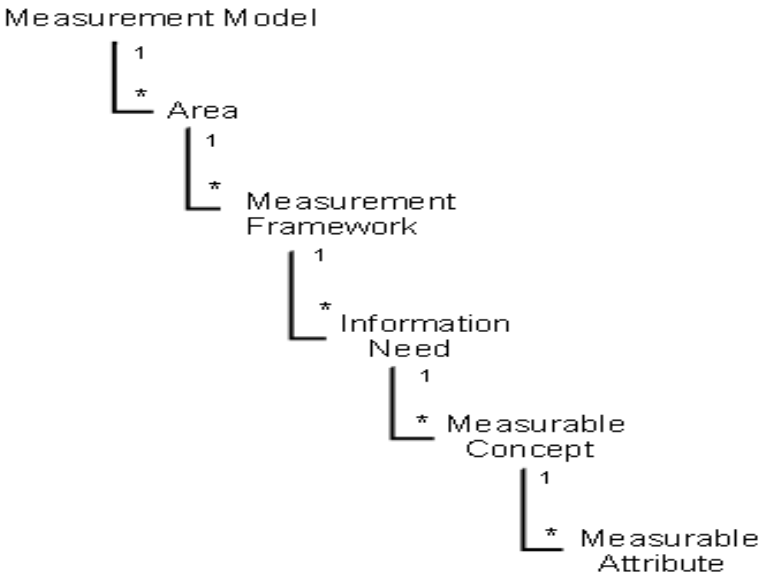

Figure 9 Measurement Model Hierarchy

By selecting a node at *i* level on the tree and by left-clicking on it the user can:

1) Add a node at the level i+1 among the available elements, that are the elements associated to the selected node (for example the measurable concept of a specific information need). If there are no more elements available, for example because all have been already associated, the system displays an information message.
2) View the detail
3) Delete the node and the sub-tree of which it is the root.

The user can also associate a KPIs to elements at the three lower levels, that is KPIs can be associated to information need, measurable concept and measurable attribute. To set this association the system shows a list of all KPIs.

Figure 10: Measurement Model Tree Structure

Figures 10 and 11 show the detail of a node of a tree, that represent a measurable concept.

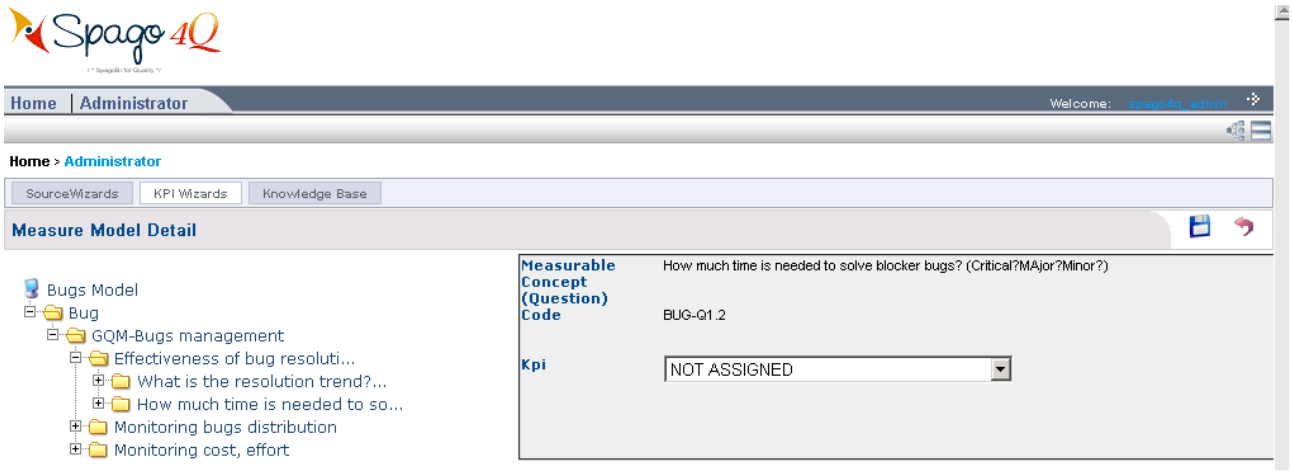

Figure 11: Measurement Model Tree Structure

3.4.1.5 Associate KPI to Assessment Models

For what concerns Assessment Models, KPIs are treated in the same way as described in the previous chapter for the Measurement Model. Figure 12 shows the hierarchy of an Assessment Model.

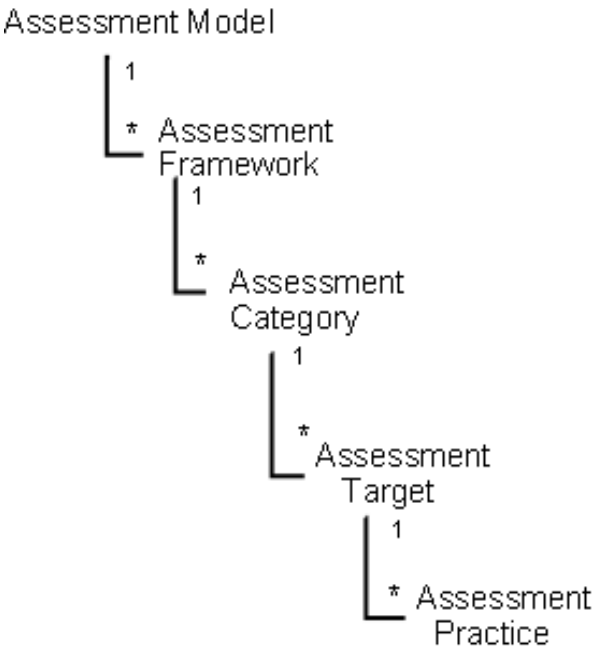

Figure 12: Assessment Model Hierarchy

3.4.2 SOURCE WIZARDS

Source Wizards functionalities are the following:

- Data source management.
- Extraction process management.

3.4.2.1 Data Source

The wizard lets the user choose a source type among the ones defined in the system . It shows therefore a list of all the data sources associated to the selected type and lets the user insert new ones or update or delete the current ones. Each Data Source is characterised by a name and a description.

By selecting one specific data source the user can view its associated parameters and change their values. The user can also add new parameters, and corresponding values, among the ones provided for the particular source type previously chosen, as shown in figure 13.

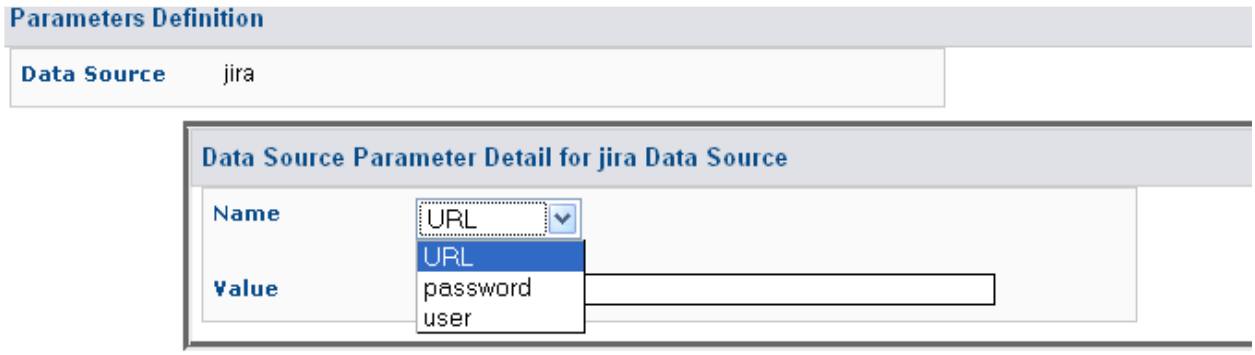

Figure 13: Data Source Parameters Definition

3.4.2.2 Extraction Processes

The wizard shows a list of all extraction processes, each one characterised by a name, a description and an interface. The user can create a new extraction process, creation that is composed in three steps, in which the user:

- selects a source type
- selects an interface among the ones associated with the chosen source type
- defines the name and in case, a brief description.

Of course the user can also update or delete current extraction processes.

By selecting a specific extraction process the user can view all operations associated, inserting new ones or updating or deleting current ones. The insert of a new operation is composed in two steps, in which the user first chooses a data source among the ones associated to the source type of the process (defined by the interface), as a second step chooses its name. Of course the user can also update or delete current operations.

The interface associated to an extraction process defines which parameters and fields can be associated to its operation. Selecting a specific operation the user can view its associated parameters and fields and change their values, add new ones among those provided by the

interface or delete them. Figure 3.2.a shows the list of all operations associated to a particular extraction process.

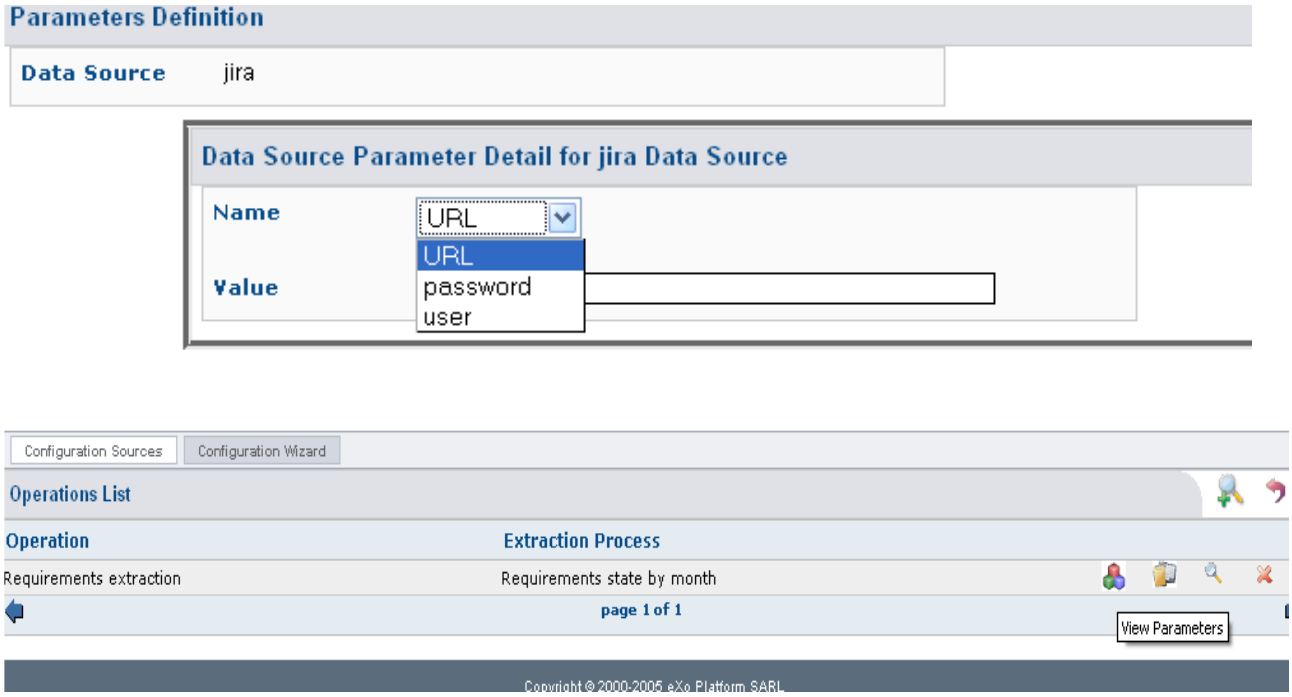

Figure 14: Operations list

3.4.3 KNOWLEDGE BASE

Knowledge base Wizards functionalities are the following:

- Assessment framework management.
- Measurement framework management.

3.4.3.1 Assessment Framework

The system shows a list of all assessment frameworks defined and lets the user create new ones and update or delete existing ones. For each one, the user can navigate, with a tree view, the framework structure; he can then associate to a framework one or more assessment categories, to each assessment category one or more assessment targets, and to each assessment target one or more assessment practices.

3.4.3.2 Measurement Framework

The system shows a list of all measurement frameworks defined and lets the user create new ones and update or delete existing ones. For each one, the user can navigate, with a tree view, the framework structure; he can then associate to a framework one or more information needs, to each information need one or more measurable concepts, and to each measurable concept one or more measurable attributes.

3.5 End-user

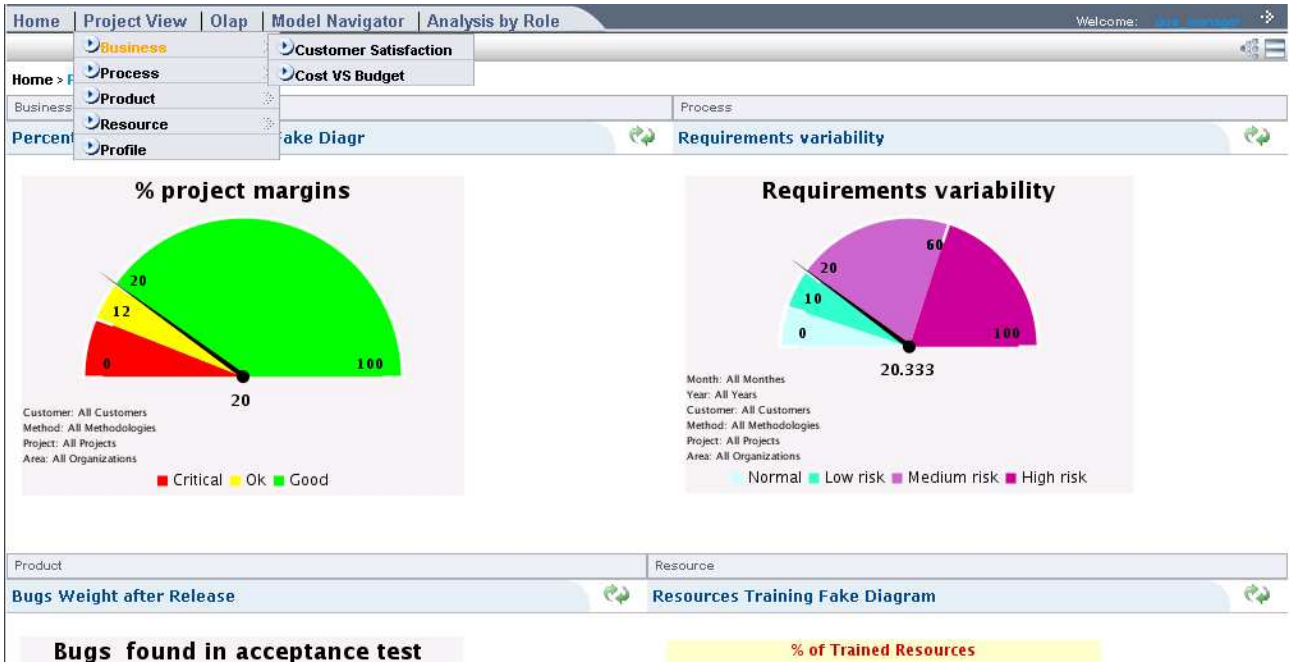

Figure 15: Project Business view examples.

"Project View" is a default view of the KPIs.

The aim of "Project View" is to display the more interesting and frequently used KPIs , in order to simplify the access to them.
In the next releases it is planned to develop a new functionality called "MY View", that will let the user choose which KPIs to displayed in his page.

Click on "Project View" to display a page containing four KPIs, one for each area: Business, Process, Product, Resource. In the example these KPIs are assumed to be the most important for the user.

Click on "Profile" under "Project View" to list the projects and areas under the responsibility of the currently logged user

Click on "Business", "Process", "Product", "Resource" and their sub-pages to browse all KPIs. All KPIs are calculated from the data concerning all projects and areas listed in the user's "Profile" page.

Click on "Model Navigator" to display the relationship between KPIs and an assessment framework (i.e. CMMI) or another customized measurement model (these associations are managed by Spago4Q administrator).

Click on "Analysis by Role" to display all the analytical documents available to the logged user. (these associations are managed by SpagoBI administrator)

4 In more depth

4.1 How to Implement Spago4Q

Scenario 1

Needs
Create a new KPI and represents it in a SpagoBI analytical document.

Assumptions
Measure attributes needed to calculate KPIs are already available in the DWH.

Step1 SPAGO4Q ADMIN uses KPI functionalities in KPI-Wizards to define the KPIs in terms of : name, description, code, metrics (algorithm description)

Step2 SPAGO4Q ADMIN uses Threshold functionalities in KPI-Wizards to create thresholds.

Step3 Developer SpagoBI uses SPAGOBI environment to implement the analytical document that represent the KPI.

Step4 SPAGO4Q ADMIN uses KPI functionalities in KPI-Wizards to associate the KPI defined in step1 to the threshold (step2) and the name of the SpagoBI analytical document (step3). Figure 16.

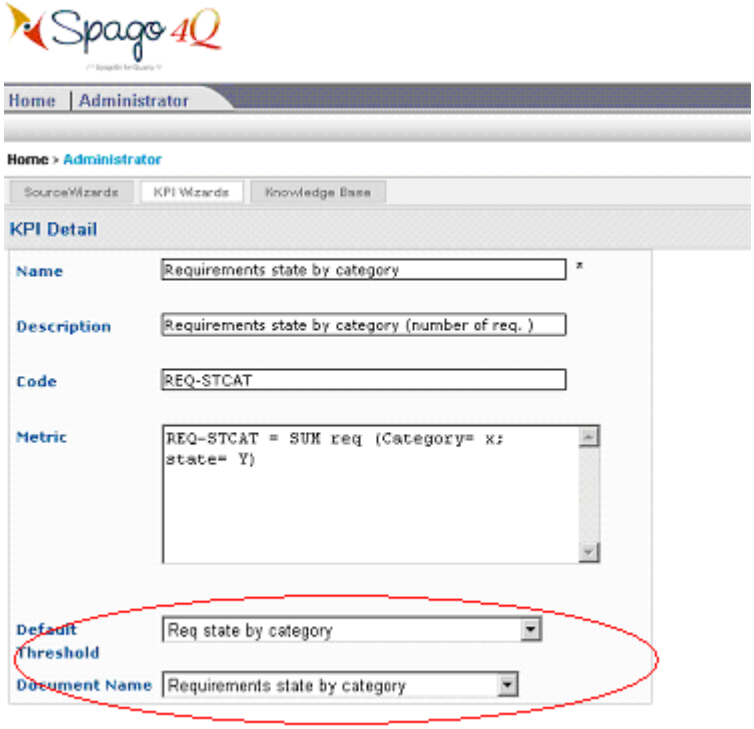

Figure 16: KPI wizard examples.

Step5 SPAGO4Q ADMIN uses "Associate KPI to measurement model" or "Associate KPI to assessment model" functionalities in KPI-Wizards to create the association between KPIs and models.
If needed Knowledge Base Wizard allows to create a new measurement or assessment model.

Step 6 SPAGOBI ADMIN manages analytical documents organization and security policy.

Scenario 2

Needs
Create a new KPI and represent it in a SpagoBI analytical document.

Assumptions
Measure attributes needed to calculate KPIs are not available in the DWH, so you have to implement a new fact table in DWH, new extractors and ETL procedures.

Step1 Refer to the chapter 4.6 about how to create a new area in dwh.
Step2 Refer to the chapter 4.4 about how to create a new extractor.
Step3 Refer to the chapter 4.5 about how to create a new ETL.
Step4 You need to execute all the steps described in scenario 1.

4.2 How to admin Portal

Portlets are autonomous and independent application windows. They are freely usable inside portal contexts, supporting the JSR 168 specification, by means of a simple configuration. No development is necessary.

Every function in Spago4Q runs in portlets included into a corporate portal.

The portlets organization into the portal is realized by the Portal Administrator.

Spago4Q releases specialized portlets according to the different user typologies (administrator, developer, tester, end-user).
Each user is assigned to a specific typology by the Portal Administrator.

For a better understanding of the user typologies refer to the analytical Document life-cycle section.

4.2.1 USER DEFINITION AND ROLES MANAGEMENT

For further information about settings of ExoPortal platform refer to the documentation, available on the project site www.spagobi.org .

4.2.2 PORTAL DEFINITION

For further information about settings of ExoPortal platform refer to the documentation, available on the project site www.spagobi.org .

4.2.3 USERS AND ROLES IN SPAGO4Q DEMO

Every user is characterized by one or more functional roles.
SpagoBI manages users by their functional roles in order to regulate:

1. the analytical documents visibility;
2. the visibility of the data shown by documents;
3. the behaviour rules of their parameters and the filters.

4.2.3.1 Groups/Users structure

The fundamental tree structure of the prototype refers to the first level group “Roles” which is reported below with the indication of the users contained in each group (with their Code Number and user_id)

Roles/
- Users: All users
- **General Manager**
  - Users: general_man Code:GM000 id:2
- **Quality Manager**
  - Users: qua_manager Code: QM000 id:1
- **Area Manager**
  - Users: ac1_man Code:A1000 id:11
    ac2_man Code:A2000 id:10
- **Project Manager**
  - Users: project1_man Code:A1P100 id: 9
    project2_man Code:A1P200 id: 8
    project3_man Code: A1P300 id: 7
    project4_man Code: A2P400 id: 6
- **Project Team**
  - Users: project1_team1 Code: A1P101 id: 5
    project1_team2 Code: A1P102 id: 4
- **BI Management**
  - **Spago4Q Administrator**
    - Users: spago4q_admin Code:SPAGO4Q id:13
  - **BI Administrator**
    - Users: biadmin Code:BIADMIN id:12
  - **BI Developer**
    - Users: bidev Code:BIADMIN id:12
  - **BI Tester**
    - Users: bitest Code:BIADMIN id:12

4.2.3.2 Community structure

A community is associated to every group of the tree structure "Roles". This leads to a particular navigation for every community as it is reported below:

Roles/
- Community: comm_share
- **General Manager**
  - Community: comm_gen_man
- **Quality Manager**
  - Community: comm_qua_man
- **Area Manager**
  - Community: comm_area_man
- **Project Manager**
  - Community: comm_prj_man
- **Project Team**
  - Community: comm_prj_team
- **BI Management**
  - **Spago4Q Administrator**
    - Community: comm_spago4q_admin
  - **BI Administrator**
    - Community: comm_bi_admin
  - **BI Developer**
    - Community: comm_bi_dev
  - **BI Tester**
    - Community: comm_bi_test

4.2.3.3 Community "comm_share"

All users belong to the group "Roles", to which the community "comm_share" is associated. This community describes the navigation structure common to all users. Actually it is composed by 2 nodes:

- Model Navigator: it contains the navigation structures which permit to search KPIs through the measurement and assessment frameworks
- Analysis by Role: it contains a portlet that allows every user to see its proper analytical documents (filtered through the visibility rules of SpagoBI)

This navigation could include other useful nodes in the future.

4.2.3.4 Others communities

The remaining communities are specialized for the group to which they refer. Actually, all of them contain the node "Project View" which gives an overview of the most important KPIs divided by Business (only visible to general, quality and area managers), Process, Product and Resource and the node "Olap" which contains an Olap for each fact table(in this case Requirements, Bugs, Tests).

4.3 How to develop analytical document

4.3.1 Analytical Document life-cycle

Every SpagoBI document usually follows a three steps life-cycle:

1.**Development**: this is the proper state of every document that has to be developed, corrected, modified or improved, and, therefore, it is the initial state of every new document;
2.**Test**: it is the state of a document which has to be tested in order to check if it works correctly returning the requested result for each possible configuration;
3.**Released**: this is the state of a document that has been properly developed and tested and can be employed by the final user.

Moreover, a 4th state (**Suspended**) can be assigned to a document that will not be used any more.

Referring to this life-cycle, SpagoBI users can have a specific function which is assigned by the portal administrator.
Users can be classified in 4 different typologies:
1.**Administrator**: he deals with configuration and security aspects.
2.**Developer**: this type of user can create or modify documents;
3.**Tester**: he takes the responsibility to verify the formal correctness of the registered documents and if they fulfil the requirements.
4.**User**: he can use all the business objects in a 'released' state, according to his role and with the modalities previously defined in the parameters configuration.

The *User* is characterized by his functional roles, which regulates:
- the analytical documents visibility;
- the visibility of the data shown by documents;
- the behaviour rules of their parameters and the filters.

It is very important to notice that administrators, developers and testers are also users and, therefore, they can act as specialised users with additional functions.

Every user will access a specialized main page that will contain specific tools.

When completed his own phase, a Developer can update the document state to Test, while a Tester, referring to test results, can change it to Development or to Released.
The administrator is the only one who can modify a document state without any constraints allowing extraordinary maintenance of the documents.
Notice that the simple user cannot modify the document state.

Finally, it is important to observe that in order to develop, test or execute a particular document, it is necessary to have specific rights which can only be assigned by the administrator. For a better understanding of the Security Policy please refer to next paragraph.

4.3.2 REPORTS

It has been chosen to use JasperReport for reports, in order to take advantage of its possibility of customization, which is necessary to punctually intervene on the report queries in order to correctly manage profiles, scales and graphical representations of KPIs.

This is why a scriptlet has been implemented to construct dashboard reports. Every dashboard has its own scriptlet and shares some common methods with the other dashboars through the scriptlet ReportUtilities.java (These classes are contained in the scriptletSpago4QReport.jar in the directory

exo-home\webapps\SpagoBIJasperReportEngine\WEB-INF\lib). This technique is used by every dashboard report in order to obtain scales, colours and thresholds from the database.

The following code fragment shows the method "buildProfile" which implements the query profiling that will be executed following the parameters passed to the report through the behaviour model.

The logical condition that wants to be implemented is:

- select KPI_VALUE from DWH
- where areas IN (myAreas)
- and projects IN (myProjects)

If no value is passed for myAreas or myProjects, the referred condition will not be created.

```
private String buildProfile(String pArea, String pProject) {
   String queryProfile = "";
   if ((pArea != null) &&  (!pArea.equalsIgnoreCase("-1"))){
      pArea = pArea.replace("'", "");
      String[] arrArea = pArea.split(",");
      String pArea2 = "";
      for (int i = 0; i < arrArea.length; i++) {
         pArea2 += "'" + arrArea[i] + "'";
         if (i < arrArea.length - 1)
            pArea2 += ",";
      }
      queryProfile += " and dj.id_organization IN (" + pArea2 + ")";
   }
   if ((pProject != null) &&  (!pProject.equalsIgnoreCase("-1"))){
      pProject = pProject.replace("'", "");
      String[] arrProject = pProject.split(",");
      String pProject2 = "";
      for (int i = 0; i < arrProject.length; i++) {
         pProject2 += "'" + arrProject[i] + "'";
         if (i < arrProject.length - 1)
            pProject2 += ",";
      }
      queryProfile += " and ft.id_project IN (" + pProject2 + ")";
   }
   return queryProfile;
}
```

4.3.3 DOCUMENT ORGANIZATION AND SECURITY POLICY

SpagoBI sorts documents in a "Functionalities Tree" which is a File System that can be modified only by an administrator user.

This allows to better organize documents, grouping them by folders, and to realize a Security Policy. In fact, a user can develop, test or execute a document only if he has at least one role belonging to the corresponding permissions on the folder containing it.
Only an administrator user can set these authorizations for each role and each folder.

For instance, in order to develop a document it is necessary:

- to be defined as *Developer* by the portal administrator;
- to have at least a role that belongs the *Development* rights on the folder that contains the document.

To execute a document it is required to:

•to have at least a role that belongs the *Execute* rights on the folder that contains the document.

The document tree, figure 17, is composed by the following folders:

- KPI: it contains analytical documents of Spago4Q
- Model Navigator: it contains SpagoBI documents (reports) used to navigate and access KPIs

The KPI folder has as many sub-folders as the number of existent groups: General Manager, Quality Manager, Area Manager, Project Manager, Project Team. Each of those folders contains 3 sub-folder referring to the functional areas covered by Spago4Q: Requirements, Bugs, Tests.

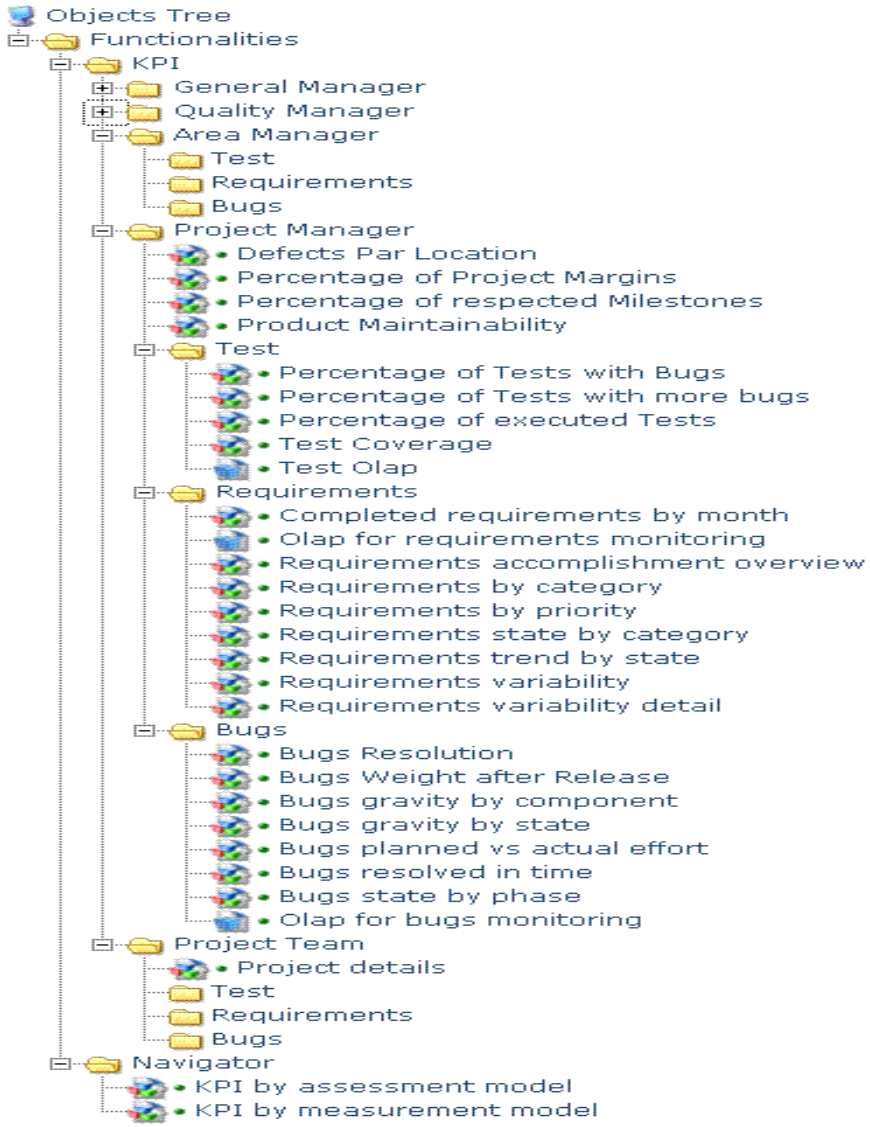

Figure 17: Objects tree examples.

4.3.4 Analytical Document sample.

Naming convention: the parameters to be passed to reports have to be always in the form NamePar (the main ones are AreaPar, ProjectPar, YearPar, MonthPar). This, in order to give common default parameters in the navigation documents.

Following, are the default parameters for every type of group, that will have to be passed through the portlet SBIFunctionality. These values, combined with the behaviour model, permit to directly view the KPIs without requiring any parameter values to the user. This is used in pages such as "Project View" where the objective is to have directly an overview of the situation without inserting any parameter value.

General Manager:
ParThresholdType=0&MonthPar=0&YearPar=0&AreaPar=-1&ProjectPar=-1&CustomerPar=-1&MethodPar=-1&param_output_format=HTML

Quality manager:
ParThresholdType=0&MonthPar=0&YearPar=0&AreaPar=-1&ProjectPar=-1&CustomerPar=-1&MethodPar=-1&param_output_format=HTML

Area manager:
ParThresholdType=0&MonthPar=0&YearPar=0&ProjectPar=-1&CustomerPar=-1&MethodPar=-1&param_output_format=HTML

Project manager:
ParThresholdType=0&MonthPar=0&YearPar=0&CustomerPar=-1&MethodPar=-1&param_output_format=HTML

4.3.5 Behaviour Model

Through the behaviour model it is possible to show to the logged user the KPIs calculated only on the group of Areas and Projects under his responsibility.

At the login moment, a profile parameter called "UserID" is associated to every user. In the demo its value is the user's code number. This profile attribute will then be used to determine the profiling characteristics: areas and projects under its responsibility.

Following are some useful parameters. Every Spago4Q document will necessary refer to these ones.

4.3.6 Parameter "Area"

It has 3 use modes:

- AREA_ALL: gives access to all existing areas (associated to General Manager and Quality Manager)
- AREA_PRF: gives access only to the areas visible to the logged user (associated to Area Manager)
- AREA_BY_PROJECT_PRF: gives access to all areas associated to the user's projects (associated to Project Manager and Project Team)

4.3.7 PARAMETER "PROJECT"

It has 3 use modes:

- PROJECT_ALL: gives access to all existing projects (associated to General Manager and Quality Manager)
- PROJECT_BY_AREA_PRF: gives access to all projects of the user's areas (associated to Area Manager)
- PROJECT_PRF: gives access only to the projects visible to the logged user (associated to Project Manager and Project Team)

4.4 HOW TO CREATE A NEW EXTRACTOR

To start to work with extractors implementation you need to download the Spago4Q package Spago4q-Extractors-1.0.0-RC1-src.zip. Our tip is to import the source code into your favourite IDE (like Eclipse). For the projects inside this archive we do not provide an automatic build environment.

4.4.1 WHAT IS AN EXTRACTOR

The extractor is what we consider to be the actor that collects data from a source tool (database, jira, svn, …) and arranges the information according to the specified area interface (bug, test, requirement, …).

The key points of the extractors design are the following:

- Each extractor should be easily integrated in a Talend job
- The extractor behaviour should be manageable through its configuration
- An extraction process could involve more than one data source (not yet implemented).

The current implementation provides the possibility for one extraction process to extract data only from one data source.
Each extraction operation uses an extractor to get the items (the list of bugs, …) from the data source. For each extracted item field it's possible to apply a mapping (script) that translates the real value to one acceptable for the area interface field itself.

4.4.2 IMPLEMENTATION

If you want to work with extractor implementations, we provide in this paragraph some useful information starting from the next class diagram (figure 18):

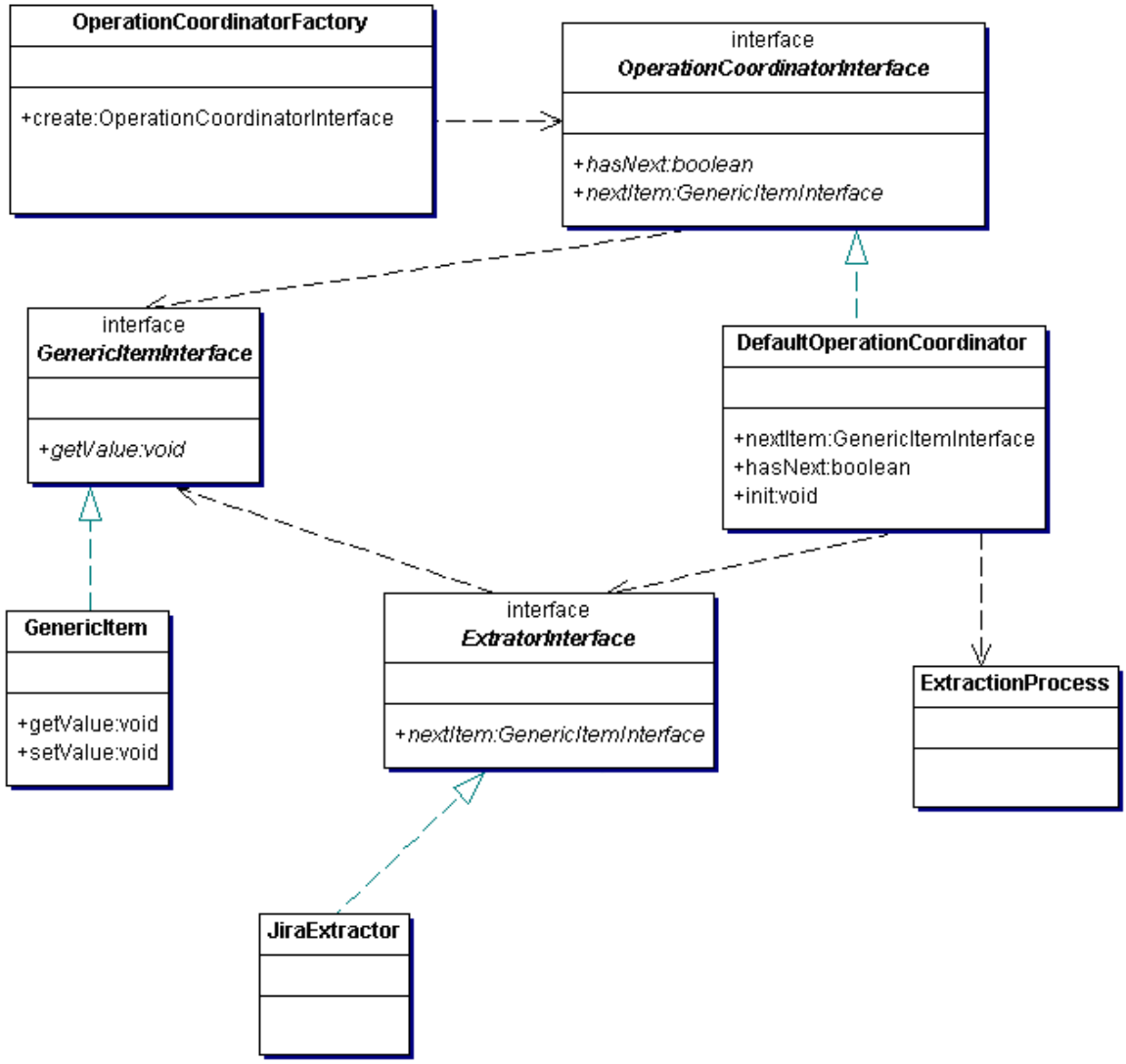

Figure 18: Class diagram.

Every extractor implementation has to be based upon a given interface that respects the iterator paradigm:

```
package it.eng.extractors;

public interface ExtratorInterface {

	/**
	 * Extractor initialization
	 *
	 */
	void init(ExtractionProcess process);

	/**
	 * Return the next available item
	 */
	GenericItemInterface nextItem();

	boolean hasNext();
}
```

This choice has been done to simplify the Talend integration.
Looking at this interface definition we can say that the extractor configuration, usually, is retrieved inside the **init** method, while the other two methods allow to use the extractor as a common iterator.
The GenericItemInterface (and its provided implementation: GenericItem) forces each item interface to be a kind of map of key/value pairs. The set of keys to be used depends on the specific area interface (test, bug, …) considered for the extraction process.

At this point to add a new extractor for a new tool, or another implementation of an existing one, the first step is to create the extractor class like this (jira example):

```
package it.eng.extractors.jira;

public class JiraExtractor extends AbstractExtractor implements
ExtratorInterface {

	void init(ExtractionProcess process){
	 …
	};

	GenericItemInterface nextItem(){
	 …
	};

	boolean hasNext(){
	 …
	};

	…

}
```

All the needed information will be stored inside the database configuration tables to let the extractor be dynamically configured and work properly. The extractors use a shared, custom library to access the configuration database (see APIConfiguration project). This library is based on Hibernate, so all the configuration information are manageable as plain java beans.

If you are going to define a new data interface for a new area (bug, test, …) follow the steps about the DWH reported in the relative section.

4.4.3 CONFIGURATION

As previously specified all the information concerning the extraction processes configurations are stored in the database and they are inside the tables that start with an **e_** prefix. Here you can find the configurations of the available data sources and the currently configured extraction processes. For a data source instance we have to specify the parameters used to connect to that specific type of data source. For each extraction process you have to associate an operation, that uses a data source to extract data for one area (interface). The operation needs a list of properties, that defines which items to select, and a list of fields that define the associations and mappings, between the item read and the common area interface to load in the DWH.
The APIConfiguration project helps each extractor implementation to read its own configuration information.

To improve the easiness of each extractor configuration task, all the information are stored inside the database and are managed with provided portlets.

Some database table is only used to simplify the configuration phase via portlet. These tables, are not manageable from configuration portlets. Here is a brief description of these tables: if you are going to add a new source type you have to specify which are the area interfaces that could be obtained from that kind of tool in **e_source_type_interface** we keep this association. To see which are the parameters to set for a new data source instance take a look at **e_source_type_parameter**. To set the parameters and fields for an operation (that is directly related to an area interface) see, respectively, **e_interface_parameter** and **e_interface_filed**.

4.5 HOW TO CREATE A NEW ETL IN TALEND

4.5.1 GET PREPARED

- Install the latest version of TALEND
- Download and expand the Spago4Q package **Spago4Q-ETL-1.0.0-RC1.zip**. Inside this package you will find the folders:
  - userComponents
  - extractors
  - fileInputExcel
- Copy fileInputExcel folder into the TALEND directory (for example in C:\Programs\TOS-All-r6191-V2.2.0GA\)
- Open TALEND and Import Project from extractor folder

- From the Import window Browse to find the **extractors** folder

- Select the **extractors** folder and click OK

- Click FINISH
- Click OK
- Go to Window-> Preferences and select Talend->Components

- Browse and select the userComponent Folder. This will tell Talend, where to find new components. (for example the component "extractors")

4.5.2 TALEND JOBS

4.5.2.1 Talend Job Examples

In the current workspace you will find some ETL job examples to extract data from an Excel File or from JIRA in order to populate the fact tables (see directories Bugs, Requirements and Test) .

You will also find other examples to populate all the DWH tables concerning the assessment and measurement frameworks (see directories MeasurementFramework and AssessmentFramework) .

In order to make those jobs work, you have to configure Metadata->Db Connections->Spago4Q 0.1 with your own data (password; IP etc) . Figure 19.

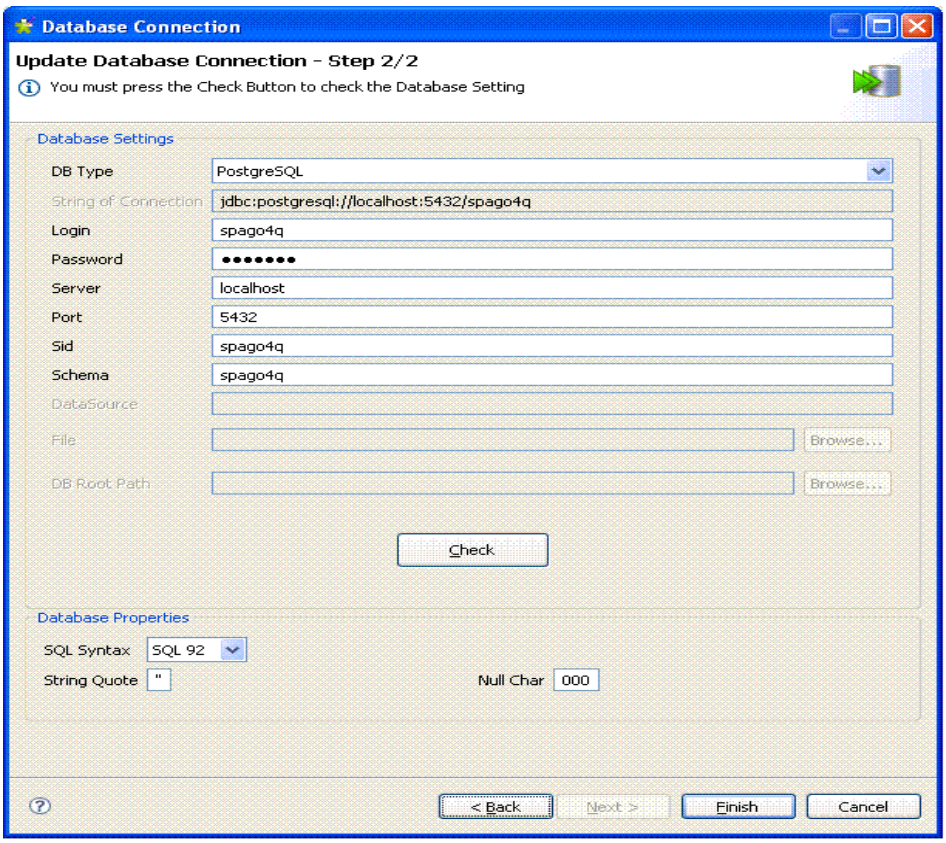

Figure 19: Configuration examples.

4.5.2.2 Data from Excel File

- Open all the jobs of the folder you want to use (for example Bugs-->Excel)
- Open the file named "areaName"Extraction

•Click on the excel Component. Go to the properties sheet and change the origin File name to the one you want to use in the directory TalendHome/InputFiles (for example C:\Programs\TOS-All-r6191-V2.2.0GA\InputFiles\Bugs).

•Open ExtractorsRoutines in Code/Routines and replace the date "20-12-2007" in the methods getDayNow(), getMonthNow(), getYearNow() with the date in which you want to load your data.

•To run the entire job run the file named ETL_"AREANAME"_TOTAL with the green button. If it works, it should fill up the concerned fact table and return Exit Code = 0

4.5.2.3 Data from JIRA

Open all the jobs of the folder you want to use (for example Bugs-->JIRA)
Open the file named "areaName"Extraction_JIRA
Click on the "extractors" Component. Go to the properties sheet and change the process from "PROCESSO_JIRA" to the name of the process you are using.
Open ExtractorsRoutines in Code/Routines and replace the date "20-12-2007" in the methods getDayNow(), getMonthNow(), getYearNow() with the date in which you want to load your data.
To run the entire job run the file named ETL_"AREANAME"_TOTAL_JIRA with the green button. If it works, it should fill up the concerned fact table and return Exit Code = 0

4.5.3 NEW TALEND JOBS

4.5.3.1 Data from Excel File

•Create a new Directory for your jobs
•If you want to use an Excel file for one of the 3 areas: Bugs, Requirements; Test, just copy all the jobs from the folder that concerns you (for example from Bugs-->Excel) and paste them in your new folder.
•Make your own Excel file by modifying one of those already present in the directory InputFiles. If you only modify data and do not change the columns number and order, you can keep the "areaName"Extraction job as it is. Just check the column names of your fact table to see if they match.
•If instead you want to completely change the columns of your Excel file or you want to fill up a fact table for an area different from the ones above, you will have to redo the job Talend by following the Talend User Guide.
•Click on the excel Component. Go to the properties sheet and change the origin File name to the one you want to use.
•Open ExtractorsRoutines in Code/Routines and replace the date "20-12-2007" in the methods getDayNow(), getMonthNow(), getYearNow() with the date in which you want to load your data.
•To run the entire job run the file named ETL_"AREANAME"_TOTAL with the green button. If it works, it should fill up the concerned fact table and return Exit Code = 0

4.5.3.2 Data from JIRA

•Create a new Directory for your jobs
•If you want to make a job for one of the 3 areas: Bugs, Requirements; Test, just copy all the jobs from the folder that concerns you (for example from Bugs-->JIRA).
•Check all the column names by editing the Schema of every component.
•Configure your PROCESS_JIRA through the Spag4q wizards.
•Click on the "extractors" Component. Go to the properties sheet and change the process from "PROCESSO_JIRA" to the name of the process you are using.

•Open ExtractorsRoutines in Code/Routines and replace the date "20-12-2007" in the methods getDayNow(), getMonthNow(), getYearNow() with the date in which you want to load your data.
•To run the entire job run the file named ETL_"AREANAME"_TOTAL with the green button. If it works, it should fill up the concerned fact table and return Exit Code = 0

4.6 HOW TO CREATE A NEW AREA IN DATAWAREHOUSE

The following chapter gives some guide lines in order to introduce a new measure area in Spago4q.

Naming convention

Datawarehouse tables are characterized by a prefix identifying the role:
t_*: dimension table
ft_*: fact table
e_*: configuration table for extraction process
w_*: temporary table (staging area) .

4.6.1 DATAWAREHOUSE

- Add a new line in the table dt_area, with an id, a name (es. bug, test...) and a reference table which would usually be called ft_"areaName"
- Analyze the different measures you might want to use for this Area (planned effort etc.) in order to define which fields to introduce in the fact table.
- Construct the fact table with the relative dimension tables.

4.6.2 FACT TABLE

The fact table will have to be called ft_"areaName" and should contain the following fields:

- id_transaction as unique identifier of the table records. It should be an incremental number.
- id_data_source linked to the table e_data_source that contains information of the source from which data was retrieved.
- id_begin_time and id_end_time, linked to the table dt_time, which indicate the validity period of the object represented in the fact table (bug, test etc.). Following the bug area example, the first time that a bug is iserted in the fact table, its id_begin_time will be the date in which the ETL runs and its id_end_time will be -1 (meaning undefined time). The next time that the same requirement will be introduced in the fact table, its new record will have the date of the new ETL run as id_begin_time and -1 as id_end_time. The earlier record instead, will have its id_end_time changed into the actual date of the new ETL run (which is exactly the one of the new record id_begin_time).
- id_"areaName" usually numeric and unique made up of 4 digits
- name referred to each id_"areaName"

- id_project_structure_level3 relative to the table dt_project_structure_level3 in order to indicate to which project the specific bug, requirement etc. is referred
- it's then possible to introduce other fields called id_"dimensionName" in which to insert the ids referred to the relative dimension tables (to eventually create). Ex. id_phase linked to dt_phase
- add other fields, not linked to any dimension table, as for example: planned_effort, actual_effort; and other fields estimated important to measure the specific area

4.6.3 DIMENSION TABLES

- Some of the dimension tables already in use can be reused for the same purpose. For example dt_time, dt_project_structure_level3 and dt_role have already been used several times.
- Create new dimension tables necessary to your needs, by naming them dt_"dimensionName" and inserting at least 2 fields: id_"dimensionName" as unique identifier and name.

4.7 HOW TO LOAD A SCRATCH DATAWAREHOUSE

Starting from a scratch datawarehouse you have to execute this steps sequence:

- Load lookup tables. You can insert, modify, delete data using tools like TALEND or by inserting them by hand.
- Load dimension tables. They define Organizations, projects, customers, … You need to collect data from your own repository implementing specialized extractors and ETL procedures.
- Load Configuration tables. You can manage them by Spago4Q Administration wizards.

Refer to the chapter 3.4

- Load Fact tables. You can load them using the extractors and ETL procedures already available in Spago4Q or if your project infrastructure tools are different you have to develop new extractors.

Refer to the chapters 4.4 and 4.5