Metrics

Release 15.4.0.2

CONTACT Software

Sep 12, 2018

Contents

CHAPTER 1

Introduction

CONTACT Metrics provides a framework for managing development processes using key performance indicators (KPIs). *CONTACT Metrics* is used to record, analyze and monitor product and process KPIs and thus provide objective criteria for ongoing improvement in the development process. KPI cockpits are available for both types of KPIs. They provide an overview of all the KPIs and dynamically display the history of a KPI and, where applicable, any actions defined for it. Integrated action management also allows targeted corrections to be made in the event of deviations from the target.

In order to lead the competition, companies have to continuously improve their products and processes. This works best with the help of Key Performance Indicators, Balanced Scorecard and other KPIs. PLM/Project management systems represent the most important source for data relevant to products, projects and processes from the product development process. Therefore, they are fundamentally capable of making KPIs comprehensively available, they are ideally suited also for supporting an integrated KPI management for targeted improvement of the development process.

In day-to-day business, the relevant KPIs differ depending on the role of the observer in the company:

- **Design to x**

  in terms of stated goals for a product and its key properties, such as costs, weight, installation space, consumption, emissions, etc.

- **Project controlling**

  based on dedicated KPIs, such as Earned Value, Milestone Trend, etc.

Both of these areas are concerned not with basic process improvements, but 'only' with the controlling and checking in the current case.

- **PLM Performance Improvement**

  based on KPIs that can fundamentally characterize a process.

Here it is insufficient to consider a single product or project; instead one must look back at the average and the trend of the development organization or a process. Typical Key Performance Indicators (KPIs) are, for example, cycle times or the documentation level of engineering changes.

The CONTACT KPI management traces these domains back to a uniform technical basis, which enables simultaneous support of the three above-mentioned use cases.

There is a distinction between process and object KPIs in this regard. Object KPIs include the properties of a single business object. On the other hand, process KPIs, also referred to as class KPIs in the following, mediate effectively between the object KPIs and thus enable the overall evaluation of a business process. The system supports the management of both KPI types: process (class) and object KPIs.

To summarize, KPIs in KPI management are the quantitative characterization of business objects and processes. But they include more than the individual KPI itself: Usually other aspects are also relevant, such as the automatic computation of actual KPI values according to a freely definable cycle, the automatic aggregation of actual values via object structures, the recording of the actual value history, or the option of defining stated goals for KPIs via formula expressions.

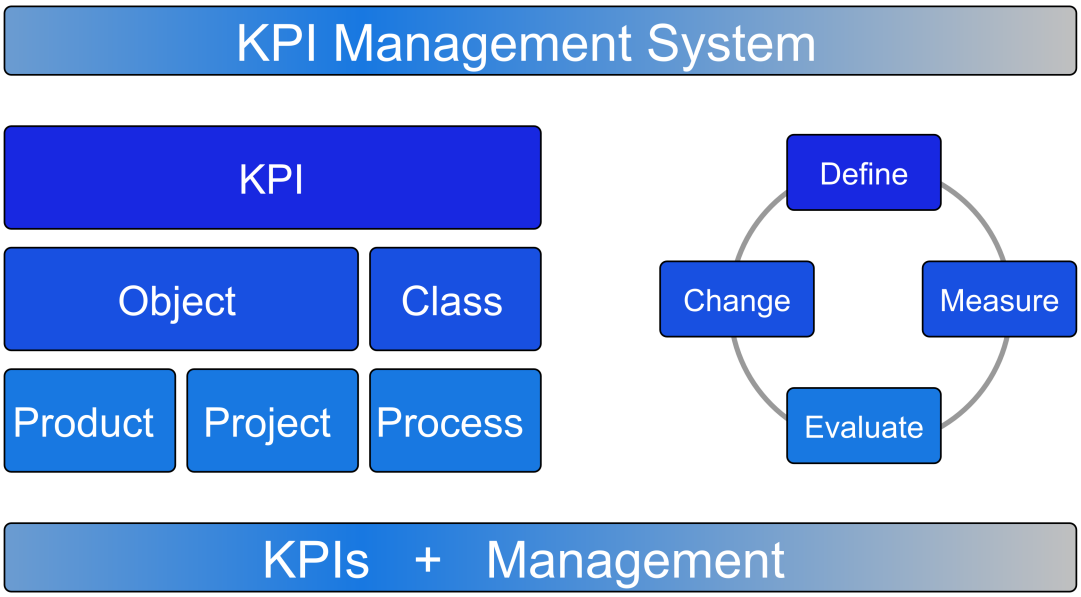

Fig. 1.1: KPI management

CHAPTER 2

Architecture

The following begins with a look at the basic architecture of the KPI management system, in order afterwards to use concrete examples to show the necessary steps for defining different KPIs. A very simplified UML chart (*Simplified representation of the architecture of the KPI management system* (page 4)) is presented at this point to illustrate the architecture and dependencies within the KPI management system.

In this case, very simplified means that only the entities, including their designation in the Data Dictionary and the links to one another, are depicted. Proceeding from the base classes (red), the derived classes for object and class KPIs are differentiated by color (class KPIs: green; object KPIs: blue). Classes displayed in gray were already implemented and are only referenced by the KPI management system.

The difference between object and class KPIs was already discussed in the *introduction* (page 1). The separation of both KPI types becomes clear again in the architecture of the KPI system. Divergent properties or relationships are required for object and class KPIs both for the KPI definition and for the class assignment. Thus, for example, predecessors and successors have to be defined for aggregated object KPIs, which enable an aggregation via a hierarchical structure in the system.

Furthermore, there is an essential differentiation between KPI definitions and class assignments. The KPI definition is used to determined which basic properties the desired KPI should have. The class assignment specifies the concrete objects in the system for which a KPI with characteristic properties determined by a KPI definition should be created.

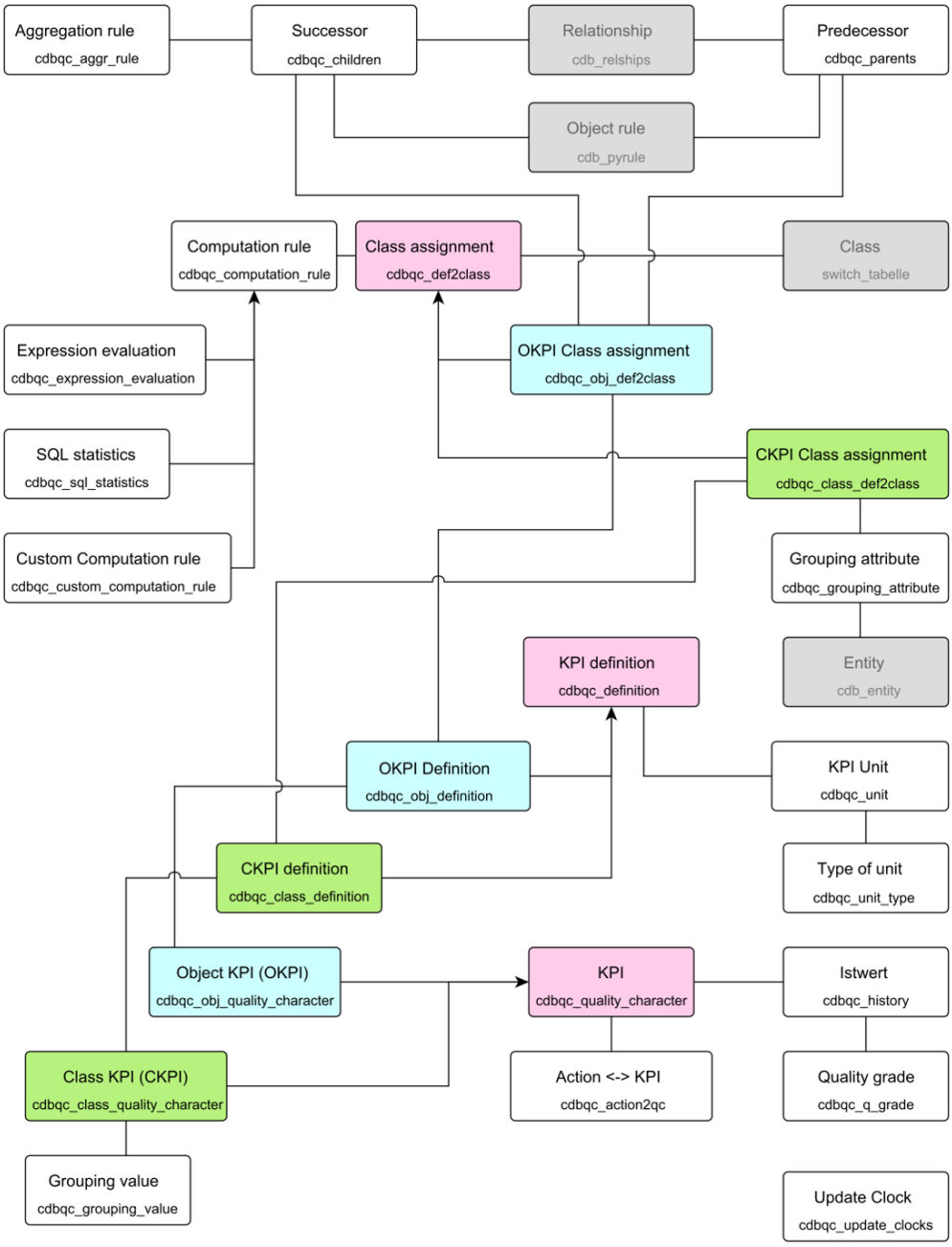

Fig. 2.1: Simplified representation of the architecture of the KPI management system

CHAPTER 3

KPI definition

In accordance with the architecture, two types of KPIs can be defined in the system. For the object KPIs, there is also a distinction between computed and not computed KPIs. The specification is made already in the definition. To make the differences in the definition understandable, the following explanations each reference an example KPI from the corresponding area, that is,

- **Computed object KPI: Cost Performance Index (EVA KPI)**
- **Not computed object KPI: Product weight**
- **Class KPI with SQL statistics: Number of documents**

The entry point for the KPI management is located in the navigation under *Administration/Configuration → Configuration → KPI definition*. Below the *KPI definition* menu item, you can access each of the two KPI definition types, object and class KPI. Based on a search, the predefined KPIs are listed according to the KPI definition types.

Initially, all KPI definitions are in the *New* state. The significance and consequences of a state change are described in the *Activating KPIs* (page 17) and *Deactivating KPIs* (page 18) sections. The following presents a detailed explanation of the procedure for defining object and class KPIs.

A KPI definition is created via the pop-up menu of the *KPI definition* navigation item or directly at the *Object KPIs* and *Class KPIs* subitems. It can also be created via the pop-up menu of a results list from object or class KPIs. In each case, a dialog is called up.

The information required for a KPI definition is nearly identical to the information for object and class KPIs. The equivalent information includes the name (in German and English), a code, the selection of the type, and specification of a unit and/or a default target range of the KPI definition.

> **Note:** The target range does not have to be specified during the KPI definition; it can be entered or changed any time later.

In both dialogs you can also make settings for the cockpits (see qc-menuezugang).

Mandatory fields here include the German *Name*, the *Type* and the *Assignment of persons or roles* in the KPI Cockpit area. The *Name* is for specifying the KPI definition or the corresponding KPIs. The *Type* input field refers to the *Unit* field and defines an area for the concrete unit. For example, you could specify *Currency* as the type of unit. Concrete units for this could be *Euro*, *Pound* or *Dollar*. It is selected from a list of options, which can also be directly expanded.

The person or role is likewise specified using a list of options. The authorization for displaying and editing the KPI(s) is specified based on the person or role. However, it is not possible to create persons or roles here directly.

In addition to the mandatory fields, the fields *Name(s)*, *Code*, *Unit*, *Default Target Area* and in the KPI Cockpit area the fields *Visible*, *Position* and *Time Window from*. Specifying values in these fields is optional.

Name (en) simply requires the designation of the KPI definition in English. A unique abbreviation can be specified for the KPI definition as the *Code*. This code can be accessed when defining *computation rules* (page 15) to

implement the computation, based on additional KPIs in the system. The preceding section has already dealt with the *unit*.

For the *default target range*, the value or value range desired can be specified for the KPI. Valid expressions for the target range include combinations of these characters

- >
- <
- (
-)
- >=
- <=

the logical operators

- and
- or
- not

as well as whole and real numbers. Furthermore, you can specify a list of numbers separated by semicolons.

In the KPI Cockpit area, you can define the visibility and position of the key figures and the start of the time window to be displayed in the cockpit. If the key figures should be listed in the cockpit, the control box *Visible* must be activated. An integer numeric value can be specified for the position. The position is used to sort key figure entries in the process and object KPI Cockpit tables.

The Object KPI definition dialog also includes the *Computed* and *Asynchronous aggregation* mandatory fields or check boxes. A check box can be used here to specify how the respective KPIs are to be computed. All entries are confirmed by closing the dialog with the *New* button.

The process when defining the example KPIs mentioned is dealt with here.

Computed object KPI: Cost Performance Index

The Cost Performance Index KPI is an object KPI. In addition, this is to be computed automatically. The creation dialog for the object KPI definition of this KPI requires the following information:

Dialog field	Value
Name (de)	Cost Performance Index
Name (en)	Cost Performance Index
Code	CPI
Type	Indicator
Unit	
Default target range	
Computed	Yes
Asynchronous aggregation	No
Visible	Yes
Person/Role	Project manager
Position	0

When specifying a code, keep in mind that it has to be unique among KPI definitions throughout the system. The *Type* can be selected from a list of options. No additional unit is needed to select the indicator. Furthermore, no *Default target range* and no particular *Position* are specified in this example. Likewise, there is no aggregation for this KPI, but instead a computation via other KPIs in the system. Therefore the *Computed* check box is selected.

Not computed object KPI: Product weight

Product weight is also an object KPI, but it should not be computed automatically; instead it should be aggregated via a product structure. The information necessary for the KPI definition would be:

Dialog field	Value
Name (de)	Product weight
Name (en)	Product weight
Code	
Type	Weight
Unit	Kilograms
Default target range	
Computed	No
Asynchronous aggregation	No
Visible	Yes
Person/Role	public
Position	0

No *Code* is specified here for the KPI definition. Likewise, no target range is specified, and none of the available computation rules is selected. In this example, the value for the respective object results from the values of the subordinate objects; in other words, the weight of a product results from the weights of potential subproducts. Specifying the `public` role defines that KPIs of this definition can be accessed by all users.

Class KPI with SQL statistics: Number of documents

Number of documents is described by a class KPI. The computation is carried out using SQL statistics, which is explained in the *Computation rules* (page 15) section. This means that the following information is required for the KPI definition:

Dialog field	Value
Name (de)	Number of documents
Name (en)	Amount of documents
Code	
Type	Indicator
Unit	
Default target range	
Visible	Yes
Person/Role	public
Position	20

The *Type* is an indicator again, which means that no special unit is necessary. For this KPI definition as well, no target range is specified at first; in this case, however, an example value is entered for the *Position*.

CHAPTER 4

Class association

This section deals with the particularities and differences of the two KPI types, with regard to the configuration of class associations for KPI definitions. First, the basic procedure for assigning a class to a KPI definition is defined. Then the continuous example KPIs are explained.

Note: A prerequisite for assigning a class is that a KPI definition was already created.

A class can be associated in the context of a KPI definition. This means that you can call up the data sheet of a KPI definition via the result list of KPI definitions. Starting from the data sheet, you arrive at the *Class association* tab. You can create in the results area of this tab using the pop-up menu (right mouse button).

The corresponding information for doing this will be requested by a dialog. To assign a class in the context of an object KPI definition, in addition to the mandatory input fields *Class name* and *KPI definition* there are the fields *Object rule*, *Computation rule*, *Calculation Frequency* and *Next computation* as well as the check boxes *Computed* and *Automatically apply aggregated value as actual value*.

Note: If the *Computed* check box was already selected during the KPI definition, the *Computation rule* and *Calculation Frequency* dialog fields are also mandatory fields.

The creation dialog for associating the class of a class KPI is less extensive. The only input fields here are *Class name*, *KPI definition*, *Object rule*, *Computation rule*, *Calculation Frequency* and *Next computation*. Furthermore, in addition to *Class name* and *KPI definition*, the *Computation rule* and *Calculation Frequency* fields are also mandatory fields.

The *Class name* refers to the class of the objects for which the KPIs are to be generated according to the definition. It is selected from a list of options with all of the available classes in the system. The *KPI definition* field is usually prepopulated with the name of the corresponding definition, since a class can be associated only in the context of a KPI definition.

Before assigning a class, make sure that a fully qualified Python name is entered for the class to be assigned. Another prerequisite for generating and computing KPIs later is the existing derivation of the class of the class assignment from the class `WithQualityCharacteristics` (`cs.metrics.qcclasses.WithQualityCharacteristics`).

Different configurations can be implemented for class and object KPIs by using object rules. You can use an *Object Rule* to define the condition for evaluation using SQL statistics for class assignments in the context of a class KPI definition. The general function and configuration of object rules can be found in the administration manual (see Configurable rules). The different meaning of object rules with regard to a class assignment and object KPIs is explained in the corresponding subsections (*Object rule in the context of a class assignment* (page 10), *Object rule in the context of class KPIs* (page 11) and *Obejct rules in the context of object KPIs* (page 10)).

You can configure additional settings for the computation in the *Computation* input area. To do so, the *Computed* check box must have already been selected during the KPI definition, otherwise this area cannot be modified in the current dialog. If these values can be edited, you can select a *computation rule* (see *Computation rules*

(page 15)) and specify a time cycle for the computation. If necessary, this information results in an entry in the *Next computation* field.

In the last section of the dialog, the system decides whether the aggregated value should be copied automatically as the current value for the relevant KPI. Under certain conditions, the transfer is also possible manually within the object KPI Cockpit. The exact procedure is described in qc-menuezugang-obj.

This input dialog is also confirmed via the *New* button. The example KPIs should clarify the meaning of the dialog fields.

Computed object KPI: Cost Performance Index

The following information is provided for the class assignment for KPI definition Cost Performance Index:

Dialog field	Value
Class name	cdbpcs_project
KPI definition	Cost Performance Index
Object rule	cdbpcs: Active Project
Computed	Yes
Computation rule	EVA: Cost performance index
Calculation Frequency	Daily
Next computation	
Take as AV	No

This KPI is a project KPI, therefore the corresponding name of the class for projects is entered for the *class name*. The *KPI definition* is prepopulated according to the name of the KPI definition (Cost Performance Index).

Likewise, the value of the *Computed* check box is taken over according to the selection when creating the KPI definition; in this case a computation is to take place. Accordingly, a computation rule was selected. The target date for the next computation is determined during the *activation* (page 17) of the KPI definition.

In addition, an *object rule* was specified for this class assignment. This filters only active projects in the system, and the KPI should be generated only for projects that correspond to this filter. The abbreviation *Take as AV* stands for the designation of the check box *Automatically apply aggregated value as actual value*. In this example it is not confirmed.

Not computed object KPI: Product weight

Necessary information for the class assignment:

Dialog field	Value
Class name	part
KPI definition	Product weight
Object rule	
Computed	No
Computation rule	
Calculation Frequency	
Next computation	
Take as AV	No

For this class assignment you need only the name of the class for whose objects the KPI is to be created, that is, the *class name* `part` for products or parts. Information for computation cannot be provided, because no computation is desired for the product weight KPI definition. The computation results from an aggregation of values, via the product structure, but the aggregated value should not be taken over automatically as the actual value.

Class KPI with SQL statistics: Number of documents

Finally, the configuration information is listed for the number of documents class KPI.

Dialog field	Value
Class name	document
KPI definition	Number of documents
Object rule	cdbqc: non unvalid documents
Computation rule	Number of documents without revision
Calculation Frequency	Monthly
Next computation	dd.mm.yyyy

Class objects with specific properties are also filtered via this *object rule*; here, it concerns the documents that are valid.

4.1 Object rule

When you assign a class to the KPI definition, you can also restrict the objects for which a KPI should be generated. This limitation is realized by using *Object Rules*. The general function and configuration of object rules can be found in the administration manual (see Configurable rules).

Since it can happen that the same class can be assigned to a KPI definition multiple times while specifying different object rules, one must make sure that these object rules are mutually exclusive. This means that no class object fulfilling two or more object rules with the same KPI definition is allowed to exist in the system.

Furthermore, a distinction is made between object rules for class KPIs and object rules for object KPIs during the class assignment. The differences for each KPI type are explained in the following *Class KPIs* (page 11) and *Object KPIs* (page 11) sections.

4.2 Calculation Frequency

The calculation frequency can be specified as part of a class assignment. A prerequisite for the information is the specification during the KPI definition that computed KPIs are involved.

Class KPIs are generally computed automatically. For object KPIs this can be decided on during the KPI definition. If a computation is to take place, you can specify a computation rule during the class assignment first and then the calculation frequency.

The specified update clocks include the

- Daily
- Weekly
- Monthly
- Quarterly
- Semestral
- Yearly
- Manual

computation of the KPIs. The target date for the next computation is always specified based on the selected update clock. The computation is always carried out at the beginning of the corresponding period. This means a daily computation occurs every day at midnight, a weekly computation occurs every Monday, a monthly computation occurs on the first of every month, etc.

In addition, you can trigger a manual calculation for individual KPIs at any time. On the one hand, this option exists in the context menu of a KPI and on the other hand, a corresponding operation (for each KPI) is offered in the KPI cockpits (see qc-functions-general-operations).

Manual KPIs are not computed regularly by the computation service, but manually by the above-mentioned operation or in event-driven form by other modules.

4.3 Class KPIs

When associating a class for class KPIs there is also the option of specifying attributes for the grouping of KPIs. Furthermore, in the subchapters of this section, reference is made to the particularities with regard to the object rules for class KPIs.

4.3.1 Object rule

If a grouping has been specified for a class KPI during the class assignment, in very rare cases of exception it can happen that an event that, in principle, would lead to a new grouping is not intercepted and the generation of the KPI is not initiated. Events that are definitely monitored are: `create`, `copy`, `modify` and `state_change`.

The administrators must always make sure to structure the object rules so that in fact only the listed events lead to a change of the quantity of the corresponding objects.

4.3.2 Grouping

Class KPIs can be organized based on a grouping. This means that, starting from the data sheet of a successful class assignment for a KPI definition, class attributes can be selected via the *Grouping attributes* tab. In doing so, exactly one KPI is generated for each possible combination of grouping values.

Note: It is possible to specify multiple grouping attributes for a class assignment which are taken into account during the organization of KPIs.

4.4 Object KPIs

After successfully associating a class, parents and children for implementing the aggregation may have to be defined for object KPIs. That will be dealt with in particular in this section.

4.4.1 Object rule

Likewise, as for the class KPIs, the `create`, `copy`, `modify` and `state_change` events are intercepted for object KPIs during the class assignment and the specification of an object rule.

Here, too, the administrators are advised to define the object rules accordingly, and thereby limit the changes to the mentioned events.

4.4.2 Aggregation

The value of most object KPIs depends on the values of subordinate objects. This means that this value is determined based on an aggregation via the subordinate objects. For a successful aggregation, therefore, the referenced parent and child objects have to be known, in each case as viewed from the initial object.

If the *Asynchronous aggregation* and *Computed* check boxes were enabled, this causes no computation to occur, since in this case the computation services (see *Services* (page 20)) run in parallel and block each other.

The respective parent and child objects are defined in the context of a class assignment. Access to the *Parents* and *Children* tabs is provided next to the class assignment data sheet. The individual steps required for configuring these two elements is explained in the following.

Parent

Objects that have a higher-level position within a hierarchical structure, starting from an individual class object, are called parent objects. This means that the respective KPI value of the class object currently being viewed is aggregated upward to the (potential) parent object.

It was already mentioned at the beginning that the *Parents* tab can be reached from the data sheet of an already configured class assignment. After selecting the corresponding tab, you can use the pop-up menu of the results list to create (operation: *New ...*).

The creation dialog provides four input fields. The mandatory fields, which must be filled in, include the *Class name*, *KPI definition* and *Relationship*. Furthermore, you can specify an *object rule* for parent objects. The *Class name* and *KPI definition* are usually already suitably populated from the context of the KPI definition and class assignment; they usually cannot be filled in manually.

An object rule has to be specified to give an exact definition of the current object. Delimiting the object itself can ensure that corresponding parent objects exist. Then you can select the specific *Relationship* from a list of options.

Filtering occurs here already based on the given class name. This means that relationships, starting from the class name, are provided for the possible parent objects as a selection option. An assigned *relationship* should fulfill certain criteria, which are explained in more detail in the *Relationships* (page 14) section.

In the case of the example of the product weight object KPI, the parent configuration looks as follows:

Not computed object KPI: Product weight

Necessary information for configuring the parent object:

Dialog field	Value
Class name	part
KPI definition	Product weight
Object rule	
Relationship	cdbqc_part2assemblies

In accordance with the preceding explanation, the *class name* and *KPI definition* in this example are suitably prepopulated and a relationship to the parent objects (here, the corresponding assemblies) is specified starting from the `part` class. An object rule is not required in this case, since only a single parent object is configured.

Children

Unlike the parent objects, the child objects describe the class objects that possess a subordinate position within a hierarchical structure, starting from an individual class object. The respective KPI value of the class object currently being viewed results from the aggregation of the KPI values of the (potential) child objects.

You can create child objects via the *Children* tab as well as via a pop-up menu operation (*New ...*). The creation dialog is somewhat more comprehensive than for the parent objects, because the aggregation rules are also defined via the child objects.

The mandatory fields *Class name* and *KPI definition* are prepopulated according to the context, in the same way as for the configuration of parent objects. The optional *Object rule* field is also described at the same time. It can be specified for an individual child object and has to be for multiple child objects.

An additional mandatory field requires the selection of an *Aggregation rule*. How the KPI values are to be aggregated within the hierarchical structure is determined based on this rule. The following are available as default options:

- Sum calculation
- Sum calculation taking into account a coefficient
- Maximum and minimum value calculation
- Calculation based on a self-defined rule

. If the sum is to include a coefficient in the calculation, it has to be selected in the *Coefficient name* field via an attribute list. Usually a coefficient is an attribute of the class specified during the class assignment.

Another piece of information is required in the *Python Name Custom Rule* field if a self-defined rule is to be used. Normally this involves a rule that is implemented directly in Python. The fully qualified Python name has to be entered for this. The *Custom computation rules* are described in detail in the section *Aggregation rules* (page 13).

Specification of a *relationship* is also expected for child objects. The selection options are prefiltered again according to the class name and include potential child relationships. Here, too, refer to the chapter *Relationships* (page 14).

The configurations of the child object should be listed following the example of the product weight object KPI:

Not computed object KPI: Product weight

Necessary information for configuring the child object:

Dialog field	Value
Class name	part
KPI definition	Product weight
Object rule	
Aggregation rule	Sum with coefficient
Python Name Custom Rule	
Relationship	cdbqc_part2subparts
Coefficient name	Amount

The *class name* and *KPI definition* are suitably prepopulated again. No *object rules* are required in this example either, since here, too, only a single child object is configured. *Sum with coefficient* was selected as *aggregation rule* and the `menge` attribute was specified in the corresponding field (*Coefficient name*).

It is not unusual for individual components within an assembly to be used more than once; in that case, the amount of the individual components has to be taken into account in the calculation of the total weight. In this case, the *relationship* describes the hierarchically subsequent objects within the structure, the subassemblies or other individual subordinate parts.

Aggregation rules

As part of the configuration description of a child object we already briefly touched on the aggregation rules. Now we will explain the default rules and, above all, the custom rules again.

Default rules

The default rules in the system include:

- Sum formation
- Sum formation taking into account a coefficient
- Maximum and minimum value calculation

As expected, the KPI values are added during the sum formation. A coefficient is multiplied by the corresponding individual value before the addition. The maximum value is determined in the maximum value calculation; similarly, the minimum value is determined in the minimum value calculation, and these are taken into account in the aggregation.

Custom rules

The custom rules represent a special case. The computation is programmed in Python. Therefore, a valid path for the corresponding Python code has to be included in the creation dialog of a child object. When evaluating this

information, an object of the class `cs.metrics.qualitycharacteristics.AggregationContext` is transferred as a parameter. The significance of the parameter is dealt with separately in the chapter *Custom computation rules* (page 15).

Relationships

When specifying relationships, starting from the parent and child objects, the relationships have to fulfill certain properties, which will be listed here.

First, ensure that the specified *Role name* of the relationship in question also corresponds to a valid and accessible reference in the Python code. Second, enter a fully qualified Python name for the logical class.

CHAPTER 5

Computation rules

The following will deal with the respective computation rules that can be used for class and object KPIs.

5.1 Expression evaluation

The expression evaluation concerns an arithmetic expression, which is particularly useful for object KPIs. Elements that could be used for this are listed in the following.

- **Numbers**
- **Attributes of the technical object**

  The class attributes that can be used for assessing objects must be either numbers or date values.

- **Other KPIs**

  In the *KPI definition* (page 5) section above, we explained that a *Code* can be specified during the definition of a KPI. This *Code* can also be used for the expression evaluation, for example, if the value of a KPI depends on the values of other KPIs. Here, the codes of the KPIs in question can be used in the arithmetic expression.

5.2 SQL statistics

If SQL statistics are to be used for computing a KPI value, this happens according to the pattern of the SQL query `SELECT statistics(attribute) FROM relation`. Furthermore, additional WHERE constraints can be specified or defined via an object rule during the class assignment. This results, for example, in the ability to select a reference period and a reference action, which enables computation rules such as *Number of new data records per month* or *Number of changed data records per month*.

The attribute to be evaluated does not have to be a class attribute. All information that the DBMS (database management system) can evaluate is allowed. Thus the wildcard character * can also be set for the attribute in combination with the *Count* statistic.

5.3 Custom computation rules

As already mentioned more than once in the course of this description, a custom rule is implemented in the Python code. Here, it is important for a fully qualified Python name, which references an object that is valid and can be called up, to be entered at the relevant place of the configurations.

When called up or queried, a single parameter is transferred: the suitable KPI object. Based on this parameter, it is possible to determine:

- The necessary technical object

- The KPI definition
- The class assignment
- The class of the class assignment
- The (potential) grouping

etc. Valid return values of the query are a numerical value, as a result of the computation or the expression `cdb.objects.NULL`.

Note: Calling up the custom rule should return the new value, without updating the KPI itself. If it is not possible to compute the KPI, then an exception should be thrown. The system updates the KPI value and logs all exceptions.

Activating KPIs

An activation must be carried out subsequent to the complete configuration of KPI definitions, class assignments, aggregation and computation rules. This means that, in the pop-up menu of a certain KPI definition, the status has to change to *Valid*.

Only afterwards are the individual KPIs generated, corresponding to the KPI definition, for the individual objects. These fulfill an object rule where applicable. If there are already corresponding class objects at this time with the desired KPI, a new KPI is not generated in this case. This can occur, for example, if a valid KPI definition was reset in order to be able to make changes to the definition and was re-activated after a successful change.

Note: All KPIs are generated during the activation; that is, this process can last several minutes under certain circumstances. An example of this is the KPI “Weight”, which is to be created for parts. Since it is entirely possible for there to be a large number of parts in the system, the generation of a KPI for each individual part can be time-consuming under certain circumstances.

CHAPTER 7

Deactivating KPIs

In addition to the option of activating a KPI definition there is, of course, also the option of deactivating. KPI definitions have to be deactivated, for example, if the settings of the KPI definition are to be changed.

If a KPI definition is deactivated, this does not mean that the associated KPIs are deleted. However, these are no longer taken into account in the computation or aggregation until a reactivation. Also in the respective KPI Cockpit, KPIs whose KPI definition has been deactivated are no longer listed.

CHAPTER 8

Catalogs for KPI management

Catalogs are required at various places for configurations related to KPI management. In part, for selecting computation rules or types of units. These browsers are accessible via the navigation bar and can, if necessary, be supplemented with additional entries.

The available catalogs are in the navigation bar under *Administration/Configuration* → *Catalog Administration* → *KPIs*. In the standard version, browsers are made available for:

- Computation rules
- Type of unit
- Unit
- Quality grade
- Aggregation rules
- Update clocks

The preceding material has already dealt with how to use the individual catalogs. The catalog entries, in turn, are expanded by adding a new entry in the pop-up menu of the respective results list (Operation: *New*).

Note: The expansion of the *Quality grade*, *Aggregation rules* and *Update clocks* catalogs has no effect if the corresponding Python code is not adjusted or expanded simultaneously.

CHAPTER 9

Services

To enable the automatic computation or aggregation according to a specified calculation frequency, the corresponding (two) services have to be activated in the system. The services are accessed via the navigation menu *Administration/Configuration —> Administration —> Services*.

The result list will include the services relevant for the KPI management.

9.1 Quality Characteristic Computation Engine

Service name: `cdb.uberserver.services.qc_engine.QCComputationEngine`

If this service is active, the scheduled KPIs are computed daily at midnight.

9.2 Quality Characteristic Aggregation Engine

Service name: `cdb.uberserver.services.qc_engine.QCAggregationEngine`

This service computes the aggregated value of those KPIs, which have been selected for asynchronous aggregation.

The service is implemented as *Message Queue* (see Message Queues).

You can find an overview of the pending and failed tasks in the navigation menu at *Administration/Konfiguration -> Administration -> KPIs -> Aggregation tasks*.

CHAPTER 10

General tips

First, we should point out that the existing interface should always be used when making changes and adaptations in the Python code. The `set_actual_value`, `set_target_value`, `createQC` and `deleteQC` methods, which are necessary for the aggregation and saving of changed values in the history, among other things, are available exclusively in the

`cs.metrics.qualitycharacteristics.ObjectQualityCharacteristic`

and

`cs.metrics.qualitycharacteristics.ClassQualityCharacteristic`

classes.

CHAPTER 11

Examples

Numerous KPIs are already configured in the standard version. If you search for class or object KPIs via the navigation menu, you get a list of all preconfigured KPI definitions.

List of Figures