DR9.15: Concept for Data Quality Analysis and Visualization for PROMISE

Written by:
SAP

DELIVERABLE NO	DR9.15: Concept for Data Quality Analysis and Visualization for PROMISE
DISSEMINATION LEVEL	**CONFIDENTIAL**
DATE	15. September 2007
WORK PACKAGE NO	WP R9: Development of PROMISE Information management system
VERSION NO.	01
ELECTRONIC FILE CODE	DR9 15-v1 0.pdf
CONTRACT NO	507100 PROMISE A Project of the 6th Framework Programme Information Society Technologies (IST)
ABSTRACT	The PROMISE architecture provides PEID-equipped sensors to capture data about product conditions and usage to guide business decisions as well as production automation processes. A challenging issue in this application area is posed by the restricted quality of sensor data due to limited sensor precision as well as sensor failures and malfunctions. Moreover, the data quality is further decreased by data processing to meet resource constraints in streaming as well as storage environments. The issue of how to efficiently provide information about data quality (DQ) is still an open research problem.

STATUS OF DELIVERABLE		
ACTION	**BY**	**DATE** (dd.mm.yyyy)
SUBMITTED (authors)	SAP	19.09.2007
VU (WP Leader)	Andreas Edler	19.09.2007
APPROVED (QIM)	Dimitris Kiritsis	19.09.2007

Revision History

Date (dd.mm.yyyy)	Version	Author	Comments
02.07.07	0.0	Anja Klein	First draft
09.09.07	0.1	Hong-Hai Do	

Authors' contact information

Name	Organisation	E-mail	Tel	Fax
Anja Klein	SAP	anja.klein@sap.com	0351-48116120	
Hong-Hai Do	SAP	Hong-hai.do@sap.com	0351-48116115	

Table of Contents

List of figures

List of Tables

Abbreviations

Abbreviations used in this document:

DoW	PROMISE Description of Work
DSMS	Data Stream Management System
DSS	PROMISE Decision Support System
GUI	Graphical User Interface
OLAP	Online Analytical Processing
OMG	Object Management Group
PDKM	Product Data and Knowledge Management
PEID	Product Embedded Information Device
PLM	Product Life-cycle Management
PROMISE	PROduct life cycle Management and Information tracking using Smart Embedded systems
UML	Unified Modelling Language
WP	Work package of PROMISE project

1 Introduction

1.1 Objectives of Task TR9.15 and scope of this deliverable

In the PROMISE DoW [Pro06] document (version 5.0, page 89), Task TR9.15 ("Data Quality Analysis and Visualization") was described as follows: *"Methods for automated data collection, i.e. sensors, RFID etc., are subject to system-inherent errors, e.g. imprecision of sensors, noise of data transmission channel, etc. If not dealt with properly, data errors are propagated from PEIDs, through the Middleware, and finally to the PDKM and DSS, where they derogate the quality of decision support. This task thus studies the data quality issues and analyzes the influence of data quality on decision making*".

Task TR9.15 comprises:

- Concept of Data Quality considered for various types of data sources
- Analysis of the influence of data processing methods (e.g. in Middleware and PDKM) on Data Quality
- Data Quality Visualization

The objective of task TR9.15, and thus of deliverable DR9.15, is to provide descriptions how to incorporate data quality information in the data streaming environment of the PROMISE middleware as well as how to persistently store data quality characteristics in the PDKM. Besides the metamodel extension, the influence of data processing operators used in both Middleware and PDKM shall be analyzed. Finally, the visualization of data quality dimensions enables the adequate evaluation of imprecise data and measurements.

1.2 Brief document overview

The document layout comes as follows:

Section 2 discusses the term "data quality" by comparing different definitions and introducing accuracy, confidence and completeness as three important quality dimensions for sensor data. In Section 3 this knowledge is used to design a metadata extension to represent data quality information in data streams. Here, the efficient DQ transfer plays an important role. Furthermore, the metadata extensions are incorporated in the PDKM object model presented in the deliverable DR9.2. In Section 4 typical data stream processing operators are analyzed regarding their influence on different data quality dimensions. With the help of the proposed functions, the quality of streaming sensor data can be tracked from the data sources through various processing steps to the respective applications. Approaches for data quality visualization are presented in Section 5 enabling the comprehensive information evaluation to prevent from faulty decisions due to imprecise data. This deliverable closes with a short summary.

2 Incorporating Data Quality Information

The PROMISE architecture provides PEID-equipped sensors to capture data about product conditions and usage to guide business decisions as well as production automation processes. A challenging issue in this application area is posed by the restricted quality of sensor data due to limited sensor precision as well as sensor failures and malfunctions. Moreover, the data quality is further decreased by data processing to meet resource constraints in streaming as well as storage environments. The issue of how to efficiently provide information about data quality (DQ) to applications is still an open research problem.

Based on a discussion of data quality definitions and DQ dimensions, this deliverable provides a concept to efficiently transfer the DQ information along with the measured sensor data. Furthermore, the system object model of the PROMISE PDKM is extended to allow for the efficient storage of data quality information. To enable the data quality tracking through complex data processing steps, the influence of processing operators on data quality was analyzed. The deliverable closes with approaches for data quality visualization supporting effective data evaluation.

2.1 System Overview

This section gives an overview over the application area as well as our proposed solution illustrated in Figure 1. The real world environment, for example a manufacturing area, is monitored with the help of sensors. The measured sensor data is streamed towards the target applications, where the data is processed and actions or decisions are derived. There are two modes of data processing. On the one hand, the data is consumed directly from the stream for basic data analysis in the automatic process control, e.g. during production processes. On the other hand, many business applications require data spanning a wider time interval aggregated in a persistent database. Here, complex data mining and knowledge discovery is performed. Both application scenarios are supported by the generic and flexible solution presented in this deliverable.
Our solution for data quality transfer and management consists of data quality recording, data quality propagation, system object model extensions for persistent data quality storage and data quality visualization.

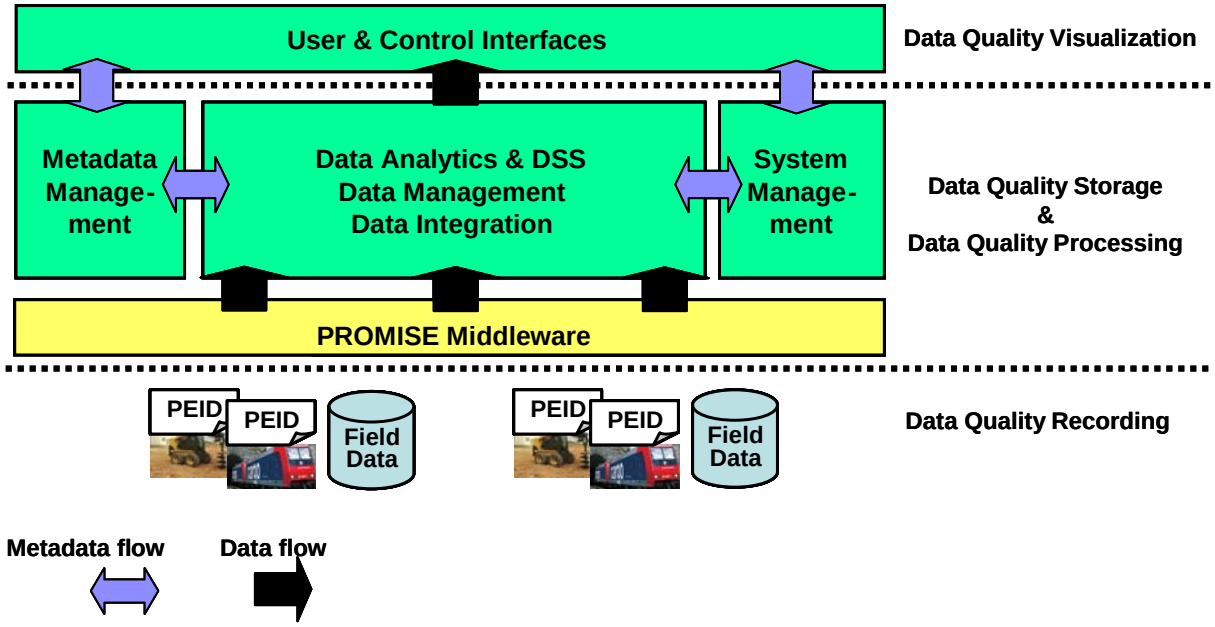

Figure 1: Data Quality within the PROMISE Architecture

2.2 Data Quality Dimensions

2.2.1 Different Data Quality Definitions / Classifications

There are multiple definitions of the term „data quality“. According to the application context, different perceptions of data quality can be recorded and various data quality dimensions can be found. The most general notion is stated as follows:

Data quality describes the suitability of the data for the respective data processing application.

Rahm and Do [Rah00] distinguish data quality problems resulting from a single data source and data errors resulting from the integration of multiple sources as shown in Figure 2. Furthermore, they classify data quality problems according to the level, where the data errors are settled. Thus, there are data errors based on the schema level (uniqueness or naming conflicts) as well as on instance level (duplicates, missing values, inconsistencies).

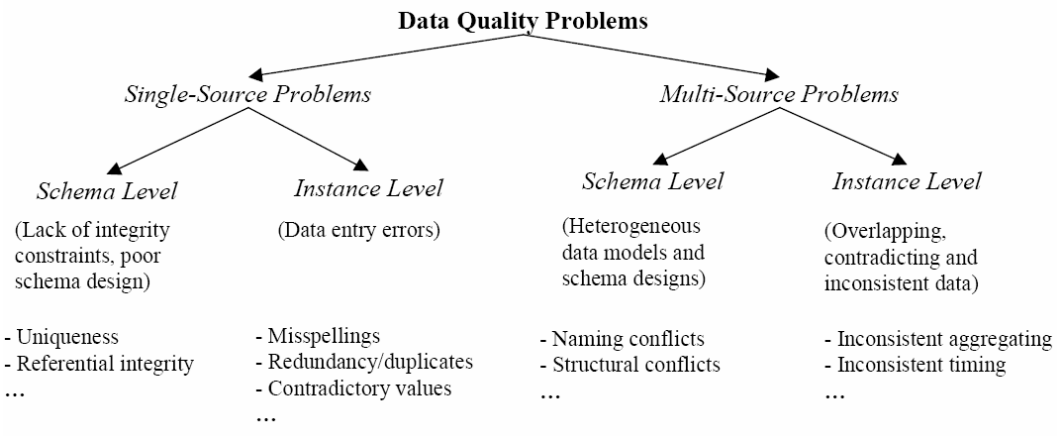

Figure 2: Classification of Data Errors [Rah00]

Wang and Strong go beyond this classification by defining additional data quality dimensions like understandability, completeness or reputation of a data source describing not only single data values or tuples, but complex data volumes (see Table 1).

Table 1: Data Quality Categories [Str97]

Quality Category	Quality Dimensions
Intrinsic Data Quality	believability accuracy objectivity reputation
Contextual Data Quality	value-added relevancy timeliness completeness amount of data
Representational Data Quality	interpretability understandability representational consistency representational conciseness
Accessibility Quality	accessibility access security

Hence, data quality can be regarded as a set of data quality dimensions, whose collection and definitions correspond to the respective application area. In the following, we will focus on numerical sensor data and give a set of data quality dimensions considered as of high importance for this application field.

The data quality dimensions accuracy and completeness are essential for the evaluation of sensor data and thus are described in detail in the following sections. The accuracy describes the systematic error due to sensor imprecision. The sampling operator introduces a new statistical error represented in the confidence.

2.2.2 Accuracy

The DQ dimension accuracy describes the numerical precision of a data item. It is stated in the absolute error *a(j)* of a physical value *v(j)*. The accuracy of a sensor is given by the measurement precision class in the manufacturer's technical specification. In the example, the pressure sensor p1 has a maximum range of 315bar and a precision class of 1%. Thus the maximum absolute error *a(j)* of this sensor is 3,15bar (1% of 315bar).

2.2.3 Confidence

With the help of sampling, the data volume can be reduced significantly. Thus, the sampling operator is an important tool in context of data stream processing, providing the possibility the meet stringent resource capacities. However, this operator introduces a new statistical error. The information loss provoked by sampling the data stream is described by the confidence interval according to [Haa97]. Consider a data stream consisting of 10 values 1,1,1,1,1,1,1,1,1,100 with the average of 10.9, where every 2nd data item is sampled. If the sample contains the value 100, the average will be determined as 20.8. Otherwise the average based on the sample would be computed to be 1. It is easy to see, that those two results contain a non negligible deviation compared to the true average.

Similar to the error resulting from the sensor's accuracy class, the confidence interval is given as an absolute error. The statistical error influences the overall accuracy only if aggregation operators are applied later on, as described in the example above. The goal is to handle every operator of the data processing independently. Thus the new data quality dimension confidence is

introduced. The confidence is described by the parameter ε defining the confidence interval [x - ε; x + ε] enclosing the measured value x containing the true value x' with a given confidence probability *p*. According to [Haa97] ε is defined as

$$\epsilon = \frac{d \cdot \sigma}{\sqrt{n}} \cdot \sqrt{1 - r},$$

where *n* and *r* constitute the sample size and sampling rate, respectively, σ gives the data variance and *d* expresses the confidence probability *p* as the $(1 - p/2)$-quantile.

2.2.4 Completeness

The completeness addresses the problem of missing values due to sensor failures or malfunctions. Multiple strategies exist to deal with missing values in ETL processes and data cleansing [Lee99], whereas the estimation or interpolation of missing values is aspired in the majority of the cases. The data quality dimension completeness helps to distinguish between measured data items and estimated or interpolated ones.

2.3 Data Quality Recording

During the data quality recording, DQ information is captured directly from the sensor. Therefore, the sensor has to be analyzed in detail as shown in Figure 3.

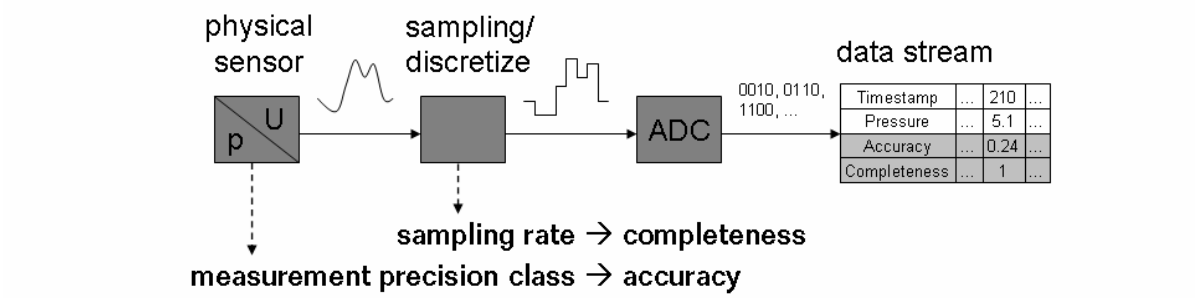

Figure 3: Design of a Sensor

The output of a sensor is a discretized and digitized data stream representing the measured physical values. The characteristics of the sensor define the data quality dimensions of the outgoing data stream. The metadata models presented in this paper are designed to cope with an unlimited number of data quality dimensions. Without loss of generality, we focus on the two important DQ dimensions accuracy and completeness.

The accuracy describes the numerical precision of a data value. It is stated in the absolute or relative error of a physical value. The accuracy of a sensor is given by the measurement precision class in the manufacturer's technical specification.

The sampling rate of the discretization defines the stream rate *r* (e.g. 100/s, 1/10min), which determines the stream length *m* depending on the time *t* and thus serves as reference for the stream completeness *c*.

3 Enhanced Metadata Model

In this section the metadata model extension for data stream systems will be introduced. We present a straightforward approach of so-called naive data quality annotations as a first solution to the problems arising from restricted sensor DQ. To overcome the explosion of data volume given by this naive approach, we then propose the incorporation of jumping windows in the data stream

metamodel. Finally, the DQ calculation and propagation in the presented jumping stream windows in described in detail.

3.1 Naive DQ Annotations

The naive approach of data quality annotations consists in streaming the data quality information for each DQ dimension (gray) with the same stream rate as the measurement stream (white) as shown in (see Figure 4). The data item is not only defined by its numerical values, but further described by its DQ information.

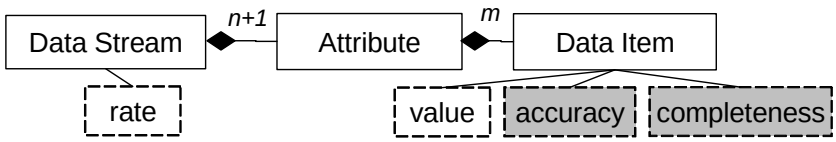

Figure 4: Naive DQ Annotations

A sensor data stream D of length m and rate r consists of n+1 Attributes A_i ($0 \leq i \leq n$), where A_0 represents the timestamp t of the sensor data stream. Each timestep t_j ($0 \leq j \leq m$) indicates a tuple T_j with n measurement values $v_i(j)$. For the naive DQ annotations every measurement value $v_i(j)$ is enhanced with a data quality vector $q_i(j)$ enclosing d data quality dimensions (e.g. accuracy $a_i(j)$ and completeness $c_i(j)$.).

$$v'_{ij} = \{v_{ij}, \vec{q_{ij}}\}$$
$$\vec{q_{ij}} = \begin{pmatrix} a_{ij} \\ c_{ij} \end{pmatrix}$$

Figure 5 shows an extract of sensor measurement data combined to calculate the residual lifetime of a truck's engine. Every 10 days, the residual lifetime is estimated. It is calculated based on several sensors (i.a. oil pressure, oil temperature, mileage, number of cold starts) with the given data quality dimensions accuracy and completeness. Analogue to the sensor measurements, the sensors' data quality information were combined and aggregated to compute the quality of the residual lifetime.

Timestamp	...	210	220	230	240	250	260	270	280	290	300	310	320	330	340	350	360	370	380	390	400	...
Lifetime	...	300	298	295	292	292	292	292	283	274	265	255	252	250	242	233	206	195	190	187	184	...
Accuracy	...	3.5	3.5	2.9	2.1	3.0	2.7	4.2	5.1	3.8	3.7	2.3	2.6	1.9	3.7	3.4	2.9	2.7	3.6	3.2	1.9	...
Completeness	...	0.9	0.9	0.8	1	0.9	0.85	0.8	0.75	0.8	0.8	0.9	0.9	0.9	0.8	1	1	1	1	1	1	...

Figure 5: Lifetime with Naive DQ Annotations

Obviously, this approach significantly increases the data volume, which is multiplied by the number of considered DQ dimensions. The additional data volume S to transfer data quality results in $S = m \cdot n \cdot d$. Hence, this approach is not suitable for those applications with stringent resource constraints and should only be employed in case that communication costs for data transmission are not significant.

3.2 Jumping DQ Windows

To reduce the additional data volume to transfer data quality information in a data stream, we propose the usage of jumping data quality windows. The concepts of our solution focuses on

flexibility, which is achieved through by an unlimited number of supported DQ dimensions, a variable window size and adaptable aggregation functions to summarize the window data quality.

During the data transfer in the PROMISE middleware, the result of a subscription to data items of a certain sensor node can be understand as a data stream. Therefore, in this section jumping DQ windows are introduced in the data stream metamodel. The traditional DSMS metadata model has to be extended. As already mentioned, a sensor data stream D consists of n attributes A_i $(0 \leq i \leq n)$ representing sensor measurements. In the traditional metadata model, each attribute A_i is associated with an unrestricted number of data value items $v_i(j)$. At this place the model extension is inserted. The notion of jumping windows is interposed in the relation between attribute and data item as shown in Figure 6: DSMS Metadata Model Extension.

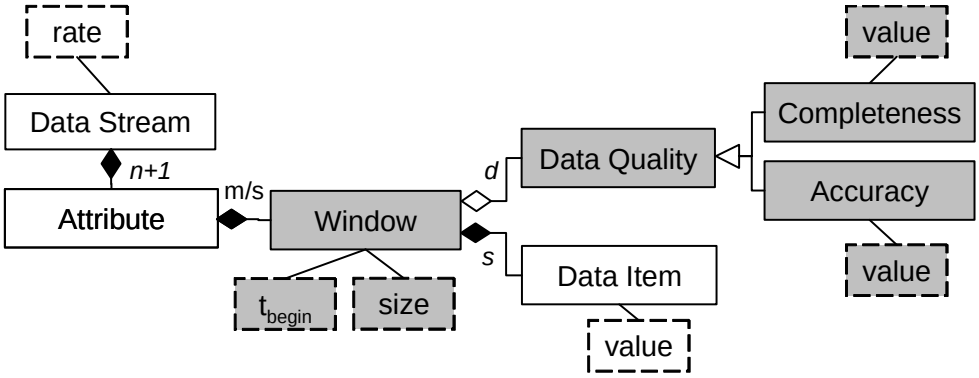

Figure 6: DSMS Metadata Model Extension

For the jumping window based annotations, the data quality information is not sent together with every single data item but window-wise for each DQ dimension. The additional data volume is reduced to an acceptable degree by aggregating the data quality for each attribute A_i $(0 \leq i \leq n)$ in jumping stream windows $w_i(k)$ of the given size s_i starting at timestamp $t_{begin} = t_k$. Thereby, the aggregation functions can be flexible determined for each DQ dimension corresponding to the underlying application. The attribute A_0 represents the timestamp, not a sensor measurement and thus is not equipped with data quality information.

The definitions in the following hold for each attribute A_i. For the sake of simplicity, we refer to the windows $w_i(k)$ as windows $w(k)$ of size s, etc. The window $w(k)$ includes s sensor data items $v(j)$ $(k \leq j \leq k+s-1)$ as well as the data quality vector $q(k)$ describing d data quality dimensions. The vector $q(k)$ represents the aggregated data quality information $q(j)$ $(k \leq j \leq k+s-1)$, which was associated with each data item (see section 3.1).

The vector function f incorporates the aggregation functions f_l $(1 \leq l \leq d)$ for all enclosed data quality dimensions. In the following the data quality vector is shown for $d = 2$, including the window accuracy $a_w(k)$ and window completeness $c_w(k)$.

$$\vec{q}_i(k) = \begin{Bmatrix} a_i(k) \\ c_i(k) \end{Bmatrix}$$

There are several possibilities to calculate the window accuracy $a_w(k)$ for a window $w(k)$ of size s starting at timestep $t(k)$. The accuracy function f_a is not fixed, but rather can be adjusted to the application's requirements. At this point, the quality propagation model is configured as generic as possible to be adaptable for every possible use case.

$$a_w(k) = f_a(a(j)|k \leq j \leq k+s-1)$$
$$e.g.\ a_w(k) = avg(a(j)|k \leq j \leq k+s-1)$$

The window accuracy a_w can be defined as the (weighted) average or sum of all value accuracies *a(j)*. Moreover, it may be defined as the minimum or maximum of the absolute error in the corresponding window. Section 2.2.3 introduces the confidence as data quality dimension describing the statistical error resulting from sampling operations. The calculation of ε is defined according on Haas [Haa97]. Based on the given function, the window confidence $\varepsilon_w(k)$ for a window starting at *t(k)* with window size s is declared by

$$\epsilon_w(k) = \frac{d \cdot \sigma(j|k \leq j \leq k+s-1)}{\sqrt{s \cdot r_s}} \cdot \sqrt{1-r_s}.$$

The window completeness $c_w(k)$ describing the window *w(k)* is stated as the ratio of originally measured, not interpolated values *v'(j) (k ≤ j ≤ k+s-1)* compared to the window size s.

$$c_w(k) = \frac{count(v'(j)|k \leq j \leq k+s-1)}{s}$$

With the knowledge of the window completeness $c_w(k)$, each data value in the respective window [*k* ; *k+s*-1] has the probability $c_w(k) \cdot 100\%$ to be an originally measured data item and not an interpolated one.

Now, we have all information needed to model the quality dimensions accuracy (systematic as well as statistical) and completeness for sensor data streams. Every *s* timesteps, the window DQ vector $q_w(k)$, consisting of window completeness $c_w(k)$, window accuracy $a_w(k)$ and window confidence interval $\varepsilon_w(k)$ is calculated.

For $s = 1$, $q_w(k)$ corresponds to the quality of a single data item. If $s = n$, the data quality is not calculated and propagated until the complete data has been propagated through the PROMISE middleware to the PDKM. Hence, the choice of the window size *s* presents the trade-off between the communication effort and the data quality granularity shown to the user.

The number of data quality dimensions is not fixed but variable for each attribute. Further, the window size *s* can be defined independently for each stream attribute. The additional memory space to cover d_i data quality dimensions for each of *n* attributes A_i depends on the attributes' window size s_i and the stream length *m*.

$$S = m \cdot \sum_{i=1}^{n} \frac{d_i}{s_i}$$

Figure 7 shows the resulting data quality for the residual lifetime of the truck engine. The DQ information provided for each data item (see Figure 4) are aggregated in jumping windows of size s = 5. Compared to the naive DQ annotation the gained resource saving is clearly visible.

Timestamp	...	210	220	230	240	250	260	270	280	290	300	310	320	330	340	350	360	370	380	390	400	...
Lifetime	...	300	298	295	292	292	292	292	283	274	265	255	252	250	242	233	206	195	190	187	184	...
Accuracy	...	3.0					3.3					2.78					2.86					...
Completeness	...	0.9					0.8					0.9					1					...
		t_{begin}=210					t_{begin}=260					t_{begin}=310					t_{begin}=360					

Figure 7: Lifetime DQ in Jumping Windows

The window-wise calculation of the data quality dimensions may be executed at the embedded intelligent device the sensor is connected to or at every other point in the data stream system. However, to be as efficient as possible, the DQ aggregation should take place as near to the sensor as possible.

3.3 DQ in the System Object Model

In this section the metamodel extensions described above shall be integrated in the system object model defining the data schema of the PROMISE PDKM. The object model is described in detail in the DR 9.2 "*Specification of the System Object Model*" [DR9.2]. To allow for the incorporation of (sensor) data quality information, the section 4.2.2 "*Life cycle related information: description of life cycle phases and the related field data*" will be further analyzed. Figure 8 shows the part of the system object model focusing on the handling of field data.

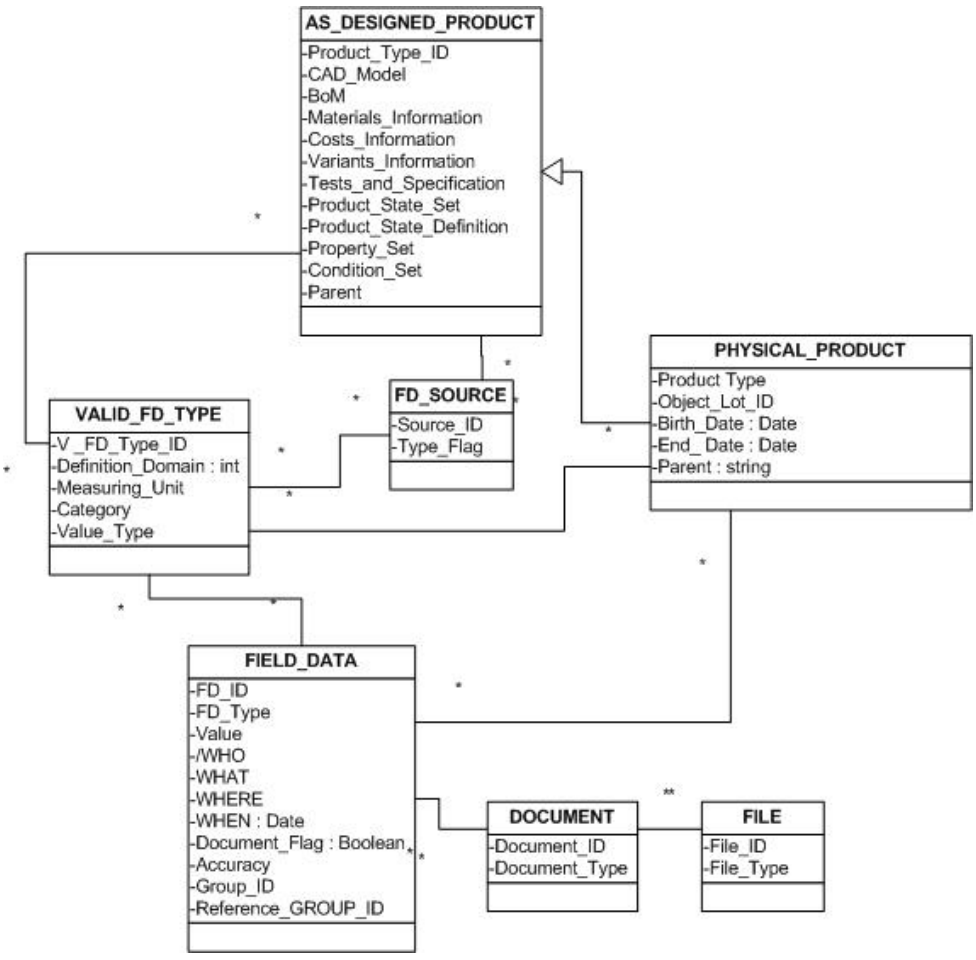

Figure 8: The FIELD_DATA class and its relationships with other model components

The FIELD_DATA class (Figure 8) is a crucial class in the semantic model, in the sense that it enables the overall PROMISE system to collect data from the field with the help of e.g. PEID-equipped sensors. Field data can be of different types (VALID_FD_TYPE class), and is collected by means of sources like e.g. sensors (FD_SOURCE class). It might be organized in documents (DOCUMENT class) with attached physical files (FILE class). The associations among these

classes show the most important existing links; the cardinalities are all zero to many, to take into account the most general cases.

Important attributes belonging to the FIELD_DATA class are:

- *FD_ID*: This attribute represents the identifier of each single field data record. The cardinality is one and only one.
- *FD_Type*: This attribute shows the type of the field data, and directly corresponds to a specific object of the VALID_FD_TYPE class, i.e. the string defining the FD_Type must be equal to the string defining the ID of an object of the VALID_FD_TYPE class. The cardinality is also one and only one.
- *Document_Flag* (Boolean): This attribute states if the field data is contained in or represented by an attached document.
- *Value*: This contains the value of the field data record. The cardinality is zero to one depending on the *Document_Flag*.
- *Accuracy*: With cardinality zero to one, this attribute states the accuracy of the field data measurement if needed.
- *WHO*: This attribute shows who is responsible for the field data measurement, i.e. which is the source of the field data, and can be derived e.g. from a corresponding object of the FD_SOURCE class linked with the object of the VALID_FD_TYPE class that is associated to the present FIELD_DATA object.
- *WHAT*: This attribute can for instance explain what the field data stands for, i.e. the meaning of the data itself.
- *WHERE*: This attribute states the location where the measurement was made, if needed by the specific application scenario (the cardinality is zero to one), i.e. the location where the product was situated when the present field data was collected.
- *WHEN* (Date): This is simply the timestamp indicating when the field data was recorded.

The VALID_FD_TYPE class is intended to model the information concerning the type of a given field data object e.g. “Temperature_Sensor_1” or “Average_Temperature_Sensor_2”, its set of categories e.g. {”Temperature Measurements”, “Calculated Values”}, its measuring unit, e.g. K, W, N, kg, m, etc., and the data type, e.g. integer, double, float, string etc. These pieces of information are provided all with cardinality of one and only one or zero to one depending on the data type. The VALID_FD_TYPE class is associated to both the PHYSICAL_PRODUCT class (representing the physical instance of the product) and to the AS_DESIGNED_PRODUCT class (representing the BOL as-designed structure of the product). Both associations have zero to many cardinalities, to state that a field data type might be defined on the one hand on the level of a product type, and so is also inherited by all derived physical product items, and on the other hand for a specific product item (instance) only.

The FD_SOURCE class is used to define the identity of the source of a given valid field data type. The *Source_ID* is the ID of the source of field data; for instance, it can represent the identity of the sensor of the PEID being responsible of the measurement of that specific field data, e.g. "sensor #5 linked to the on-board computer", which performs measurements over time of the temperature at which the product operates.

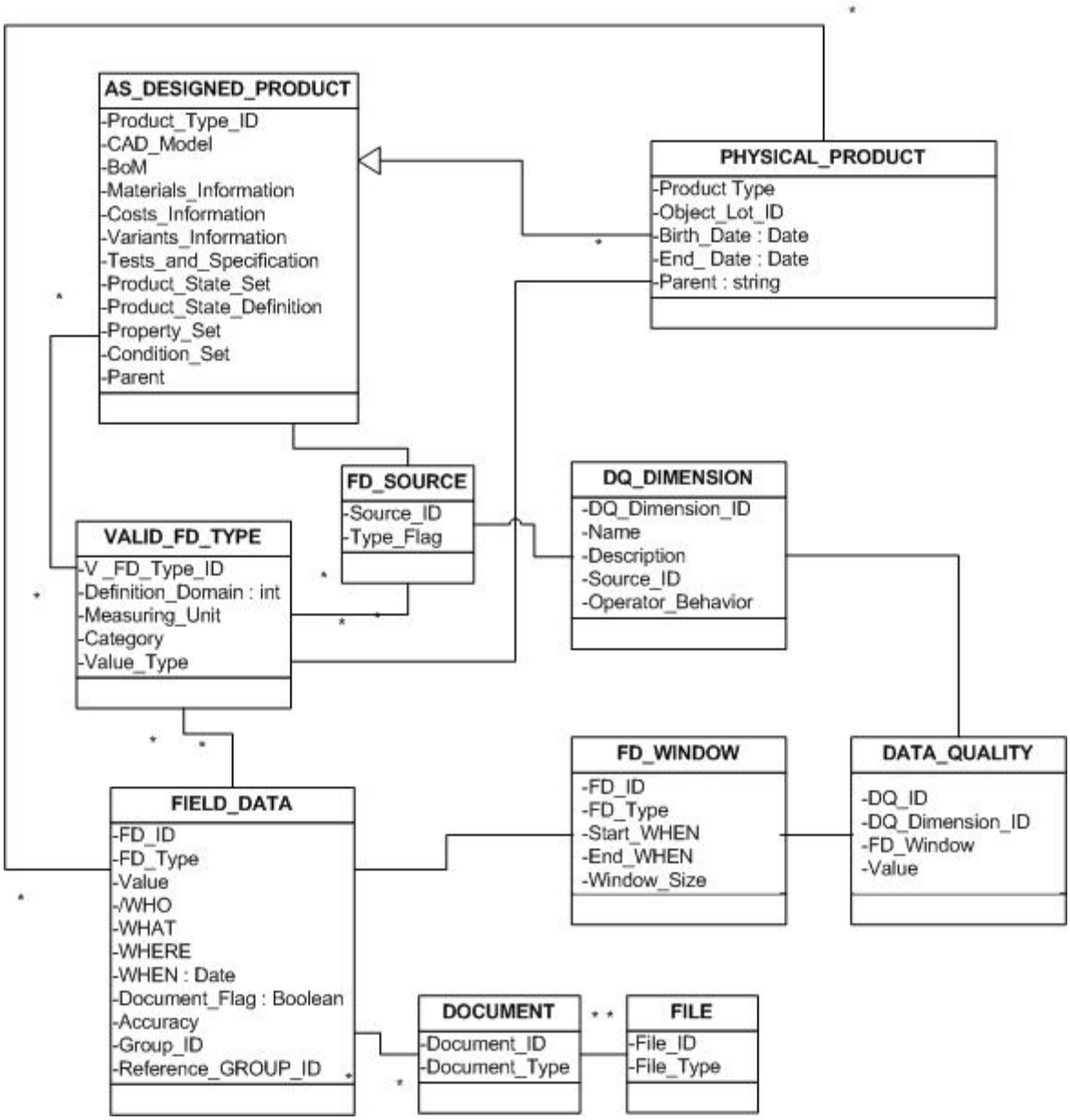

Figure 9: The DATA_QUALITY classes in relation to the FIELD_DATA class

The FD_WINDOW class (Figure 9) represents a jumping data quality window (see section 3.2) containing a set of temporally consecutive field data measurements. It constitutes a logical set of field data information of a single data source and thus a certain field data type (FD_Type). The boundaries of the data quality window (FD_WINDOW) are defined by the starting timestamp and the end timestamp or the window size, respectively.

The attributes belonging to the FD_WINDOW class are:

- *FD_ID*: This attribute represents the identifier of each single field data record. The cardinality is one and only one.
- *FD_Type*: This attribute shows the type of the field data, and directly corresponds to a specific object of the VALID_FD_TYPE class, i.e. the string defining the FD_Type must be equal to the string defining the ID of an object of the VALID_FD_TYPE class. The cardinality is also one and only one.

- *Start_WHEN*: This attribute defines the starting timestamp of the field data quality window.
- *End_WHEN*: This attribute defines the end timestamp of the field data quality window. While using the end timestamp to determine the field data set contained in FD_WINDOW, the window size is defined time-based. Due to fluctuating data rates, the field data sets contained in the FD_WINDOWs may be of different size.
- *Window_Size:* This attribute gives the length of the FD_WINDOW. It defines the count of measurements contained in one FD_WINDOW. During this tuple-based window size definition every FD_WINDOW contains the same number of field data items.

As described in section 0 there are multiple data quality dimensions. Further it was stated, that accuracy, confidence, completeness build a basic set describing the quality of sensor data. However, the system object model extension shall go beyond these three characteristics. To allow for a generic metadata modelling, the extensions will support an unlimited number of DQ dimensions.

The class DQ_DIMENSION represents one specific data quality dimension. The n:m relation between DQ_DIMENSION and FD_WINDOW allows the introduction of an unrestricted number of dimensions. Compared to the object-oriented programming, the DQ_DIMENSION provides the class-view onto data quality information.

The attributes belonging to the DQ_DIMENSION class are:

- *DQ_Dimension_ID*: This attribute defines a unique identifier of the respective DQ dimension.
- *Name*: This attribute gives the name of the DQ dimension, for example accuracy, completeness or timeliness.
- *Description*: Because there exist different definitions for data quality and data quality dimensions this attributes gives a detailed description of the respective dimension.
- *Source_ID*: This attribute defines the source (FD_SOURCE) of the field data.
- *Operator_Behavior:* This attribute defines the behaviour of data processing operators on values of the respective data quality dimensions. During the data processing, the data quality can be carried along by retracing the processing operations analogue on the DQ information.

The DATA_QUALITY class represents the value of a specific data quality dimension associated to a certain FD_WINDOW. It gives the object-wise view onto data quality information.

The attributes belonging to the DATA_QUALITY class are:

- *DQ_ID*: This attribute defines a unique identifier of the respective DQ value.
- *DQ_Dimension_ID*: This attribute gives the reference to the data quality dimension the DQ value describes.
- *FD_Window*: This attribute gives the reference to the field data set defined by WD_WINDOW the DQ value describes.
- *Value:* This attribute gives the parameter value of the referenced data quality dimension.

4 Data Quality Processing

In this section the influence on data quality of different basic operators in data stream processing is analyzed. Mathematical functions are introduced to compute the effects of different stream operators on the DQ dimension completeness, accuracy and confidence. Thereby, we focus on the following operators:

- Data reduction
  - Sampling
  - Aggregation
- Stream combination
  - Join
  - Branching
- Stream value computation
  - Algebraic operators
  - Threshold control

First, the sampling is analyzed. It provides an efficient data reduction and further is required to allow for the join of two data streams. To prevent from aliasing due to sampling, the low-pass filtering of data streams is introduced. Then, the aggregation as another tool to reduce the stream volume is examined. The join allows for the combination of two distinct data streams, whereas the branching enables the re-use of different paths in the model graph. Finally, algebraic operators such as addition or square root are analyzed.

4.1 Sampling operators

The downsampling reduces the data volume of the stream while randomly skipping a given amount of data items. To allow the correct reconstruction of the original signal from the sample, aliasing effects have to be prevented. Therefore, the low-pass filter which eliminates high frequencies in the signal stream is introduced. At the end of this section, the up-sampling (interpolation) as an important premise for stream joining is analyzed.

Downsampling

This data stream operator creates a systematic sample of the incoming data stream. The downsampling is a tuple-based operator to reduce the stream size by inserting every k-th tuple of the incoming stream into the resulting sample. The parameter k defines the sampling rate r_s and thus the resulting stream rate r' based on the former stream rate r.

$$r_s = \frac{1}{k}$$
$$r' = \frac{r}{k}$$

To guarantee a uniform sampling, the first data tuple sampled from the stream is not simply the first in the list, but is chosen randomly as the i-th tuple, where $1 \leq i \leq k$. Only then, each element in the population has a known and equal probability of selection $p = 1/k$. This makes systematic sampling functionally similar to the simple random sampling, while benefiting from the efficient sample creation. However, this sampling technique is vulnerable to periodicities in the data stream. If periodicity is present and the period is a multiple of k, then a biased sample will result.

Furthermore, one should clearly keep in mind the sampling theorem stating that the sampling rate r_s must be larger two times the signal's highest frequency f_{max} [Kie05].

$$r_s > 2 \cdot f_{max}$$

If this condition is not met, aliasing effects occur and prevent the correct reconstruction of the signal.

The sampling operator has no direct influence on the completeness of a data stream. If each data item is sampled with equal probability $p = 1/k$, the fraction of original sensed and interpolated data values does not change. However, similar to selection or aggregation the sampled values build up new windows, where the window completeness c_w is the average of the data items' former completeness values.

The new window accuracy a_w is computed analogue to the window completeness. The average of the incoming data accuracies determines the resulting window accuracy a_w. The statistical error introduced by the information loss during the sampling of this parameter is given in section 2.2.3. If the incoming ε_w are not null due to prior sampling steps, the new window confidence ε'_w consists in the squared average of the former ε_w of the tuples composing the corresponding window plus the newly introduced statistical error.

$$\epsilon'_w(k) = \frac{\sqrt{\sum_k \epsilon_w(k)^2}}{s} + \epsilon'_{w,new}(k)$$

Low-Pass Filter

Using a low-pass filter signal inherent high frequencies are eliminated to smooth the signal as shown in Figure 10 to avoid aliasing effects during sampling operations. Thus, not only the influence of the sampling operator, but as well the affect of the low-pass filter on data quality has to be analyzed. The low-pass filter can be described with the help of the Laplace transformation [Kie05], which transforms the signal *x(t)* to its frequency domain *X(f)*.

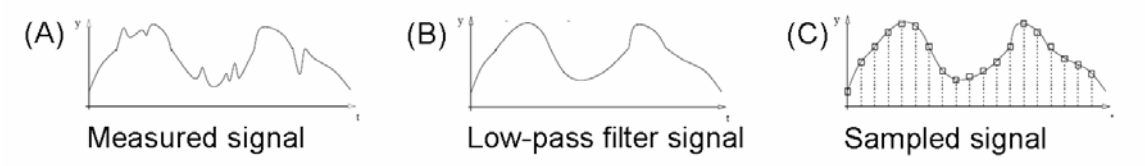

Figure 10: Low-pass filtering and sampling

After decomposing the sensor signal represented by the data stream into the spectrum of its frequency components, the bands with higher frequency than the threshold given by the requested sampling rate are cut off. In the last step the signal is reconstructed based on the remaining frequency components.

The low-pass filtering does not affect the level of completeness of the data stream. The number of measurement values is not reduced, but the stream rate is constant. Thus, the window completeness c_w remains constant. The parameters window accuracy a_w and window confidence ε_w define the absolute systematic and statistical error of the data stream signal, respectively. Hence, the data stream signal *x(t)* is composed of the true signal $x_{true}(t)$ and the absolute error $\Delta x = a_w + \varepsilon_w$. For Δx is not dependent on the time *t*, it is not affected by the Laplace transformation, but remains constant. Thus, the low-pass filter described by the Laplace transformation does not affect the window accuracy neither the window confidence.

Upsampling

In contrast to the downsampling operator, which reduces the data stream by the factor *k*, the upsampling or interpolation operator interposes a determined data item between each tuple pair and thus stretches the stream. The linear interpolation with interpolation rate $r_i = 2$ doubles the data stream length. Between every tuple pair $< T_j >;< T_{j+1} >$, a new tuple of the form

$$\left\langle \frac{t(j)+t(j+1)}{2}, \frac{v(j)+v(j+1)}{2} \right\rangle (1 \leq j \leq m)$$

is introduced. If $< T_j >$ and $< T_j{+}_1 >$ associate with different windows *w(k)* and *w(k+1)*, the new tuple is defined to belong to the preceding window *w(k)*. The completeness of a window is reduced, while it is bloated with interpolated data. After the interpolation, the new window completeness $c'_w(k)$ consists in

$$c'_w(k) = \frac{c_w(k)}{r_i}.$$

According to the data value *v(j)*, the accuracy and confidence are linearly interpolated for each data item based on the tuples' window accuracies $a_w(k)$ and $a_w(k+1)$ as well as confidences $\varepsilon_w(k)$ and $\varepsilon_w(k+1)$, respectively. Hence, the new window accuracy a'_w and window confidence ε'_w are defined as follows.

$$a'_w(k) = \frac{(s \cdot r_i - 1) \cdot a_w(k) \cdot \frac{a_w(k)+a_w(k+1)}{2}}{s \cdot r_i}$$

$$\epsilon'_w(k) = \frac{(s \cdot r_i - 1) \cdot \epsilon_w(k) \cdot \frac{\epsilon_w(k)+\epsilon_w(k+1)}{2}}{s \cdot r_i}$$

Example Application: Oil Condition Monitoring

We will refer to the example application of oil condition monitoring to illustrate the influence of the processing operators on data quality. During the condition monitoring the oil ageing is calculated based on six sensor measurements: pressure, temperature, particle contamination, humidity, viscosity and permidity [Die07], which are recorded with different stream rates due to distinctive sensor capabilities. To join these sensor streams for complex evaluations, the measuring frequencies have to be aligned with the help of up- and downsampling. For example, the pressure stream (sensor rate $r = 2$/min) is downsampled with sampling rate $r_s = 0.05$ as shown in Figure 13 a), whereas Figure 13 b) illustrates the interpolation ($r_i = 2$) of the particle contamination measurements (sensor rate $r = 0.05$/min) describing the water saturation of the oil. The information loss due to downsampling is represented in the DQ dimension confidence ε_w. The interpolated humidity values are mirrored in the increasing window completeness c_w.

a)

Timestamp	…	10.7	10.8	10.9	11.0	11.1	11.2	…
Temperature	…	44.9	55.4	58.0	58.1	54.9	60.2	…
a_w				0.6			0.6	
ε_w				0			0	
c_w				0.77			0.87	

Down-Sampling

Timestamp	…	70	80	90	100	110	120	…
Temperature	…	59.4	43.9	58.7	45.2	55.5	49.6	…
a_w				0.6			0.6	
ε_w				0. 45			0.12	
c_w				0.94			0. 75	

b)

Timestamp	…	140	160	180	200	220	240	…
Part.Contam.	…	14.1	13.4	12.0	11.4	11.1	13.2	…
a_w				0.5			0.5	
ε_w				0			0	
c_w				0.69			0.87	

Up-Sampling

Timestamp	…	170	180	190	200	210	220	…
Part.Contam.	…	12.7	12.0	11.7	11.4	11.25	11.1	…
a_w				0.5			0.5	
ε_w				0			0	
c_w				0.345			0. 435	

Figure 11: Down- and Upsampling

4.2 Aggregation

With the help of an aggregation operator data stream values can be summarized. The basic aggregation operators are sum(), average() and count(). Further, complex operators like standard deviation or variance can be built up as a composition of those and therefore will not be discussed in this paper.

There are two modes to execute the aggregation. On the one hand, the complete data stream can be aggregated to one output value. However, this application is rare. The usual practice involves grouping. Here, data items are grouped to data sets $g \in G$, which are then aggregated. In this way, the data stream is not reduced to one single value, but compressed to $|G|$ data tuples representing the groups. A second application scenario is given by moving aggregations. Both applications of the aggregation operator are supported by the DQ computation presented in this section.

The grouping can be applied based on any data attribute. However, we only discuss the timestamp based grouping in this paper, because grouping respective to another attribute would block the pipeline in the data stream processing. The grouping based on the timestamp can be applied to build groups a) representing a given time interval (e.g. 10min) or b) consisting of a given number of data tuples (e.g. 100 tuples). Thus, the grouping separates the data stream in $|G|$ intervals with a) varying lengths l_i depending on the respective stream rate or b) a fixed length $l = |g|$ equal for all intervals. It should be noted, that the timeframe defining the grouping for an aggregation operator is completely independent from the window size s for data quality calculation.

During aggregation, a set of data items is summarized to form a new single data value. This data value now represents not only a certain point in time, but a time interval. The timestamp has to be adjusted to reflect this fact. After an aggregation the data stream tuples have timestamps of the form $[t_{begin}; t_{end}]$, where $t_{end} = t_{begin} + l_i \ (1 \leq i \leq |G|)$. For the DQ processing during aggregation two steps have to be distinguished. First, the data quality of one aggregate value is calculated based on all incoming tuples' DQ information, whereas the underlying function depends on the aggregation type as well as the DQ dimension. Then, the resulting aggregates are bundled to form new windows of size s. The second step is generic for all DQ dimensions.

The aggregated window completeness is independent of the applied aggregation operator type. The completeness of one aggregate value is calculated as the average of all incoming tuples' window completeness values c_w. In contrast to the completeness calculation, during the examination of aggregate accuracy and confidence interval, it has to be distinguished between the different aggregation types.

With regard to the computation of the aggregate accuracy and confidence, the aggregation sum() can be considered similar to the algebraic addition (see section 4.4). The aggregate accuracy and confidence, respectively, consist in the linear/squared summation of all incoming tuples' window accuracies/confidences weighted with the linear/squared partial derivative. For the addition of l_i data items the partial derivates with respect to each of these variables are equal to 1. Hence, the function for the accuracy and confidence of an aggregate can be reduced to

$$a'_v(j) = \sum_{p=1}^{l} a(v(j+p))$$

$$\epsilon'_v(j) = \sum_{p=1}^{l} \epsilon(v(j+p)).$$

To compute the average, the sum of all data values is divided by the size of the group l_i. Thus, the aggregate accuracy a_{avg} and confidence interval ε_{avg} can be easily derived from the aggregate accuracy and confidence interval for sum(), respectively. The aggregation count() is not useful in data streams comprising numerical, continuous data values and thus will not be regarded at this point.

After the data quality vector $q_i(k)$ has been calculated for every resulting aggregate value, the new windows have to be built up. The window DQ vector consists in the averaged aggregate DQ vectors.

$$\vec{q}(k) = \frac{\sum \vec{q}(i)}{s} (k \leq i \leq k + s - 1)$$

Figure 12 shows the aggregation of a sensor data stream in the oil condition monitoring. To estimate the expired and thus derive the residual oil lifetime based on the given six sensor data streams (see section 4.1) the data stream is cumulated along the timescale.

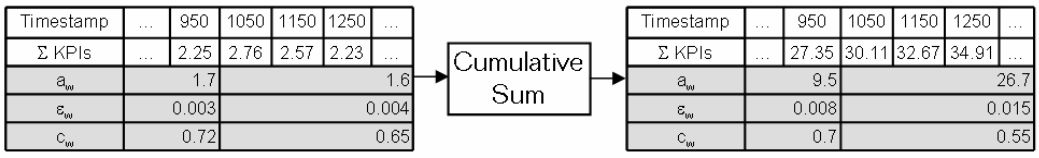

Figure 12: Aggregation - cumulative sum

4.3 Joining and Branching

In this paper we focus on the timestamp-based joining of two sensor data streams. In [Sch05] a temporal join technique is introduced, which uses up- and downsampling (see section 4.1) to find one-to-one join partners in streams of different data rates, i.e. bandwidth.

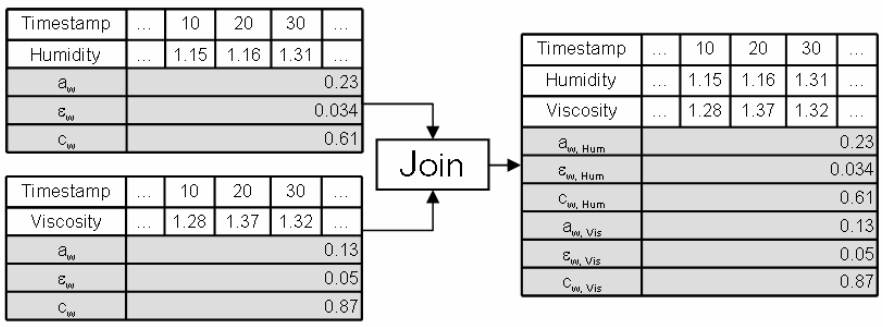

Figure 13: Data stream join

After data rate adaptation, each tuple of stream D_1 finds a partner with the same timestamp in D_2 as shown in Figure 13: Data stream join. Hence, the join can easily be executed producing a data stream with the adapted data rate. During the join operator, data quality dimensions are not affected, but copied to the resulting data stream.

4.4 Algebraic Operators

In contrast to traditional data base management systems algebraic operators play an important role in data stream applications. Sensor data streams comprise numerical data items, on which algebraic algorithms are applied for analysis and data evaluation.

With regard to completeness we distinguish between unary and multi-valued operators. A unary operator, for example the square root, is applied to one attribute of a data stream. Here, the completeness is not affected. It is constant for all windows *w(k)*. A multi-valued operator combines two or more attributes and reports the calculation result in a new attribute.

$$A = f(A_i | 1 \leq i \leq n)$$

The basic binary operators are addition, subtraction, multiplication and division, which can be combined to complex functions. In this case, the resulting completeness depends on the completeness of the input attributes. It is calculated as the average of the preceding completeness values.

$$c'_w = avg(c_{w,i} | 1 \leq i \leq n)$$

The window accuracy a'_w and confidence ε'_w are computed according to the error propagation framework used in scientific experiments. In this context a distinction is drawn between statistical and systematic errors. Statistical errors are summarized according to the Gaussian error propagation [Pap06]. The window confidence intervals $\varepsilon_{w,i}$ are summed up squared, weighted with the partial derivative with respect to the corresponding attribute.

$$\epsilon'_w = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial A}{\partial A_i} \right)^2 \cdot \epsilon^2_{w,i}}$$

On the other hand, the calculation of the systematic error is executed as a linear addition. However, similar to the Gaussian error propagation the absolute errors are weighted with the partial derivative.

$$a'_w = \sum_{i=1}^{n} \left| \frac{\partial A}{\partial A_i} \right| \cdot a_{w,i}$$

During the oil condition monitoring the key performance indicators humidity, viscosity and permedity, dependent on pressure and temperature, as well as the particle contamination are summed up weighted to estimate in the first step the expired lifetime and thus derive the residual oil lifetime.

$$lt_e = 5 \cdot hum + 4 \cdot vis + 7 \cdot perm + 2 \cdot cont$$

The equations for the window accuracy a_w and confidence ε_w are thusly derived as follows

$$a'_w = 5a_{w,h.} + 4a_{w,v.} + 7.5a_{w,p.} + 2a_{w,c.}$$
$$\epsilon'_w = \sqrt{25\epsilon^2_{w,h.} + 16\epsilon^2_{w,v.} + 56.25\epsilon^2_{w,p.} + 4\epsilon^2_{w,c.}}$$

where $a_{w,i}$ and $\varepsilon_{w,i}$ define the former window accuracy and confidence of the attributes humidity, viscosity, permidity and contamination, respectively. Figure 14 shows the addition of weighted humidity and viscosity. In this example, the attributes provide aligned windows with similar the starting point. However, the presented model supports windows shifted against each other or with different window sizes, too.

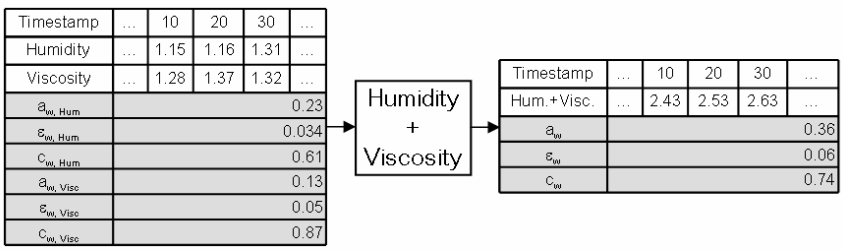

Figure 14: Addition of weighted humidity and viscosity

4.5 Threshold Control

In many applications for example in controlling systems or predictive maintenance, a data stream is monitored to detect exceeded thresholds. Such thresholds may be given as fixed values or consist in parameters defined with the help of other sensors included in the system. Thus, not only the imprecision of the monitored data stream, but also errors inherent to the threshold have to be examined.

If the jumping DQ window is not complete ($c_w < 1$) the evaluation of the value range can not be determined with certainty. It is not possible to estimate, whether the missing values exceed or remain within the given thresholds. Thus, the degree of incompleteness has to be forwarded to the output-stream of the threshold operator. The window completeness remains constant $c'_w = c_w$.

During the threshold evaluation based on imprecise data, the accuracy a_w and confidence ε_w both of threshold function b and measurement values v have to be taken into account. Due to these inherent errors it is not always possible to define a clear threshold exceeding or shortfall. Rather, an uncertain interval $[b - \delta; b + \delta]$ around the threshold b is defined where

$$\delta = a_{w,b} + a_{w,v} + \epsilon_{w,b} + \epsilon_{w,v}$$

to mirror the uncertainty and thus the inaccuracy of the operator output. The threshold operator function f: R → {0; 1} where 0 defines the uncritical value range and 1 indicates a threshold exceeding, has to be enhanced to identify tuples in the uncertain interval f': R → {0; 0:5; 1}[1].

Further, the degree of uncertainty in a DQ window is represented by the DQ dimension accuracy.

$$a'_v(j) = \left\{ \begin{array}{l} 0, v(j) \notin [b-\delta; b+\delta] \\ 0.5, v(j) \in [b-\delta; b+\delta] \end{array} \right\}$$

The window accuracy $a'_w(k)$ consists in the average of the value accuracies $a'_v(j)$ where $k \leq j \leq k+s-1$. The window confidence ε_w is incorporated into the definition of the uncertain range represented by the new window accuracy. Thus, the new window confidence ε'_w w is set to 0.

Here, the oil condition monitoring example is picked up again. A threshold to control the residual oil lifetime is applied. When the residual lifetime falls under 25% a maintenance notification is sent to the responsible service technician. Moreover, an alarm is raised, if the residual oil lifetime decreases further below 10%. Figure 15 shows the analysis of the notification threshold. The critical range around the threshold function is mirrored in the resulting signal.

[1] The output of the threshold operator is set to 0.5, if the input data lies within the uncertain interval $[b - \delta; b + \delta]$ around the threshold function b.

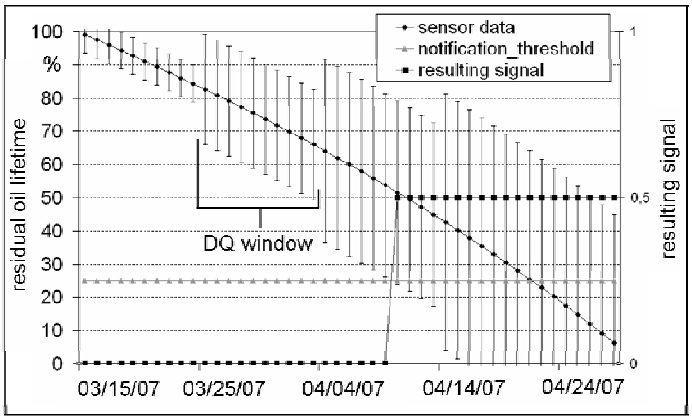

Figure 15: Threshold

The presented stream operators form a base for the processing of data streams. They can be combined to build processing environments from simple transfer paths with data reduction to complex networks where basic data analysis is executed.

5 Data Quality Visualization

The data quality visualization is a point of special interest. The results of data quality processing in the quality propagation model have to be appropriately presented to the user to allow for a fast, easy and comprehensive data evaluation. In this deliverable, the focus will lie on the visualization of different DQ dimensions

The QPM supports the data stream processing while tracking the influences of different process steps on the data quality. The model allows for one or more data sinks representing queries on the data stream either consumed directly by the target application (for example by showing the current system state on a control monitor) or stored in a persistent database for historical analysis. In the first case, data quality information is also consumed directly from the stream and visualized on the screen. In the second case, the quality information has to be stored in the database and represented to the user, when the data is queried later on. In both cases, the data quality has to be presented to the user in a understandable, straightforward way to allow for the fast and solid evaluation of query results.
Clearly, the quality has to be visualized in connection with the data described. Thus, the graphical representation of the sensor data or derived knowledge, as shown in Figure 16 at the example of a state counter, has to be extended to allow for the quality visualization, too. The extensions to visualize the data quality are exemplarily shown for a pressure data stream of a hydraulic cylinder.

Figure 16: Sensor data visualization in the PDKM

5.1 Accuracy and Confidence

The DQ dimensions accuracy and confidence describe absolute errors of the underlying data items. Similar to the threshold control, these absolute errors can be used to illustrate the range around the measured or computed data items, which include the true value with a given confidence possibility (see Section 2.2.3 and 0). Figure 17 shows exemplarily the upper and lower bound defining the interval around the measured pressure including the true pressure value.

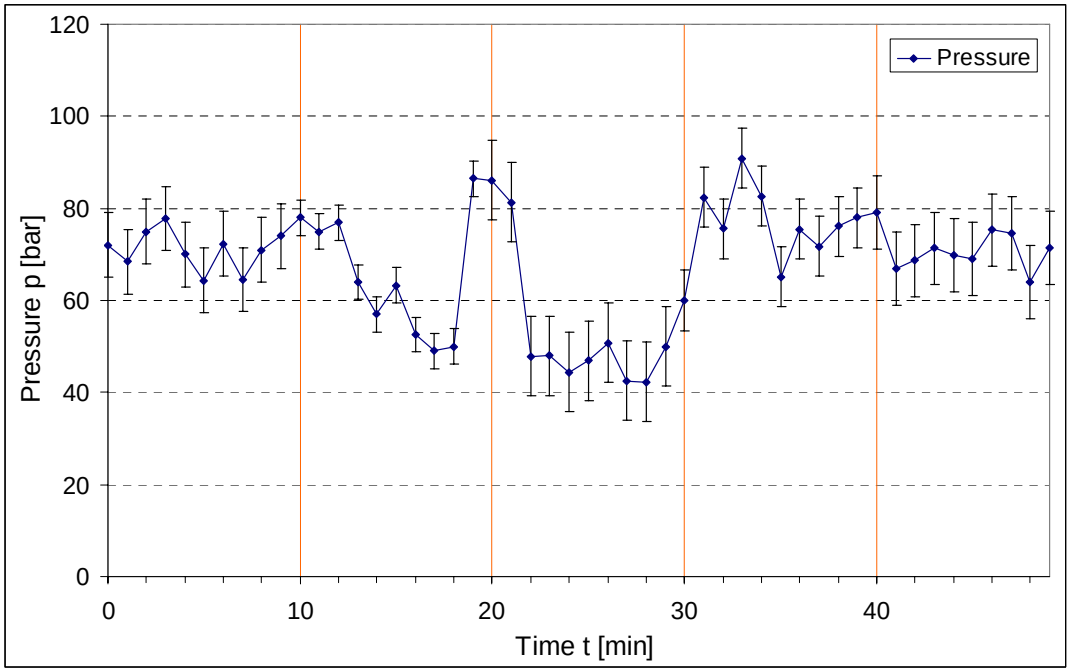

Figure 17: Visualization of accuracy and confidence

To enable a clear evaluation, zoom mechanisms as already provided for the time-axis can be introduced for the y-axis, too.

5.2 Completeness

The completeness of a data quality window can take values from 0 to 1 representing a completeness of 0% up to 100% originally sensed measurement data. Circle or pie charts are appropriate diagrams to illustrate such completeness rates. For example, a small pie chart can be attached to the measurement graph in timeseries diagram as shown in Figure 18.

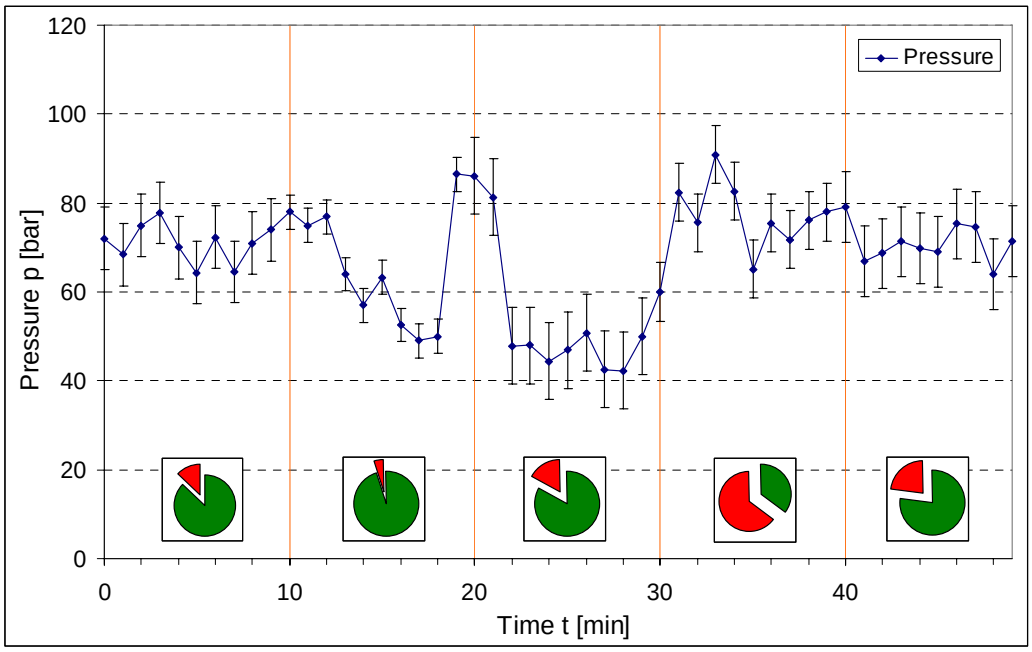

Figure 18: Visualisation of completeness

5.3 Other DQ dimensions

Section 2.2.1 introduces further data quality dimensions such as timeliness or reputation. Not all DQ dimensions listed there can be expressed numerically. For example, reliability or relevancy are better formulated verbally. For purpose of clarity, these DQ dimensions have to be structured for example in pop-up windows.

5.4 Quality Classification

For a better user-support, a quality classification can be introduced defining different levels of data quality. Each jumping data quality window is marked with its corresponding data quality level. For example, an absolute error of less than 4bar can be estimated as high quality, where the user can totally rely on the presented data. Data quality windows providing an absolute error 4bar $\leq a_w + \varepsilon_w \leq$ 8bar are defined as of mediocre quality, where the user has to take a closer look to unusual measurements and behavoir. Windows with an absolute error beyond 8bar contain data, which are no longer useable and have to be excluded from the further processing and data analysis. Quality classification cannot only be applied to DQ dimensions representing absolute data errors, but to all data quality dimensions.

The classification improves the possibilities for quality visualization. Different colors (e.g. green, yellow, red for accuracy and confidance) or saturations (e.g. for completeness) can be applied to mark DQ levels of the data stream as shown in Figure 19. Traffic lights are another possibility.

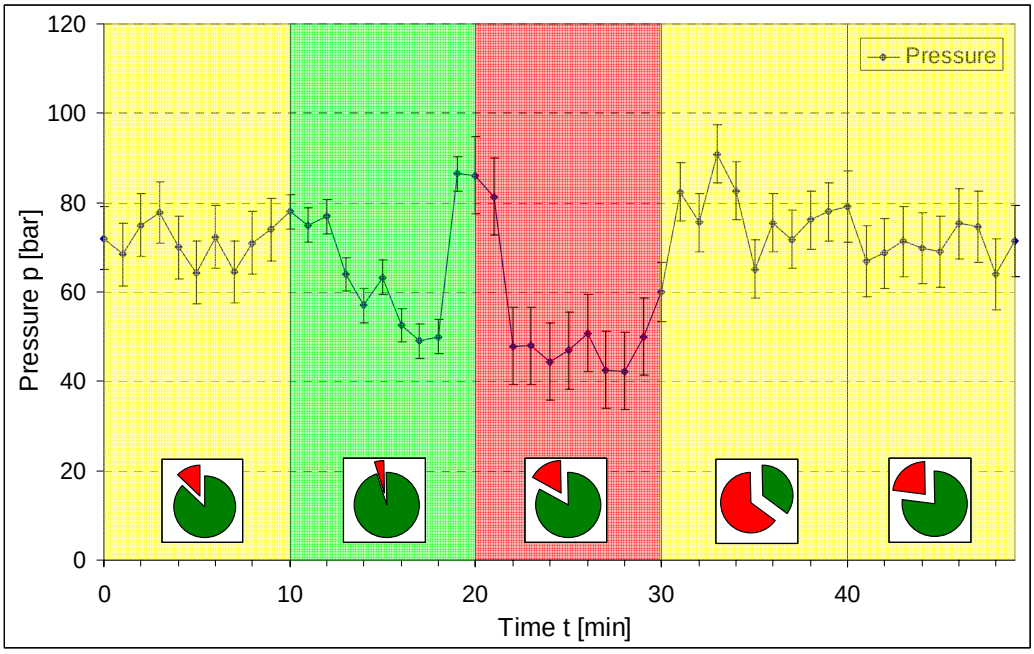

Figure 19: Application of quality classification

The listed visualization modes – especially the graphical ones - support the user during the data evaluation. Thus, the fast detection of data with low quality is enabled to diminish faulty business decisions.

6 Conclusions

In this deliverable the problem of restricted sensor data quality is discussed. The provision of data quality information is an important challenge in smart item environments, where various sensors are applied to support business decisions and automation processes. Thus, the discussion of data quality and the proposition of approaches for the capturing, propagation, storage and processing of data quality information is of high relevance in the context of the PROMISE project.

First, different definitions and classifications of data quality are introduced, comprising various data quality dimensions. In the context of sensor data, the deliverable focuses the DQ dimensions accuracy, confidence and completeness and gives an approach for their quantitative expression.

Moreover, an efficient way to model data quality in data streams is presented. Jumping data quality windows enable the resource saving propagation of data quality information from the sensors through the PDKM system up to the target applications.

To meet the resource constraints posed in smart items environments, data processing is essential to reduce the transferred data volume. Furthermore, operators retrieved from database querying are applied to extract complex knowledge from raw data. The analysis of these operators allowed to track their impact on various data quality dimensions. Hence, a comprehensive result evaluation is enabled and thus faulty business decisions are decreased. Though we referred to accuracy, confidence and completeness as three important DQ dimensions for sensor data streams, the proposed DQ metamodel is of generic structure to be easily extended by additional data quality dimensions.

Last but not least, different ideas for data quality visualization are presented to enable the comprehensive evaluation of data stored in the PDKM to prevent from faulty decisions due to imprecise sensor measurements with restricted data quality.

7 References

[Pro06] PROMISE DoW document, version 4.0, 2006

[Rah00] E. Rahm, Hong-Hai Do: Data Cleaning: Problems and current approaches. IEEE Data Eng. Bull. 23(4), 3-13, 2000

[Str97] M. Strong, Y. Lee, R. Wang: Data quality in context. CACM, 1997

[Haa97] Peter Haas: Large sample and deterministic confidence intervals for online aggregation. SSDBM, 1997

[Lee99] M. Lee, H. Lu.: Cleansing data for mining and warehousing. DEXA, 1999

[DR9.2] PROMISE DR9.2. Specification of the System Object Model, version 6.0, 2006

[Kie05] U. Kiencke, H. Jkel.: Signale und Systeme. Oldenbourg Verlag, 2005

[Die07] M. Dieter, F. Bauer.: Hydac condition monitoring – vom Sensor zum System. O+P - Zeitschrift für Fluidtechnik, May 2007

[Sch05] S. Schmidt, M. Fiedler, W. Lehner.: Source-aware join strategies of sensor data streams. SSDBM, 2005

[Pap06] L. Papula.: Mathematische Formelsammlung für Ingenieure und Naturwissenschaftler. Vieweg Verlag, 2006

