Key Performance Indicators in Data Warehouses [1]

Manfred A. Jeusfeld[1], Samsethy Thoun[2]

[1] University of Skövde, IIT, Box 408, Portalen, 54128 Skövde, Sweden
Manfred.Jeusfeld@his.se
[2] Pannasastra University, 313, Phnom Penh, Cambodia
samsethy@gmail.com

Abstract. Key performance indicators are a widely used tool to manage any type of processes including manufacturing, logistics, and business processes. We present an approach to map informal specifications of key performance indicators to prototypical data warehouse designs that support the calculation of the KPIs via aggregate queries. We argue that the derivation of the key performance indicators shall start from a process definition that includes scheduling and resource information.

Keywords: key performance indicator, data warehouse, business process

1 Introduction

Key performance indicators (KPI) evaluate the success of an organization or of a particular activity in which it engages (source: Wikipedia). They are used to continuously monitor those activities [1] in order to understand and control them. Deming [2] pioneered this field by statistically correlating independent process parameters to dependent performance indicators known as statistical process control (SPC). In SPC, the process parameters are kept in certain ranges such that the dependent variables such as KPIs or the product quality also remains in certain predictable ranges. These ideas were later also applied to software engineering [3], and to business process management [1]. Typical examples of KPIs are number of defects of a product, customer satisfaction with a service, the profit margin of a product, the percentage of deliveries before the promised delivery time, the machine utilization in a factory, and so forth. All these examples relate in some respect to an activity or to sets of activities. Moreover, they involve the interaction of multiple objects or subjects such as customers, employees, or machines.

In this paper, we investigate the relation of KPIs, data warehouses, and business process management. Specifically, we propose a guideline on deriving a prototypical data warehouse design from annotated KPI definitions, which themselves are derived from business process model fragments. This yields a top-down data warehouse design that strictly supports the calculation of the KPIs via aggregate queries.

[1] Part of the research was carried out while the second author was carrying out his master thesis project in the Erasmus IMMIT program at Tilburg University, The Netherlands.

A data warehouse consists of multi-dimensional facts representing measurable observations about subjects in time and space. The subjects, time and space are forming the dimensions, and the measures are representing the observations about the participating subjects. A data warehouse is essentially a large collection of measurements covering a certain part of the reality. In practically all instances, the measurements are about processes. If it were not, it would only provide a static account of objects in the reality. The key problem of this paper is *how to design the data warehouse from annotated KPI definitions such that the KPIs can be calculated by aggregate queries on the data warehouse.*

Another angle to KPIs is their summarizing nature. A KPI is not based on a single arbitrary observation but it aggregates a large number of observations about the same entities (or activities) to be statistically meaningful. The concept of an observation is the atomic building block of KPIs. Once the common properties of observations are set, one can start to collect such observations systematically and create the KPI on top of them. Different types of observations lead to different KPIs. So, given the definition of a KPI, what is the type of observations belonging to this KPI? KPIs can also be formed as expressions over other more simple KPIs. For example, the productivity of a process can be measured the division of a KPI on the output of the process divided by a KPI on the resources used for producing the output. Such KPIs are called *derived KPIs*. Since their computation is simple once the part KPIs are computed, we shall focus on *simple KPIs* that are not defined in terms of other KPIs but that are defined in terms of sets of atomic observations of the same type.

2 Related Work

Key performance indicators are a technique to quantify the performance of an organization or of its processes to achieve business objectives. In this chapter we view KPIs as used in conceptual modeling, in particular business process modeling, and in data warehousing.

2.1 Key performance indicators in conceptual modeling

Wetzstein et al. [1] investigate the definition of KPIs in the context of business process models, in particular from a service-oriented architecture perspective. Simple KPIs (called process performance metrics, PPMs) are the basis of more sophisticated, context-specific KPIs such as determining whether a customer has received the promised quality of service QoS (e.g. response time) can depend on the customer class and further parameters that we can view as dimensions of the KPI measurement. In their view a KPI is based on PPMs, a QoS definition, and a decision tree that determines whether a PPM measurement fulfills the QoS definition.

Strategic business modeling based on the Business Intelligence Model BIM [4] extends the goal modeling language i* by metrics linked to i* goals on the one side and tasks on the other side. The goals are monitored by the metrics and the tasks are the measures to achieve the goals. The metric interval is decomposed into performance regions (target, threshold/acceptable, worst value). The approach reminds of balance scorecards but extends it to the rich goal modeling language i*.

In software engineering, KPIs were introduced to manage the software development process [22], in particular in combination with the capability and maturity model CMMI [23]. Measurements such as the defect density in source code are used to control the software development process. Oivo and Basili´s goal-question-metric (GQM) approach [24] provides an informal guideline on which metrics need to be monitored in order to assess that a certain goal (like improving the software quality) is reached. A quality goal is decomposed in a set of quality questions, which is itself decomposed into a set of quality metrics. The metrics are comparable to KPIs. Hence, the GQM approach allows to group KPIs by the goals of stakeholders. An agreement on goals allows to focus only on those KPIs that are needed to assess to which extent the goals have been reached. The GQM approach highlights that metrics (and thus KPIs) should not be mixed up with goals. Nevertheless, quality goals are often formulated in terms of KPIs such as the average cycle time of a certain process must be below a certain threshold.

Statistical process control (SPC) [25] was introduced by Deming [2] and others into the manufacturing domain as a tool to monitor the production and product quality. Specifically, it measures parameters and establishes statistical correlations between the parameters (called variables in statistics). The correlations between variables are translated into a set of equations for predicting values for dependent variables from independent variables. The idea is to control the independent variables (such as the quality of input materials) at early stages of the production process in order to guarantee that the dependent variables (such as product quality parameters) are within a desired interval. The variables in SPC are comparable to KPIs.

2.2 Data Warehouse Design and KPIs

A central issue in data warehousing is to design appropriate multi-dimensional data models to support querying, exploring, reporting, and analysis as required by organizational decision making. DW design has received considerable research attention. However, there are different methodological approaches proposed by the literature. Some approaches are data-driven in the sense that they aim at deriving facts and dimensions from the structures of operational sources that are usually represented as Entity Relationship Diagrams (ERD) or Unified Modeling Language (UML) diagrams. The outcome of this approach is a set of candidate facts or even data schemas, among which only relevant ones are selected to include in DW systems. For instance, Golfarelli et al. [5] proposed the DW design approach based on E/R scheme. Golfarelli and Rizzi [6] also developed a data-driven method for DW design based on Dimensional Fact Model.

Song, Khare, and Dai [7] developed the SAMSTAR method that is a semi-automated approach to generating star schema from operational source ERD. Although, the authors mentioned that the SAMSTAR method was both data-driven and goal-driven, this method is primarily data-driven because it derives star schema based on the structures and semantic of operational sources. Zepeda, Celma, and Zatarain [8] proposed a conceptual design approach consisting of two stages. The first stage is to generate multidimensional data structures from UML-based enterprise schema. The second stage is to use user requirements to select relevant schema. Moreover, the algorithm for automatic conceptual schema development and evaluation based on Multidimensional Entity Relationship Model (M/ER) was invented by Phipps and Davis [9]. Similarly, Moody and Kortink [10] proposed a methodology for designing DW schema based on enterprise models.

On the other hand, a goal-driven approach gives more relevance to user requirements in designing DW. Prakash and Gosain [11] present a requirement-driven data warehouse development based on the goal-decision-information model. In addition, Giorgini, Rizzi, and Garzetti [12] propose a goal-oriented requirement analysis for DW design in which the organizational goals are made explicit and decomposed into sub-goals and then the relationships among sub-goals and actors are identified and analyzed. Their method starts with identification of corporate goals (i.e., user requirements) and actors involved. The actor can be either a responsible persons or resources that are needed to accomplish the goal.

We focus on the conceptual design phase to provide a blueprint for lower level logical design that is consistent with the KPI definitions from which we start. Tryfona, Busborg, and Christiansen [13] developed the starER model for conceptual design of Data Warehouses and argued that DW design should be exposed to higher level so that it becomes more understandable, and easier to identify conceptually what are ingredients are actually needed in the DW. In addition, it is advisable not to use computer metaphors such as 'table' or 'field'

Jones and Song [14] developed Dimensional Design Pattern (DDP) that assists designers to effectively determine commonly used DW dimensions. In this sense, the DDP framework consist of six classes of dimension domain, from which DW designer can choose specific dimension and attributes during the mapping process.

Moreover, an important issue in designing DW schema is additivity of facts. A fact is additive relative to a dimension if it is summarizable along that dimension. The importance of summarizability is discussed by Shoshani [15]. Horner, Song, and Chen [16] present a taxonomy of summary constraints that can be used for this purpose.

The other issue in designing DW schema is the choice between the various types of multidimensional data models, among which star schema and snowflake schema are most common in data warehouses. However, the most data warehouses use star schema for two important reasons. First, it is the most efficient design because less joint operations are required due to denormalized tables. Second, the star schema is supported by most query optimizers for creating an access plan that use efficient star join operations [17].

A study of data warehouse in connection with KPIs can be found in the triple-driven data modeling methodology presented by Guo et al [18]. This methodology consists of four major stages: (1) goal driven stage, (2) data driven stage, (3) user driven stage, and (4) combination stage. During the first stage, business goals and KPIs are identified according to business subject area. The second stage is to obtain a data schema that supports the KPIs from the operational data sources. The third stage is to interview users in order to identify important business questions. The fourth stage is to check if the business KPIs can be calculated and questions can be answered by the obtained data schema. As indicated by its second stage, this methodology is primarily data-driven because the operational sources impose total constraints on the computation of KPIs. Moreover, the first stage is where KPIs have to be identified and the attributes needed to support these KPIs have to be determined. However, this methodology does not specify how to determine those required attributes as part of the DW data models. In other words, the practical steps to analyze the KPI structural definition are not provided. In addition, the generation of star schema is based on the data-driven method that was developed by Moody and Kortink [10].

Vaisman and Zimány [21] propose a classification of KPIs along several dimensions. First KPIs are classified wrt. to the time span of observations (past, present, future). Second, they distinguish KPIs on inputs needed for a business results from KPIs about the business result and performance. Further, there are operational vs. strategic KPIs and qualitative (obtained by surveys etc.) vs. quantitative. They multidimensional expressions (MDX) to relate a KPI value to a KPI goal (expressed as thresholds or intervals).

In the sequel, we develop an informal guideline on how to create a data warehouse schema out of patterns found in business process models. The multi-dimensional character of the KPIs is excerpted from the products serving as inputs and outputs of the processes, the resources used in the processes, and time and location information. We also shall review the role of plans and schedules (compare to targets in BIM) in formulating KPIs.

3 Data Warehouses for Structuring Observations

A data warehouse manages multi-dimensional facts, where each fact constitutes an observation about the domain of interest, e.g. an enterprise. The structure of an observation is a tuple

$$(d_1, d_2, \ldots, d_k, m)$$

where d_i are dimension entities represented by their identifier and m is a measurement value, typically a number. The measurement value attribute is functionally dependent from the combination of dimension entities. As an example assume that we have the dimensions car, location, and time and the measurement attribute ‘speed’ for representing car speed observations. Then, the observation facts would look like

```
('Marys car','Skövde',2013-09-28T10:31:19,385)

('Johns car','Barcelona',2013-03-12T21:07:47,145)
```

As functional expressions, these observations can be represented as equations

speed(‘Marys car’, ‘Skövde’,2013-09-28T10:31:19)=385 *speed('Johns car','Barcelona',2013-03-12T21:07:47)=145*

We learn from this example that the dimensions of the observation determine the circumstances under which the speed observation was made. The car parameter is representing an entity participating in the observation. Location and time are dimension entities that frequently occur in observations. Other than the car, they are not entities/objects of the real world but we can reify them to be entities. This reification is common in data warehouses by creating dimension tables where temporal and special dimension values get surrogate identifiers. The goal of this paper is to derive the dimensions for a simple KPI from a high-level specification for this KPI.

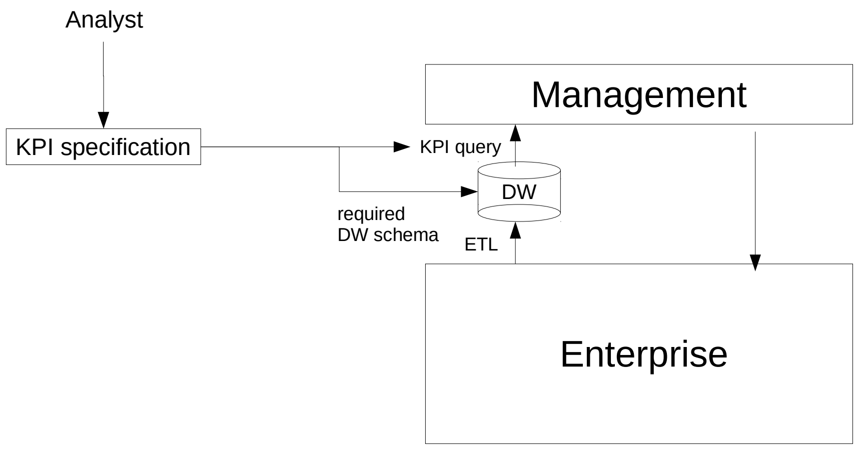


Fig. 1. Workflow of creating DW schemas from KPI definitions

The general steps for realizing the KPI are

1. Specify the KPI including its measurement context. The measurement context is defined by a combination of entities (customers, products, time, location, etc) that were present when the observation was made.
2. Create the supporting data warehouse schema. We limit ourselves in this paper mostly on the fact table.
3. Code the queries computing the KPI on top of the created schema.

Natural language KPI definitions found in practice are usually rather ambiguous by nature. Take for example the average speed of cars as a KPI for the traffic process. What is the context of the underlying observations? It can be the time of the measurement, the location, and the car involved in the measurement. However, it could also include the car driver. Some of the relevant context may be difficult to determine such as the car driver. This can limit the utility of the KPI for decision making or for understanding the process underlying the observations.

3.1 The process nature of observations

An observation is a statement made by agent (the observation) about an object in the reality, possibly involving other objects. Lenz and Shoshani [19] differentiate flow and stock observations. A *flow* observation is about recording a state change of the object recorded wrt. some time interval, a *stock* observation is a record about the object's state. As a third category, they list *value-per-unit* observations, such as the price of a product. Assume we would only record stock observations. If there are no changes, then the observations of an object would also not change. This is like listing the specific weights of elementary substances. If there are changes, then the states of objects vary over time and shall yield different observations. The reasons for changes are *processes* taking place in the reality. These processes can be natural like the radioactive decay or they are man-made, such as production processes. Consider the example of an oil refinery that stores oil in large tanks. Each tank has a fill level. There are two processes that can change the fill level: adding oil to the tank and removing oil. These processes are embedded in more complex processes taking place at the oil refinery. Flow observations about the oil tank record how much the state of an object has changed between two points of time. For example, how many liters of oil have been added and how many have been removed in the last month. If the state is known at the start of the time period, then the state at the end of the time period can be calculated by applying the additions and subtractions of the flow observations. The third observation type, value-per-unit break down stock or flow observations to small units, such as the oil price per liter. Assume that the oil refinery buys quantities of oil on the market at different prices and then stores the oil in the tank. Then each liter of oil stored in a tank virtually carries its unit price with it. The total value of the oil in the tank is then the sum of all oil liter unit prices of oil liters stored in the tank.

The lesson learned from this argumentation is that state changes require the presence of processes. If the processes are natural, then human influence on them is limited. For example, the water cycle on earth is driven by the sun and leads to varying levels of water in the river systems. Still, it makes perfect sense to record observations about the water cycle in order to predict the water levels of certain rivers at certain locations, e.g., to prepare for flooding. An organization with man-made processes has an interest in managing the processes to achieve its goals, e.g. to increase the profit or to raise customer satisfaction. The management includes changing the parameters of process steps (e.g. their scheduling), adapting the resources (e.g. the machines used in production steps), changing the inputs of process steps (e.g. replacing a part by another part), or changing the process itself (e.g., reordering the process steps or removing unnecessary activities).

A single observation occurs in a context, which is characterized by the participating entities. Time and space are regarded here as entities as well.. The presence of time and space indicate that such observations are practically always related to an underlying process,. The process is the reason why the entities are combined and lead to observations. As an example, consider the usage process of a customer c1 for a product p1 at a time t1. The observation for the combination of these three entities could be a defect of product p1. This is an atomic observation. The measurement attribute 'defect' is either 0 or 1.

Now, the customer c1 belongs to the set of customers, e.g. the set of customers in Brazil. The product p1 belongs to the set of all products of a given type, say ACME Phone-One. Then, we can state for example

Defects('Brazil','ACME Phone-One',2014) = 371

The example highlights that it is crucial to identify the context of an observation as a combination of participating entities. Combined with another simple KPI on the number of products of a given type sold in a country in a given year, one can define a derived KPI on the defect density:

DefectDensity('Brazil','ACME Phone-One',2014) =
Defects('Brazil','ACME Phone-One',2014) /
Sales('Brazil','ACME Phone-One',2014)

A *derived* KPI is simply a KPI that is defined in terms of other KPIs. A *simple* KPI is calculated from a set of atomic observations. Note that the arguments of the two KPIs 'Defects' and 'Sales' are the same, i.e. the context of the two underlying observation types is the same.

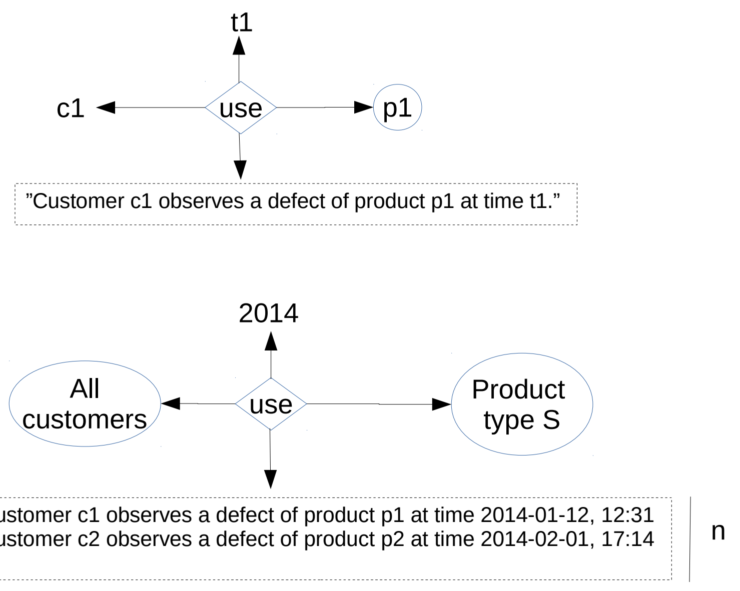
Figure 2 visualizes the step from multi-dimensional atomic observations (upper part) to multi-dimensional aggregated observations (lower part). The aggregated observations are sets of atomic observations about the usage activities of cutomers with products where the dimension entities of the atomic observations are member of the dimension values of the aggregated observation. The lower shows rolled-up dimension entities (all for customer, 2014 for time, and product group S). These dimension entities match a set of observations, which can be aggregated e.g. by counting the number of observations. Any set of atomic observations can define a multitude of KPIs by combining different dimension values. For example, the KPI

Defects('Brazil','ACME Phone-One',2014-01)

aggregates all defect observation in Brazil for the product group 'ACME Phone-One' in January 2014. We call all such KPIs simple KPIs even though equalities such as

Defects('All','ACME Phone-One',2014) =
Defects('Brazil','ACME Phone-One',2014-01) +
Defects('Brazil','ACME Phone-One',2014-02) +
...
Defects('Brazil','ACME Phone-One',2014-12)

hold true. The equality holds true due to the definition of the KPIs on the same set of atomic observation and the roll-up relations of the dimension entities.



"Customer c1 observes a defect of product p1 at time t1."

Customer c1 observes a defect of product p1 at time 2014-01-12, 12:31
Customer c2 observes a defect of product p2 at time 2014-02-01, 17:14
...

Fig. 2. Context of atomic and aggregated observations

We conclude that observations about processes are the basis to define KPIs and that the context of observations can be represented as a combination of entities such as products, resources, time, and location. These entities are the same entities that form the dimensions in a data warehouse. This view is not the only view on KPIs but it is the one used subsequently to create guidelines on how to derive data warehouse schemas and queries from KPI definitions.

4 From KPI definitions to Data Warehouse Schemas

As motivated before, any KPI is based on observations about underlying processes. We focus on simple KPIs here, i.e. KPIs that are based on a single type of observation denoted as

$O(e_1,e_2,\ldots,m)$

where e_i are the entities participating in the observation and m is the value of the observation, usually a number. Hence, an observation is a synonym to a fact in a data warehouse where all dimension values are taken from the lowest rollup level. We also use the functional representation

$O(e_1,e_2,\ldots)=m$

when appropriate. Since the majority of KPIs are process-oriented, we use process models to relate them to elements of process models. Specifically, we use the Petri net notation [20] extended by resource and input/output elements to represent process model patterns. Petri nets are the formal basis for process modeling languages such as BPMN. They provide a clear token passage semantics of the process execution, which is necessary to define performance indicators such as cycle time.

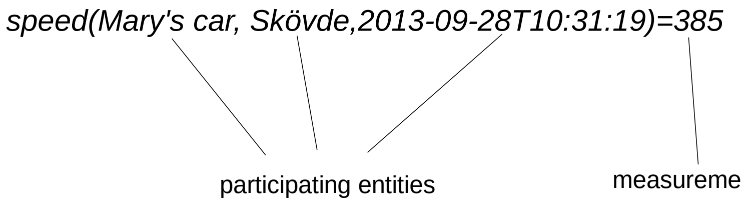


Fig. 3. Participating entities and measurements

The notion of participating entity is central to the subsequent derivation of a data warehouse schema. We treat here time and location as entities just like any other physical or immaterial object.

Motivating Example: Derive DW schema for the KPI "average speed of cars".

Is it a simple or derived KPI?

This is a simple KPI with an atomic underlying observation type.

What is the structure of the observation type?

We identify the participating entities car (given by its identification), the location of the speed measurement, and the time when the measurement was taken. The measure is a number with unit km/h. Hence the type of the observation is

speed(CAR,LOC,TIM,SPEEDM)

What is the schema of the fact table of a data warehouse supporting the KPI?

The participating entities become dimensions, e.g.

```
CREATE TABLE SPEEDS (
    CARID INT,
    LOCID INT,
    TIMID INT,
    SPEEDM FLOAT,
    PRIMARY KEY (CARID,LOCID,TIMID),
    FOREIGN KEY (CARID) REFERENCES CAR (CARID),
    FOREIGN KEY (LOCID) REFERENCES LOCATION (LOCID),
    FOREIGN KEY (TIMID) REFERENCES TIMETBL (TIMID));
```

We omit the definitions of the dimension tables since the roll-up hierarchies are not mentioned in the KPI definitions.

The query for computing the KPI is then a straightforward aggregate query on the fact table.

Pattern 1: Derive DW schema for the KPI "average processing time for task1 in a process".

Figure 4 shows a Petri-net-style process fragment to analyze the KPI. Place p1 represents that some case is currently being processed by task 1. The places are waiting positions for the cases that flow through the process. The two transitions 'begin' and 'end' start or terminate the task, respectively. A case is a data object representing an external or internal event to which an organization has to react. It carries an identifier (the case id) and possibly further attributes that describe the case. The attributes are used to decide how to route a case thru a process [20]. The *inner place* p_i is uniquely defined for each task in a process model.

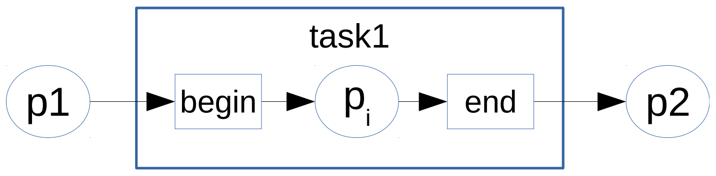


Fig. 4. Process fragment for understanding processing time

Is it a simple or derived KPI?

This is a derived KPI based on the arrival and departure times of cases at the inner place p_i of a task.

What is the structure of the observation types?

There are two observation types:

arrivaltime(CASE,PLACE,ARRTIME)
departuretime(CASE,PLACE,DEPTIME)

Here the time is not a participating entity but a measurement. There are two dimensions involved in the observation: the case dimension and the place dimension. The place dimension can be rolled up to the task to which it is connect and then to the process to which the task belongs to.

What is the schema of the fact tables of a data warehouse supporting the KPI?

```
CREATE TABLE ARRIVALTIME (
    CASEID INT,
    PLACEID INT,
    ARRTIME DOUBLE,
    PRIMARY KEY (CASEID,PLACEID));
CREATE TABLE DEPARTURETIME (
    CASEID INT,
    PLACEID INT,
    DEPTIME DOUBLE,
    PRIMARY KEY (CASEID,PLACEID));
```

The query to compute the simple KPI *arrivaltime(o,p)* is then

```
SELECT ARRTIME FROM ARRIVALTIME WHERE
    CASEID = o AND
    PLACEID = p;
```

The two fact tables can also be merged into a single one with two measurement attributes. Foreign key references and the definitions of the dimension tables are omitted. The query to compute the KPI aggregates the average of the difference of the departure time of the inner place p1 of a given task. We leave the query coding to the reader. The Petri net view on the process allows to determine what events need to be recorded by a process execution system. For pattern 1, the system has to record the time when a case is picked up by a task (arrival time at p_i) and when the task finishes a case (departure time at p_i).

Pattern 2: Average cycle time of a case in a process

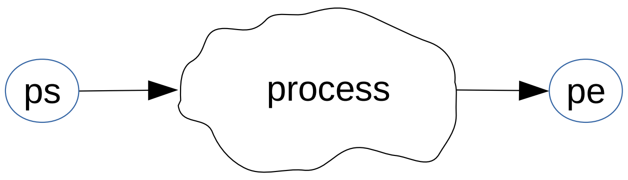


Fig. 5. Cycle time of process

The cycle time is the accumulated time of a case in a process, from start to en**d.** Assume that ps is the unique start place of the process and pe is its unique end place, then the cycle time of a case c is a derived KPI based on the arrival time:

cycletime(c) = arrivaltime(c,pe) – arrivaltime(c,ps)

We thus can reuse the definition of *arrivaltime* of the previous example. We assume that the process has a unique start ps and a unique end pe. The SQL query to compute the average cycle time over all cases is left to the reader. It multiple processes are analyzed by the same data warehouse, then one can add a process dimension to the fact table for cycletime. Processes can be rolled up to process groups at the discretion of the data warehouse designer.

The cycletime is calculated here from the simple KPI 'arrivaltime'. If the complete process definition is known, then one can establish an equality of the cycletime with the sum of all waiting times plus all processing times for a case flowing through the process.

Pattern 3: Average waiting time on a place p

This is another derived KPI that can be defined in terms of arrival and departure time:

waittime(c,p) = arrivaltime(c,p) – departuretime(c,p)

The waittime can be aggregated to the total waiting time of a case in a process. If a process proc has no cycles, then it is defined by the formula

procwaittime(proc) =
sum{arrivaltime(c,p)-departuretime(c,p)| c in CASE, p.process=proc}

If the process has cycles, then cases can visit the same place multiple times. Then, our original definitions for arrival and departure time cannot be used anymore. To solve the problem, we add an additional participating entity 'visit' that contains the identifier of the visit of a case on a place:

```
CREATE TABLE ARRIVALTIME (
    VISITID INT AUTOINCREMENT,
    CASEID INT,
    PLACEID INT,
    ARRTIME DOUBLE,
    PRIMARY KEY (VISITID,CASEID,PLACEID));
```

The fact table for departure time is updated accordingly. Then, the process waiting time can be defined as

procwaittime(proc) =
sum{arrivaltime(v,c,p)-departuretime(v,c,p)| c in CASE,
p.process=proc,v in INT}

We leave the SQL coding for calculating the KPI to the reader.

Pattern 4: Average person hours spent on a task for a given case

Person hours are an example of a resource-based metric. Resources are allocated to tasks. They are reserved during the execution of the task and typically released before the end of the task. We can distinguish consumable resources such as energy and non-consumable resources such as machines or employees. The latter can be converted to consumable resources by considering resource hours instead of the resource itself.

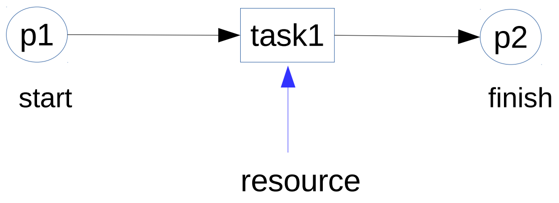


Fig. 6. Resources linked to tasks

Figure 6 links a resource to a task in a SADT-like style as also used by Fenton and Pfleeger for software processes [22]. In an SADT (structured analysis and design technique) diagram, a task has inputs, outputs, resources used for the task and control information (e.g. a time schedule). The resource consumption can be observed by identifying the current case, the task to be performed, the identifier of the resource. The measurement is the consumption of the resource, e.g. person hours. Hence the observation type is

personhours(CASE,TASK,RESOURCE,RHOURS)

The following fact table implements the observation type:

```
CREATE TABLE PERSONHOURS (
    CASEID INT,
    TASKID INT,
    RESOURCEID INT,
    RHOURS DOUBLE,
    PRIMARY KEY (CASEID,TASKID,RESOURCEID));
```

The query to compute the resource consumption per task and resource is then as follows:

```
SELECT  TASKID,RESOURCEID,AVG(RHOURS)  FROM  PERSONHOURS
GROUP BY (TASKID,RSOURCEID);
```

Pattern 5: Percentage of the truck shipment time where the truck cooling device is active

This KPI is derived from the process time of the truck shipment and the aggregated cooling times of the cooling device resource. The first KPI is discussed in example 2. Hence, we only need to handle the use of the cooling device.

cooling(ENGAGE,CASE,TASK,CTIME)

The cooling device can be engaged multiple times during a shipment. The observation has as participating entities the engagement id, the case, the task (ship) and as measurement the time of the engagement.

```
CREATE TABLE COOLING (
    ENGAGEID INT AUTOINCREMENT,
    CASEID INT,
    TASKID INT,
    CTIME DOUBLE,
    PRIMARY KEY (ENGAGEID,CASEID,TASKID));
```

The cooling time aggregated over all engagements for a given task and case is then:

```
  SELECT TASKID,CASEID,SUM(CTIME) FROM COOLING
  GROUP BY (TASKID,CASEID);
```

In a similar way one can implement KPIs on power consumption of a machine resource used to perform a given task.

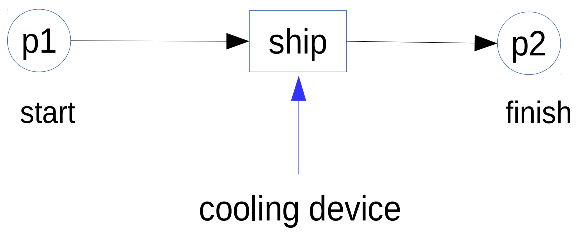


Fig. 7. Use of the cooling device

Pattern 6: Material used to create a product

Physical processes create output products using input products. The input products are not resources but they become part of the output. To model KPIs for such processes, we need to explicitly represent inputs and outputs of tasks. Figure 8 shows the inputs and outputs of a task. Note that their relationships to the task are different from the control flow between place 1, the task and place 2. A task can have multiple products as inputs and also produce multiple outputs. Each participating product can have a quantity (measured in physical units or as count). For example, to produce an engine for a car, one needs a certain quantity of aluminum poured in a form. The observation type for input products is characterized by the participating entities case, the task, and the input product. The measurement is the quantity of the product used for the task on the given case.

input(CASE,TASK,PRODUCT,QUANTITY)

The outputs can be characterized accordingly

output(CASE,TASK,PRODUCT,QUANTITY)

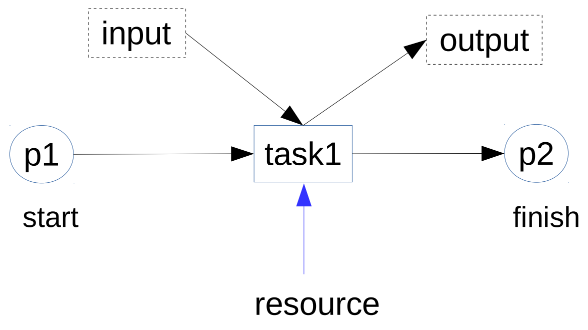


Fig. 8. Inputs and outputs of a task

Let us assume that 23.23 kg of aluminum are used to create a certain engine 123. The observation facts would then be:

input(engine123,pour,aluminum,23.23)
output(engine123,pour,engine,1)

The next engine could require slightly less aluminum:

input(engine124,pour,aluminum,23.19)
output(engine124,pour,engine,1)

```
CREATE TABLE INPUT (
    CASEID INT,
    TASKID INT,
    PRODUCTID INT,
    QUANTITY DOUBLE,
    PRIMARY KEY (CASEID,TASKID,PRODUCTID));
```

The average consumption of aluminum per engine is then a simple aggregate query over the input table:

```
SELECT TASKID,PRODUCTID,AVG(QUANTITY) FROM INPUT
GROUP BY (TASKID,PRODUCTID);
```

Pattern 7: As-Is vs To-Be comparisons

The last pattern discussed in this chapters are deviations from the plan and KPIs that relate planned performance to the actual performance. A typical example is a budget for a project. This is a planned measure. The actual execution of the project may less, all, or more than the planned budget. Another example is the deadline for a certain task. The previous patterns already discussed the actual performance of a process, including resource consumption. Figure 9 adds planned performance to our extended process model. We can regard the planned performance as a simple observation type, which has no participating case.

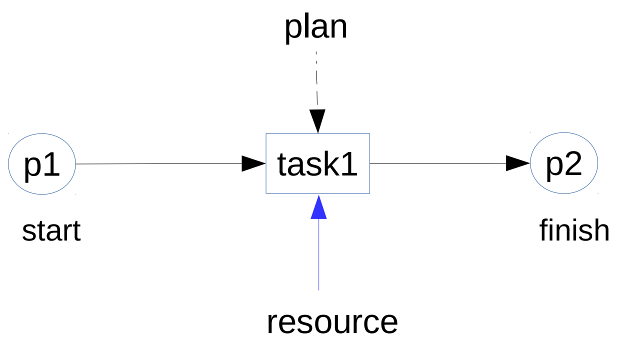


Fig. 9. Planned performance

As an example, consider the planned processing time of task 1. It can be represented in an observation fact

plannedproctime(TASK,PTIME)

This observation fact can be used like any other to form aggregate KPIs like the average planned processing time over all tasks. The more interesting use is to form derived KPIs with KPIs on the actual performance. Similar planned performance KPIs can be defined for resource consumption, and input and outputs.

5 Conclusions

This paper discussed how to map a KPI definition to a data warehouse schema and the query calculating the KPI. Rather than developing a method to automatically generate the schema and queries, we elaborated on patterns for process performance KPIs. The patterns included processing time, waiting time, resource consumption, material use, and the comparison of planned versus actual performance. An extended process model that includes places, tasks, resources, inputs/outputs, and plans was incorporated to derive the observation types underlying the KPIs. Simple KPIs have a single observation type associated to them. Derived KPIs are computed as expressions over simple KPIs.

The patterns can be used to support the top-down design of a data warehouse from a set of process-related KPIs that shall be computed by it. The starting points are the natural language KPI definition and a process model fragment that visualizes the context in which the observations belonging to the KPI are collected. The notion of Petri-net places allowed for a straight-forward definition of time-based KPIs by just using arrival and departure times of cases at and from places. The pattern on resource consumption allows dealing with a whole group of KPIs such as person hours spent on a task.

The input/output pattern allows to measure physical material flow. These patterns can also be combined with the other patterns, e.g. to measure how many person hours are needed to produce a certain number of products. Finally, planned performance is realized by a simplified observation type that has no case identifiers.

We argue that practically all KPIs are process-related because any change of a state requires some activity leading to the state change. Some KPIs are about 'stock' observations (c.f. Lenz and Shoshani [19]), e.g. observing the number of cars on a certain street segment. The observation is related to the ongoing travel processes of the car drivers, which are not made explicit in an information system about the traffic status. The observation times are independent of the underlying travel processes.: two consecutive observations could be about the very same state. If the process is not explicit, then it cannot be controlled so easily. 'Flow' observations are directly linked to a process task, since they make an explicit statement on a state change. In the traffic example, each time that a car enters or leaves the street segment, an observation would be recorded. This type of observation allows to control the traffic, e.g. by using traffic lights for the street segment that is set to red when too many cars are in the segment. In this paper, we thus focussed on flow observations.

Future work is needed to understand how to define a KPI in a formal language such that a supporting data warehouse schema can be automatically generated from the KPI definition. Another open question is whether the discussed 7 patterns cover a considerable portion of KPIs actually used in practice. The KPI Library (http://kpilibrary.com) contains more than a thousand KPIs in high level natural language that can be used to answer this question. We did not discuss how dimension tables can be created and populated. Most rollup hierarchies are domain-specific with the exception of time.

Finally, it would be interesting to investigate rules for the correct definitions of derived KPIs in terms of constraints on the use of parameters for the participating entities of the observation facts. For example, it does not make (much) sense to compare the arrival times and departure times of places belonging to different processes.

KPIs can also be regarded as statistical variables, possibly depending on each other. The long-term collection of KPIs can be used to calculate their correlation and thus to form a theory on estimating dependent KPIs from independent ones. This paper was meant to encourage the systematic collection of many process KPIs such that theories for predicting them can be developed´and validated using methods from SPC [25].

References

1. Wetzstein, B., Leitner, P., Rosenberg, F., Brandic, I., Dustdar, S., Leymann, F.: Monitoring and Analyzing Influential Factors of Business Process Performance. In: Proceedings 2009 IEEE International Enterprise Distributed Object Computing Conference, Auckland, NZ, Sep. 1-4, 2009, 141-150 (2009)
2. Deming, W.E.: On Probability as a basis for action. The American Statistician 29(4), 146-152, (1975)
3. CMMI Guidebook Acquirer Team: Understanding and Leveraging a Supplier's CMMI Efforts: A Guidebook for Acquirers, CMU/SEI-2007-TR-004. Software Engineering Institute. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=8315, March 2007.
4. Horkoff, J., Barone, D., Jiang, L., Yu, E.S.K., Amyot, D., Borgida, A., Mylopoulos, J.: Strategic Business Modeling: Representation and Reasoning. SoSym 13(3), 1015-1041 (2014)
5. Golfarelli, M., Maio, D., Rizzi, S.: Conceptual Design of Data Warehouses from E/R Schema. In: Proceedings of the 31th Annual Hawaii International Conference on System Sciences-Volume 7, 334-243 (1998)
6. Golfarelli, Rizzi, S.: Data Warehouse Design – Modern Principles and Methodologies. McGrawHill (2009)
7. Song, I.-Y., Khare, R., Dai, B.: SAMSTAR: a semi-automated lexical method for generating star schemas from an entity-relationship diagram. Proceedings ACM 10th Workshop on Data Warehousing and OLAP, Lisbon, Portugal, Nov. 9, 2007, 9-16 (2007)
8. Zepeda, L., Celma, M., Zatarain, R.: A methodological framework for conceptual data warehouse design. Proceedings of the 43nd Annual Southeast Regional Conference, 2005, Kennesaw, Georgia, Alabama, USA, March 18-20, 2005, Volume 1, 256-259 (2005)

9. Phipps, D., Davis, K. C.: Automating Data Warehouse Conceptual Schema Design and Evaluation. In: Proceedings of the 4th International Workshop on Design and Management of Data Warehouses (DMDW), Toronto, Canada, CEUR-WS.org/Vol-58, 23-32 (2002)
10. Moody, D., Kortink, M.A.R.: From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. Proceedings of the Intl. Workshop on DMDW 2000, Stockholm, Sweden, June 5-6, 2000, CEUR-WS.org/Vol-28 (2000)
11. Prakash, N., Gosain, A.: Requirements Driven Data Warehouse Development. Short Paper Proceedings CAiSE 2003, Klagenfurt, Austria, June 16-20, 2003, CEUR-WS.org/Vol-74 (2003)
12. Giorgini, P., Rizzi, S., Garzetti, M.: Goal-oriented requirement analysis for data warehouse design. Proceedings ACM 8th International Workshop on Data Warehousing and OLAP, Bremen, Germany, November 4-5, 2005, 47-56 (2005)
13. Tryfona,N. , Busborg, F. , Christiansen, J.G.B.: StarER: a conceptual model for data warehouse design. Proceedings of the 2nd ACM International Workshop on Data warehousing and OLAP, Kansas City, Missouri, USA, November 2-6, 1999, 3-8 (1999)
14. Jones, M. E., Song, I.-Y.: Dimensional Modeling: Identification, Classification, and Applying Patterns. Proceedings 8th ACM International Workshop on Data Warehousing and OLAP (DOLAP), Bremen, Germany, 29-38 (2005)
15. Shoshani, A.: OLAP and statistical databases: Similarities and differences. Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. ACM Press, New York, NY, USA, 185-196 (1997)
16. Horner, J. , Song, I.Y. , Chen,P.P.: An analysis of additivity in OLAP systems. Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, November 12-13, 2004, Washington, DC, USA (2004)
17. Martyn,T: Reconsidering Multi-Dimensional schemas, ACM SIGMOD Record 33(1), 83-88 (2004)
18. Guo,Y., Tang, S., Tong, Y., Yang, D: Triple-driven data modeling methodology in data warehousing: a case study. Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP, November 10, 2006, Arlington, Virginia, USA (2006)
19. Lenz, H.J., Shoshani, A.: Summarizability in OLAP and statistical databases. Proceedings SSDBM 1997, http://dx.doi.org/10.1109/SSDM.1997.621175.
20. van der Aalst, W.M.P., van Hee, K.M.: Workflow Management – Models, Methods, and Systems. MIT Press (2002).
21. Vaisman, A.., Zimány, E..: Data Warehouse Systems – Design and Implementation. Springer (2014).
22. Fenton, N.E., Pfleeger, S.L.: Software Metrics - A Practical and Rigorous Approach. Thomson, 1996.
23 Software Engineering Institute: CMMI for Development - Version 1.3. Technical Report CMU/SEO-2010-TR-033, Carnegie-Mellon University, November 2010.
24. Oivo, M., Basili, V.R.: Representing Software Engineering Models - The TAME Goal Oriented Approach. IEEE Trans. Software Eng., 18, 10, 886-898, http://dx.doi.org/10.1109/32.163605.
25. Oakland, J.S.: Statistical Process Control. Sixth Edition. Routledge, 2008.