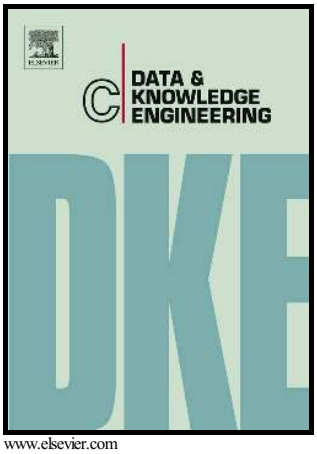

Specification and Derivation of Key Performance Indicators for Business Analytics: A Semantic Approach

Alejandro Maté, Juan Trujillo, John Mylopoulos

PII: S0169-023X(16)30377-9
DOI: http://dx.doi.org/10.1016/j.datak.2016.12.004
Reference: DATAK1578

To appear in: *Data & Knowledge Engineering*

Received date: 20 November 2015
Revised date: 5 December 2016
Accepted date: 15 December 2016

Cite this article as: Alejandro Maté, Juan Trujillo and John Mylopoulos Specification and Derivation of Key Performance Indicators for Busines Analytics: A Semantic Approach, *Data & Knowledge Engineering* http://dx.doi.org/10.1016/j.datak.2016.12.004

Specification and Derivation of Key Performance Indicators for Business Analytics: A Semantic Approach

Alejandro Maté[a,*], Juan Trujillo[a], John Mylopoulos[b]

[a] *Lucentia Research Group, Department of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n - 03690 San Vicente del Raspeig - Alicante, Spain*
[b] *Department of Computer Science, University of Trento, Via Sommarive 9 - 38123 Povo - Trento, Italy*

Abstract

Key Performance Indicators (KPI) measure the performance of an enterprise relative to its objectives thereby enabling corrective action where there are deviations. In current practice, KPIs are manually integrated within dashboards and scorecards used by decision makers. This practice entails various shortcomings. First, KPIs are not related to their business objectives and strategy. Consequently, decision makers often obtain a scattered view of the business status and business concerns. Second, while KPIs are defined by decision makers, their implementation is performed by IT specialists. This often results in discrepancies that are difficult to identify. In this paper, we propose an approach that provides decision makers with an integrated view of strategic business objectives and conceptual data warehouse KPIs. The main benefit of our proposal is that it links strategic business models to the data for monitoring and assessing them. In our proposal, KPIs are defined using a modeling language where decision makers specify KPIs using business terminology, but can also perform quick modifications and even navigate data while maintaining a strategic view. This enables monitoring and what-if analysis, thereby helping analysts to compare expectations with reported results.

Keywords: Business Intelligence, Conceptual Data Warehouse Models, Key Performance Indicators, Strategic Models, Business Analytics

1. Introduction

Key Performance Indicators (KPI) are used by enterprises to monitor the performance of their processes and business strategies [21, 33] relative to their objectives. KPIs are traditionally defined with respect to a business strategy or business objective using a Balanced Scorecard [18], to indicate what is to be monitored in different areas of the enterprise thereby providing a global overview of the enterprise's status. To monitor KPIs, enterprises rely on dashboards [10, 33] presenting one or more KPIs together with contextual information in order to help decision makers identify deviations and their root causes.

However, this practice presents several drawbacks. First, it provides only partial information to decision makers, as KPIs are created and analyzed in isolation without taking into account inter-relationships and influences between them. For example, how are we achieving our objective "Reduce costs"? How does this affect our other objective "Increase revenue"? Essentially, decision makers query for a strategic problem and obtain data, such as current cost totals, missing the rest of the strategic context as an answer. Then, they are required to interpret and link these raw data back to their business strategy and how it affects other business objectives. For example, our KPI "Manufacturing Cost" has increased as a result of an increase in the price of basic materials. What

*Corresponding author. Tel: +34 96 5909581 ext. 2737; fax: +34 96 5909326

other KPIs and objectives are affected by this change and how? Is there a way to compensate for this increase? Do we have margins to increase our prices? Or should we turn to finding cheaper materials? Second, the decision maker lacks an adequate view to verify that the business strategy and the KPIs are consistent with each other. For example, KPI "Customer satisfaction" may be indicating that we are meeting our goal, but we are not achieving another target goal "Sales". Is our KPI ill-defined? Are there other KPIs missing?Is the real influence between "Customer satisfaction" and "Sales" different from what was estimated? As described in [11] these kinds of discrepancies can have dire consequences for an enterprise, as effort and resources are wasted. Third, even if a complex dashboard is created capturing all aspects of business strategy, there is a semantic gap between the decision makers' knowledge and the decision support system. While decision makers express themselves in terms of business concepts, KPIs are expressed in technical terms, typically MDX (MultiDimensional eXpressions [27]) or SQL, which are hardly understandable by non IT staff. Therefore, this (i) limits the ability of decision makers to share and reuse their knowledge across the enterprise and (ii) requires close collaboration between IT and business staff, despite different terminologies, skills, and no mechanisms to help communication, thereby increasing the likelihood that miscommunication will happen and will go unnoticed.

In order to tackle some of these problems, researchers have proposed several solutions, including strategic and performance indicator modeling [1, 17, 36, 35], and preserving traceability from the decision models to the underlying data warehouse [22, 36]. However, these proposals do not provide a holistic framework that allows decision makers to not only define but also query KPIs using their own terms and analyze the results from a strategic point of view. Therefore, in this paper we propose a semi-automatic process that (i) allows decision makers to define KPIs with the semantics of their own business terminology, (ii) derives a data warehouse query (specifically, MDX) from the definition of each KPI, (iii) executes the MDX query against a target OLAP server and loads the value of each KPI into a strategic model, and (iv) allows the user to navigate across the data using a business strategy perspective. An overview of the steps involved in the process can be seen in Figure 1 and are further elaborated in the following Sections.

Our proposal presents several advantages. First, it provides a comprehensive view for the decision maker of the status of indicators (current qualitative satisfaction level and quantitative values), including their relationships, associated goals and known internal and external factors affecting them, thus helping in the identification of failures, while guiding further analysis. Second, it allows the decision maker to query and navigate through the data. This is achieved thanks to the definition of global constraints that dynamically alter the calculus of all the indicators included in the business strategy and updates it, thus enabling the decision maker to interpret the results in their own

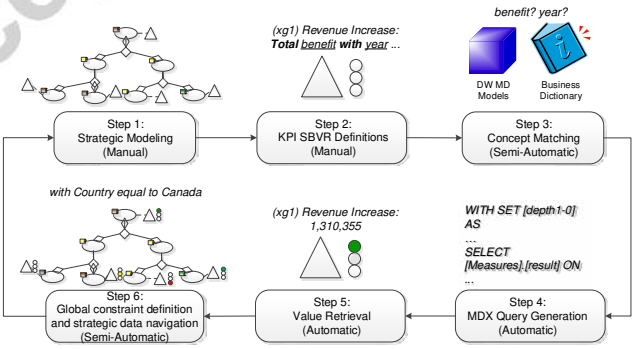

Figure 1: Overview of the steps included in our approach

terms and dynamically compare different subsets of their enterprise data while maintaining the strategic perspective. Third, it helps to shorten the development cycle by (i) providing a more business-oriented formal language for the definition of KPIs and (ii) semi-automatically deriving the corresponding MDX queries for the KPI, thus allowing for rapid prototyping of the KPIs and providing a basis for optimized implementations.

It is noteworthy that our proposal focuses on exploiting the underlying data warehouse, and thus can help identifying defects in its implementation such as missing data or faulty multidimensional structures. However, while our proposal can aid in eliciting these new requirements for the decision support system, it is not within its scope to cover the iterative development of the data warehouse, as there are already multiple approaches that cover this aspect [25, 26, 22, 13].

This paper is an extended and revised version of [23], where a first version of our language for KPI specification was introduced. Compared to that publication, this paper has been extensively rewritten and (i) includes a revised, more expressive and compact version of the language, enabling the calculus of complex values and their use as part of conditions, (ii) introduces in more detail the role and structure of the Business Dictionary expressed using the SBVR (Semantic of Business Vocabulary and Rules) standard [30], for storing and translating business terms, (iii) describes the translation of the intermediate language into executable MDX queries, (iv) offers a comprehensive account of a case study and the specification of all KPIs and values obtained, and (v) provides a detailed account of the architecture that supports our approach, including implementation details.

The remainder of this paper is structured as follows. Section 2 presents the basic concepts of Business Intelligence Model (BIM) for representing business strategy and introduces our running example for the rest of the paper. Section 3 describes how KPIs can be defined in SBVR terms. Section 4 presents how OLAP actions generated from KPI definitions can be transformed into MDX queries. Section 5 describes how the proposed model can be used as a strategic dashboard that can also facilitate navigation through the data. Section 6 presents the architecture of the system and its components including details of our implementation to support the whole process. Section 8 presents related work in this area, while Section 9 summarizes conclusions and sketches future work.

2. Basic Concepts and an Illustrative Example

In order to describe the basic concepts underlying strategies and KPIs, we present a running example modeled after a fictitious vehicle manufacturer/reseller, the Steel Wheels company[1], who desires to improve its monitoring and decision making processes. In order to fulfill its objective, the company first models and analyzes its business strategy, described textually in terms of a business plan [29]. For representing the business strategy we use the Business Intelligence Model (BIM) [15], which has been successfully applied in a number of case studies. Compared to other modeling languages, such as i* [43] or Tropos [2], that are aimed towards social and agent modeling and capture their relationships, BIM focuses on the representation of business strategies, including concepts such as (i) goal perspectives from the Balanced Scorecard, (ii) indicators, and (iii) situations (external and internal factors) that are not present in other modeling languages. Nevertheless, other modeling languages could be used as well, as long as they are expressive enough to capture four key concepts for modeling a business plan: goals, business processes, situations, and indicators. The definition of these key concepts is further elaborated below.

Goals. Goals capture the objectives of an enterprise. Goals have been extensively used for intentional modeling [5, 12, 22, 43] and represent a situation that an actor wishes to achieve. For example, the main objective of Steel Wheels is "Revenue Increased". In order to achieve it, the goal can be decomposed into two alternative paths: "Costs cut" and "Fancy Designs Created". Furthermore, in

[1] Data for Steel Wheels is publicly available and included in the Pentaho Business Intelligence distribution which can be found at http://www.pentaho.com

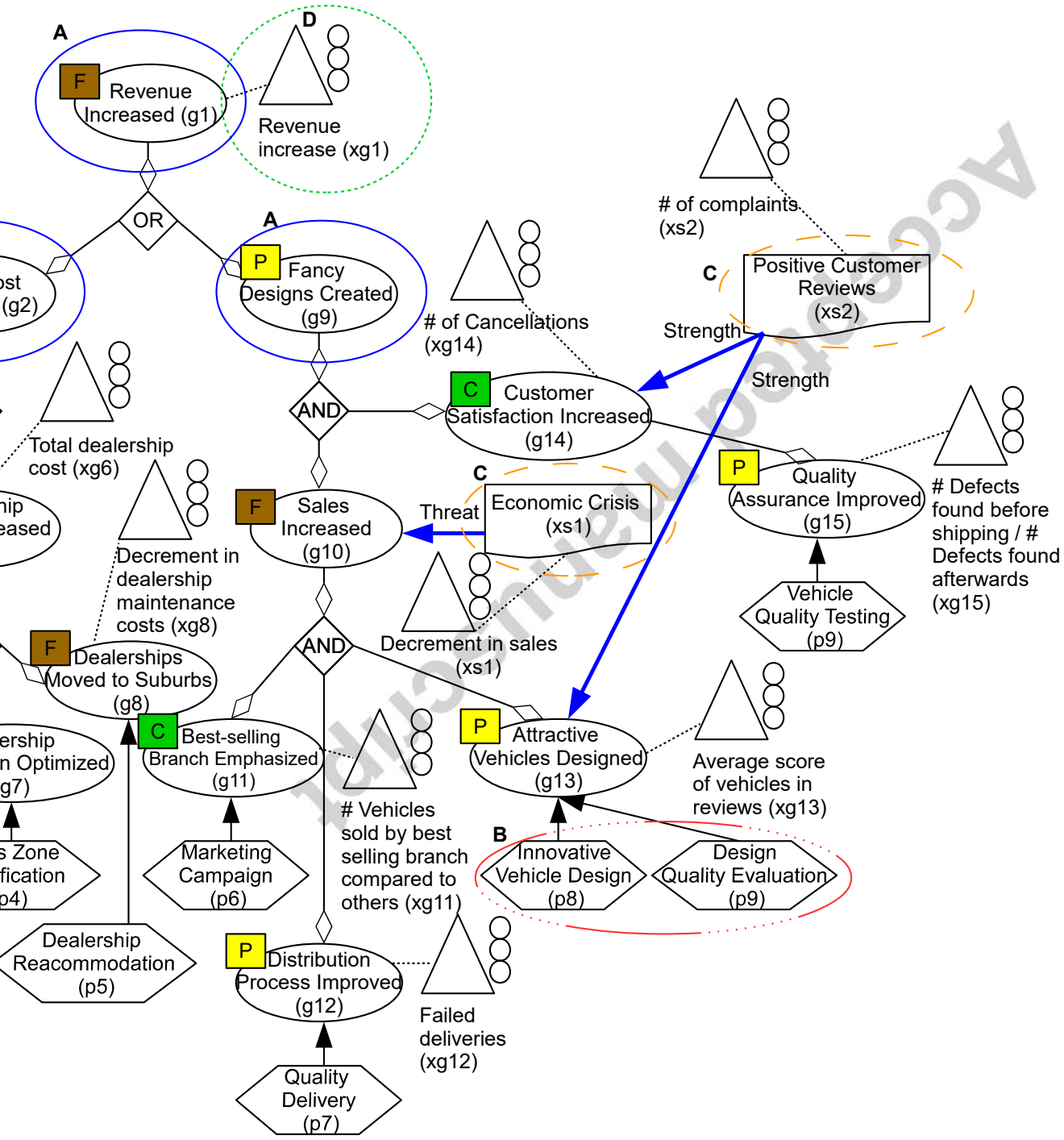

Figure 2: Steel Wheels business strategy

order to achieve success through the fancy designs strategy, both "Sales Increased" and "Customer Satisfaction Increased" goals should be met. Goals in BIM are related to situations that represent Strengths, Weaknesses, Opportunities and Threats (SWOT analysis [14]). Goals are selected from four complementary perspectives (Financial, Customer, Processes and Learning) inspired by Balanced Scorecard analysis [18] and may represent multiple levels of abstraction (Strategic, Operational/Decisional, Tactical/Information) as identified in the literature [17, 22].

Processes. Business processes are responsible for the realization of tactical goals. In this way, "Innovative vehicle design" and "Design quality evaluation" processes operationalize the goal "Attractive vehicles designed", while "Sales zone planning" operationalizes the goal "Dealership distribution optimized".

Situations. Situations represent knowledge about internal and external factors that influence goals either positively or negatively. For example, having "Positive Customer Reviews" is an opportunity (external, positive) of the "Fancy Designs Created" strategy, which helps the goal "Customer satisfaction". On the other hand, the situation "Economic Crisis" is a threat (external, negative) for the business, and hurts the goal "Sales Increased". The effect of a situation on different goals is captured by the semantics of the relationship as depicted in the model using the four SWOT classifications previously introduced.

KPIs. KPIs monitor goals, processes, or situations. Each indicator presents a target value (value to be achieved), a threshold (margin between good and bad performance), a current value and an unacceptable/failing value. According to these values, the KPI is normalized in the range [-1,1], describing how good or bad the measured element is performing. On the one hand, in the case of goals, indicators quantify the degree of satisfaction of a goal, and how much it is deviating from its target [1]. On the other hand, in the case of situations, indicators evaluate if, according to the provided data, a certain situation is considered active or not. It is noteworthy that although we describe indicators and goals in time-absent terms while modeling at business strategy level, i.e. 'Sales increased', a specific instance of the business strategy such as the current 5-year strategic plan, must always have a target point in time for all goals and indicators. Otherwise, their progress and status cannot be adequately monitored. Finally, it should be noted that when elements monitored by KPIs are related to each other this implies that their associated KPIs must be implicitly related to each other for the sake of consistency.

The BIM model for Steel Works is shown in Figure 2. The Steel Wheels business strategy has one main goal: increase revenue. In order to achieve this goal, Steel Wheels has two alternative courses of action at its disposal. On one hand, the company can decrease the manufacturing costs of its vehicles, thus making them more affordable, while also lowering their quality. On the other hand, the company can aim for high-quality creative designs, that are more expensive but also more attractive to customers. In turn, this course of action improves the image of the company, hopefully increasing its revenue.

Once the business strategy has been modeled and analyzed, the decision maker has a comprehensive view of her KPIs and the relationships between them. Now, we need to enable the decision maker to specify each KPI and calculate their values using available data. We accomplish this by using the Semantics of Business Vocabulary and Business Rules (SBVR), a business rule language adopted as an OMG standard [30].

3. KPI Definition

To conduct business analysis, we now need to gather feedback from on-going business processes, extracted from an underlying data warehouse (DW).

Unlike DW measures, which simply measure the performance of a given business process, KPIs are designed to measure the degree of achievement of a business strategy and its goals. For example, # of cars sold is a measure since we cannot say if 10.000 cars sold is an acceptable quantity for the company or not. Thus, KPIs encapsulate knowledge about the degree of performance to be achieved.

We can consider two different kinds of KPIs that are defined over a strategic model, according to how they are calculated: atomic and composite. Atomic KPIs can be calculated directly from the DW, either from one of the tables (matching a measure) or using a formula. For example, the KPI "Number of vehicles sold" which retrieves the total number of vehicles sold from the Sales table would be an atomic KPI. On the other hand, the KPI "Revenue increase" could be created as a composite KPI, calculated as the difference between the atomic KPI "Benefit" in the current and the previous year. The treatment of composite KPIs is studied in further detail in [1].

Atomic KPIs are defined in our approach by adapting Structured English, defined as part of the Semantics of Business Vocabulary and Business Rules (SBVR) [30]. This language allows decision makers to define KPIs on top of their business strategies by using a user-friendly language. Structured English uses keywords, terms, names, and verbs as syntactic categories. [2]. To avoid ambiguity and ease the derivation of queries, we will specify a grammar on Section 3.2 for defining KPIs in the spirit of Structured English.

According to the SBVR, **keywords** represent operators, logical or unary, that are applied to measures, allowing us to aggregate the data obtained. Since KPIs are normalized, single-valued, the definition of an indicator should present at least one of these operators. On the other hand, terms refer to metadata from the multidimensional schema, such as fact attributes, dimension attributes (properties), dimensions, and levels (concepts). Next, Names refer to individuals and exact values. As such, a name can refer to instances of levels or exact numeric values. Finally, a *verb* refers to a verb or prepositions as well as binary or ternary operators. Normal strings can be included within formal SBVR definitions although they carry no meaning. Some examples of how KPIs can be defined using this notation are as follows:

- "Revenue increase" (xg1), can be defined either as a single KPI with a binary operator: **total** benefit **with** year **equal to** 2004 *minus* **total** benefit **with** year **equal to** 2003
- "Number of vehicles sold" (xg10), defined as: **total** amount *of* Vehicles sold **with** year **equal to** 2004.
- "Number of cancellations" (xg14), defined as: **sum** Fact Count **with** status **equal to** Cancelled **and** year **equal to** 2004
- "Average score of vehicles in reviews" (xg13), defined as: **average** score **of** Vehicles

All the definitions presented are formalized in Section 3.3, by means of a grammar. The benefits of following this approach are that (i) indicators can be defined in a user-friendly, controlled language at the business level, and (ii) they can be included into a Business Dictionary (BD), thus they can be referenced and queried by other software and applications, used to generate documentation, or used as basis for specializing indicators (e.g. Average score of vehicles in Europe). This way, our technique helps other decision makers into defining their own indicators for their models, as well as re-use existing indicators. In the following section we take a deeper look at the Business Dictionary before describing the method to derive multidimensional queries from KPI definitions.

3.1. Business Dictionary

The Business Dictionary is an artifact designed according to the SBVR specification and its main function is to act as a semantic resource that stores business terminology. Each concept stored in the dictionary has the following properties:

- A name given to the concept (e.g. Sales)

[2] SBVR actually offers a visual syntax for these syntactic types in terms of colours; we omit this feature of SBVR for pragmatic reasons. Instead we will represent them as **keyword**, term, Name, and *verb*.

- A non-formal description
- A formal definition represented as a set of SBVR strings. Each individual SBVR string within the formal definition is represented using a different font format depending on the concept that it is referencing, as discussed above, or as a normal string if there is no concept referenced at all
- A general concept that denotes the classifier of the entry, following an "instance of" relationship (e.g. Madrid is an instance of City)
- A concept type for representing is-A inheritance between abstract concepts (e.g. a Car is a Vehicle)
- A list of synonyms of the concept, whether in the same dictionary or in a different one
- A namespace for identifying the concept (e.g. DataWarehouse for DW concepts)
- A list of notes that have no semantic meaning and allow the user to write additional annotations about the concept. Our framework includes one note for each KPI concept in order to preserve the current multidimensional query generated and allow the user, whether the decision maker or the query designer, to modify it if she wishes to do so

This set of properties is a subset of the properties listed for each concept in the SBVR specification [30], the rest of concept properties are designed to capture aspects of business rules and, thus, add unnecessary complexity for KPI specification. These properties enable us not only to store business terms, but also to (i) create KPIs that are formally defined, and can be queried in any moment, (ii) create hierarchies of concepts, including hierarchies of KPIs that are useful for data navigation, (iii) establish synonym relationships that can be exploited for translating business terms into technical references, e.g., benefit may refer to a KPI or it may refer to a value stored in the data warehouse (such as sales (iv) navigate through the concepts available, for example to identify the list of KPIs that are calculated by using a certain value from the data warehouse, and (v) extend the language with new keywords and operators.

For example, given the following two entries in the dictionary, we know that benefit is a business term used by decision makers, since it is a specific instance of Business Concept (General Concept), and therefore it is not a KPI. Furthermore, it is a synonym of sales, which is a DW concept that can be used as a valid translation for OLAP queries. Finally, we know that it does not represent a fixed value nor depends on other concepts since its definition is empty, and according to the URI it is stored within the steelwheels.bim model. If benefit was a KPI, then Definition would store its SBVR string, making reference to other concepts in the Business Dictionary.

Moreover, we also know that sales is a measure (Fact Attribute) in the data warehouse cube and thanks to its description we notice that it does not represent the number of units sold, but rather the gross income generated by them. As a measure, it can be used to calculate the value of an indicator.

Dictionary Entry: benefit
Description: Gross benefit obtained by the enterprise
Definition: n/a
Dictionary Basis: Business Indicator Dictionary
General Concept: Business Concept
Concept Type: n/a
Synonym: sales
Namespace URI: http://steelwheels.bim

Dictionary Entry: sales
Description: Represents the number of units sold by the enterprise multiplied by their price
Definition: n/a
Dictionary Basis: Business Indicator Dictionary
General Concept: Fact Attribute
Concept Type: n/a
Synonym: benefit
Namespace URI: http://steelwheels.mddw

If the first entry described an abstract concept that helps organizing the knowledge, such as "Marketing Concept", then its "Concept Type" property would be filled with "Business Concept", denoting that it is an specialization.

Entries within the Business Dictionary that take part in the specification and derivation of KPIs must be populated before or during the query derivation process in order to be used. Some of these entries are common to all scenarios, for example those concepts inherited from SBVR (Name, Term, Verb, and Keyword), or added by ourselves to support KPI formulae, such as Unary, Logical, and Boolean operators. These entries are pre-loaded into the dictionary in order to avoid unnecessary effort. However, specific business or data warehouse terminology related to each case study at hand must be loaded on a case by case basis.

As a knowledge base, most of the effort needed to build the dictionary is required at the beginning, when the business language has not yet been defined, and business terms need to be formalized, especially those included in the underlying data warehouse that present a technical nomenclature requiring to be translated into business language. Afterwards, if goals and indicators change, the effort required is minor, as the task of updating the dictionary is reduced to either updating its contents with a few new terms or defining new KPIs as a formula that uses already existing concepts.

With all these elements included in the dictionary, now we can make use of multidimensional structures for KPI definitions while maintaining the business terminology.

3.2. Multidimensional Structures for KPI Definition

KPIs defined in SBVR must be translated into executable queries so that the corresponding data can be retrieved from the target data warehouse. Deriving executable queries requires generating a valid query, (i) ensuring that the necessary data is stored in the DW, and that (ii) KPIs are correctly defined according to the DW structure.

To this aim, the derivation process is divided into two steps. First, we derive a set of OLAP actions, represented in OCL for OLAP [32], from each KPI definition. The goal of this intermediate derivation is twofold: (i) we ensure that queries are valid according to the model and that they are well formed thanks to OCL for OLAP, and (ii) we lower the complexity of the translation by dividing it in two parts. During this first step, we solve all the potential ambiguities, ensure that all elements involved in the query are known and act as part of valid OLAP operations. For example, it does not make sense to apply a logical operator over a dimension, it must be applied over attributes, thus we choose only synonyms of the concept to which the operation can be applied. Second, once we have obtained the set of OLAP actions, we transform it into executable MDX queries that can be interpreted by an OLAP engine. This second step will be covered in Section 4.

The first step in our query derivation process requires three different inputs: the KPI definition, the Business Dictionary with existing terminology, and a multidimensional representation of the data warehouse. The multidimensional representation of the data warehouse includes information about existing facts (center of analysis), fact attributes (measures, related to the performance of the business process), dimensions (context of analysis), dimension levels and attributes of the dimension levels. Typically, every BI vendor OLAP server (SSAS, Mondrian, Cognos, etc.) represents these data using their own specification, as it provides the multidimensional view of the underlying database. Although a standard for multidimensional metadata interchange was proposed (Common

Warehouse Metamodel) [31], only some vendors support it for representing their multidimensional metadata. Since the CWM specification has not been updated since 2003, it does not consider new technologies available, such as NoSQL, and due to its tight coupling to the logical level it requires additional effort to specify details that should be managed by the platform of choice of the user, making it ideal for exchanging implementation metadata but rather turn ill suited to act as a vendor-neutral modeling language for our purposes. Instead, in order to avoid the variability imposed by the different technologies and specifications, we use a multidimensional conceptual model that is platform independent [20] and abstracts from implementation details. Using this platform independent representation, we model the data warehouse that provides information for Steel Wheels[3].

According to the Steel Wheels business strategy, we are interested in the multidimensional modeling of the information regarding Sales and Budget assignation. Therefore, we will provide two multidimensional schemas: one which includes information about the costs of our strategy, shown in Figure 3, and another one which provides information about our sales process, shown in Figure 4.

The first schema represents the information about budget assignation and actual costs ("QuadrantAnalysis" fact). The fact attributes included in this schema are the planned "Budget" cost for each entry, the real "Actual" cost, and the difference between them, "Variance". As context of analysis, we know information about each "Region", "Department" and "Position", allowing us to browse the assignations in the budget as we need.

The second schema represents the information available regarding the "SteelWheelsSales" process (fact). The fact attributes included in this process are the "Quantity" of each product sold, as well as the total amount of the sale, "Sales". Regarding the context of analysis, we have information about products (Vehicles in business terms), such as its vendor and the product line. Additionally, we also have information regarding "Customers", such as their name and address, the "Markets" where the sale was performed, the month of the year when the product was sold ("Time"), and the current status of the corresponding order (Cancelled, Delivered, On Hold, etc.).

3.3. Structured English for KPI Specification

Once we have covered the underlying multidimensional models, we can proceed to interpret KPI definitions. We process each definition using our Structured English for KPI specification grammar, depicted in Figure 5. Structured English for KPI specification is a flexible grammar with multiple optional rules that allows the user to refer to a concept in several ways to account for the flexibility of the language, and manages the resulting ambiguity. Unlike in a traditional grammar, Structured English for KPI specification does not specify directly the terminals used by the language. Instead, these are extracted from the Business Dictionary. We provide a few samples of terminals included in our basic dictionary to provide the reader an idea of how KPI definitions are expressed.

[3]The original multidimensional schema is included as part of the Pentaho distribution together with the Steel Wheels database

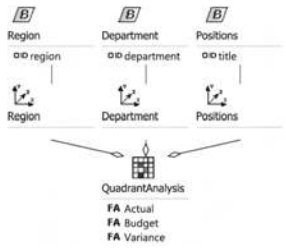

Figure 3: Multidimensional model for analyzing costs

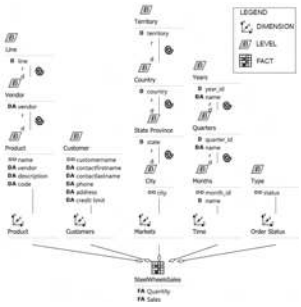

Figure 4: Multidimensional model for analyzing sales

```
(1) Indicator = expr [projectionKeyword dimensionSet] [restriction];
(2) expr = value {binaryOP value}*;
(3) value = [unaryOP] (term | propertyAccess) [levelIntroduction dimensionSet]
[conditionKeyword condition]?;
(4) propertyAccess = term levelOwnership term;
(5) condition = booleanCondition {boolOP booleanCondition}*;
(6) booleanCondition = expr compOP (name | expr);
(7) dimensionSet = dimensionOrLevel {boolOP dimensionOrLevel}*;
(8) dimensionOrLevel = (term levelOwnership term | term);
(9) restriction = restrictionKeyword condition;

projectionKeyword  = PROJECTIONKEYWORD; - Samples: by
conditionKeyword  = CONDITIONKEYWORD; - Samples: with
restrictionKeyword = RESTRICTIONKEYWORD; - Samples: restricted to
levelOwnership = VERB; - Samples: of | in | per
instanceOwnership = VERB; - Samples: of | in | per
levelIntroduction = VERB; - Samples: of | in | per
compOP  = COMPOP; - Samples: equal to | not equal to | higher than | lower than
unaryOP = UNARYOP; - Samples: count | sum | total | maximum | minimum | average
binaryOP = BINARYOP; - Samples: plus | minus | times | divided by
boolOP  = BOOLOP; - Samples: and | or
term  = TERM; - Samples: KPI | Fact Count | Sales | Quantity | cost | planned cost
name  = NAME; - Samples: Cancelled | Shipped | 2003 | Australia | USA
```

Figure 5: Structured English for KPI specification grammar following the EBNF ISO notation for symbols: definition(=), termination(;), alternative(—), optional([...]), repetition(...), grouping((...)).

According to this grammar, an Indicator (see Figure 5, line 1) is defined as an expression (indicator formula) that can be optionally projected using the multiple axis in the analysis cube or restricted to a certain data subspace. For example, **sum** cost **with** Department **equal to** Sales (xg6) would be an indicator expression. More complex indicators can be defined, for example using binary operators in the expressions or specifying multiple conditions, as in the case of **sum** Fact Count **with** status **equal to** Cancelled **and** year **equal to** 2004. Note that these indicators do not contain a projection clause, such as **by** Customer **and** City, nor a restriction, which we will exemplify later using line 9. The reason why no projection is included in these KPI definitions is that projections have been included in the language in order empower its usability as a tool for general OLAP querying and supporting ad-hoc analysis of KPI data, e.g., comparing KPI values across different vehicle lines. However, when computing a KPI to be loaded in the strategic model, it only makes sense to calculate a single value, thus although our tests include projections, no projections are defined in the KPIs included within the case study.

Expressions (line 2) can be simple or complex values that are obtained by applying successive binary operators. Binary operations are useful when calculating incremental indicators, by subtracting the values of the indicator in the previous period to the current one. An example of such expression is **sum** Total benefit **with** year **equal to** 2004 *minus* **sum** Total benefit **with** year **equal to** 2003

Each of these Values (line 3) is calculated by applying an optional unary operator to a value across a set of dimension levels. This is useful for example for calculating averages, such as **Average** score **of** Vehicles, since we require to specify the exact level (Vehicle) in order to obtain the correct result. A value can be represented by a term that may refer to a fact or dimension attribute, e.g. score (score of a Vehicle, implicit), or to a level in order to extract the value of its descriptor. Alternatively, a value can represent the explicit access to a property (line 4), as in score *of* Vehicles.

As it can be seen, although the concepts involved in the last two examples are the same, the semantics of the word "of" are different, as one is marked as a keyword whereas the other is marked as a verb. In the former, **of** denotes the introduction of a level for calculating the aggregated value of a measure using a dimension level (Vehicles), whereas in the latter *of* denotes the extraction of a property from a dimension level (Vehicles). Furthermore, a Value (line 3) can be constrained to meet a certain condition. This condition is independent from the set of levels specified for calculating the value.

Conditions (line 5) represent chains of boolean conditions that must be met by the cells in the cube. Each boolean condition (line 6) is composed by an expression (line 2) that is compared either to a constant value, represented by a Name, or to a value calculated by means of an expression. For example, we may wish to calculate the average score of vehicles only in USA. This way, the definition of the indicator in the previous example would be modified as follows: **Average** score **of** Vehicles **with** Country **equal to** USA. Another example of a more complex condition using an expression would be **with** Total benefit **of** Country **higher than** 50000, which would filter the calculus using those countries where the benefit obtained by the company is higher than the specified threshold.

Next, dimension sets (line 7) represent sets of dimension levels (line 8), used for aggregation and projection. Dimension sets are chains of levels or levels qualified by their dimensions, such as Vehicles **and** City **in** Markets.

Finally, a restriction (line 9) constrains the calculus of the KPI to a certain data space by means of a condition. A restriction is conceptually different from a condition in the sense that it defines a data subspace, essentially a subcube, in which the KPI value is calculated. For example, the restriction **restricted to** Country **equal to** USA **and** Total benefit **of** Vendor higher than 50000 defines a subcube conformed only by the data pertaining to USA for vendors that have obtained a total benefit of more than $50000. Restrictions are computed before any other OLAP operation, and their usefulness comes from applying their specification to all indicators in the strategic model. This way, restrictions allow decision makers to navigate the data, by defining and altering the data space over which all indicators are being calculated. We will exemplify this behavior in Section 5.

The remainder set of rules (terminal rules) is designed to connect concepts included in KPI

definitions with SBVR constructs. It is worth noting that given the nature of our approach, (i) tokens only represent SBVR constructs that are extracted from the dictionary, and (ii) any non-SBVR construct will just be ignored during the parsing, for example, the word "sold" in the definition of xg10 (Section 3). This allows the definition of KPIs with extra non-semantic information added to the definition. The interested reader can find more details in Section 6.

4. Query Generation

The resulting set of OLAP actions generated from parsing a KPI definition is represented in an intermediate language, OCL for OLAP [32]. Standard OCL (Object Constraint Language) [41] was initially proposed as a specification language that could be interpreted for verifying constraints over UML models. It allows us to ensure that the defined constraints are valid at design-time according to the model at hand, and respected at runtime. Extensions of the language have been proposed, in order to take advantage of the ability to specify model-compliant actions.

4.1. OLAP Actions in OCL for OLAP

OCL for OLAP is an extension of the OCL formal language allowing us to specify and validate OLAP actions over multidimensional models structured around facts and dimensions. It includes with its specification a derivation to SQL Language. However, deriving directly to SQL makes us dependent on relational technology, whereas deriving to MDX allows us to abstract from the underlying database technology, which will be dealt with by the OLAP Engine. Nevertheless, its usage is twofold, (i) it acts as an intermediate step between KPI definitions and the MDX querying language, making the translation easier, and (ii) allows us to ensure the set of actions generated is valid according to the multidimensional models of the underlying data warehouse. As an extension of OCL, OCL for OLAP is defined using already existing standard OCL constructs, thus it can be interpreted with a standard OCL engine for validating the set of actions generated. Furthermore, it is a closed language (all the operations return a cube) and includes all the basic OLAP operations. Therefore, although the complete translation of arbitrary SBVR to OCL is considered to be a challenging transformation [3] because it lacks completeness, our focus on KPIs and OLAP queries allows us to overcome this obstacle. As a result, the generation of OLAP actions is as follows:

1. Values identified correspond with sets of cells in the cube specified by the multidimensional schema. By specifying a term corresponding to a fact attribute, the decision maker denotes her interest in operating with the value, thus a *dimensionalProject(Source::Attribute)* OCL for OLAP operation is performed. This operation extracts the relevant set of cells from the cube, allowing further operations to be performed. For example, given the definition **sum** Total benefit, the corresponding projection would be *SteelWheelsSales→dimensionalProject(SteelWheelsSales::Sales)*.

2. Whenever an unary operation is performed over a given value, the corresponding OCL operation over the set of values previously projected is applied. These operations can be *sum()*, *count()*, and other unary operators specified in the UnaryOP rule. Using the previous example, the unary operator **sum** would be applied as *SteelWheelsSales→dimensionalProject(SteelWheelsSales::Sales)→sum()*.

3. For unary operators including a set of dimensions, the set of dimensions is first added to the query by means of *addDimension(Dimension,additivity)*. Then, the level of detail required on each dimension (i.e. hierarchy level) is adjusted by means of *rollUp(Dimension,level,additivity)*. Afterwards, the corresponding dimensional projection and unary operator are applied. For example, let us expand our previous definition as **sum** Total benefit **of** Vendor. The corresponding OCL for OLAP actions would be *SteelWheelsSales→addDimension(Product,sum())→rollUp(Product,Vendor,sum())→dimensionalProject(SteelWheelsSales::Sales) →sum()*

4. Whenever a binary operation is performed over a pair of values, the result is obtained by iterating over the set of cells corresponding to each value. Therefore, a binary operation is translated to an *iterate(value1,value2,result = 0 | result = value1 operator value2)* operation over the cube. For example, given the definition Total benefit *minus* Quantity, the resulting set of OCL for OLAP operations would be *SteelWheelsSales→dimensionalProject(SteelWheelsSales::Sales)→dimensionalProject(SteelWheelsSales::Quantity)→iterate (a:Sales,b:Quantity,result = 0 | result = a-b).*

5. In the case of a complex calculus (aggregation or binary operation), the whole set of OLAP actions is wrapped into a *let... in* OCL statement and the complex value is projected into a variable for further use. The statement groups all the required operations for the calculus into a resulting cube. The inclusion of let in statements allows us to verify the correctness of the set of operations aimed at calculating a value and makes it easier to derive MDX queries from OCL for OLAP. We will see an example of this statement together with the condition defined in the next point.

6. For conditions over dimensions, levels, and values we first translate the expressions involved in the condition. Afterwards, the condition is translated into a *sliceAndDice(cell | condition)* operation. Property accesses and constants are used directly in the slice & dice operation, while complex values are retrieved by referencing the variable where they are stored. For example, given the definition **sum** Total benefit **with** Country **equal to** USA, the corresponding OCL for OLAP generated would be:

   let SteelWheelSales1 = SteelWheelSales→addDimension(Markets,sum())→rollUp(Markets,Country,sum())→ sliceAndDice(c | c.Country = 'USA')→dimensionalProject(SteelWheelsSales::Sales)→sum()→dimensionalProject(SteelWheelsSales::result1) in SteelWheelsSales1→dimensionalProject(SteelWheelsSales::result1).

7. In the case that the definition includes a set of projection dimensions, each dimension is added by using the same set of actions as in step 3 after the unary operator has been already applied.

8. Finally, conditions specified in the restriction clause are translated into *sliceAndDice(cell | condition)* statements that restrict the calculus of any value during the translation into OCL for OLAP.

Given that the KPI definition is written in business terms and the resulting MDX query needs to be written in multidimensional terms, referencing DW dimensions and levels, a translation needs to be performed during the derivation process. For this task, we use the Business Dictionary. First, each business term in the KPI definition is extracted from the dictionary. If the business term includes a valid synonym candidate for the translation, we substitute it by its candidate. A candidate is valid only if it corresponds to the type of element required by the grammar at the point it is being parsed. For example, if we are trying to extract a value, valid candidates are attributes and levels (implying access to the level descriptor). If multiple candidates are available, we query the user in order to select the chosen candidate for this query generation. On the other hand, if no candidates are available, we evaluate if the term includes a formal definition. If this formal definition exists, the term is substituted by its formal definition. Otherwise, we try matching the term directly against the data warehouse schema. If no valid mapping is found, then we query the user for a valid mapping.

Second, each new concept mapping is added to the Business Dictionary, by adding the missing concepts and establishing a synonym link between the business term and the data warehouse term.

By following these steps we have generated an intermediate OCL representation of the OLAP actions to calculate the KPI value. For example, the indicator xg1 is transformed into the following OCL statement:

```
let SteelWheelsSales1 = SteelWheelsSales->sliceAndDice(c |
c.Time.Years.year_id.d == 2004)->
dimensionalProject(SteelWheelsSales::Sales)->
sum()->dimensionalProject(SteelWheelsSales::result1) in
let SteelWheelsSales2 = SteelWheelsSales1->sliceAndDice(c |
c.Time.Years.year_id.d == 2003)->
dimensionalProject(SteelWheelsSales::Sales)->
sum()->dimensionalProject(SteelWheelsSales::result2) in
SteelWheelsSales2->dimensionalProject(SteelWheelsSales::result1)
->dimensionalProject(SteelWheelsSales::result2)->
iterate(a:result1;b:result2;result3:double | result3 = a - b)->
dimensionalProject(SteelWheelsSales::result3)
```

This set of actions can be validated over the multidimensional schema and processed through an OCL for OLAP compiler to transform it into MDX queries as we will show in the following section.

4.2. Transforming OLAP Actions into MDX Queries

The set of OCL for OLAP actions [32] generated in the previous step must be translated into a MDX query in order to be interpreted by the target OLAP engine. Despite both languages representing OLAP operations, the transformation of OCL for OLAP into MDX is not completely straightforward. This is due to several reasons. First, the correspondence in MDX of certain OCL for OLAP operations depends on the order that they are executed. Second, although MDX is considered a standard, not all OLAP platforms support the whole specification (for example, Mondrian does not support the creation of subcubes natively). Therefore, these aspects must be taken into account during the MDX query generation process in order to ensure that they can be directly executed into the target BI platform. In our implementation, described in Section 6, this is the responsibility of the compiler.

The steps to generate an MDX query from the set OLAP actions is as follows:

1. Each *let... in* statement will correspond to the definition of a "WITH MEMBER" clause in MDX. The last *dimensionalProject(Source::Attribute)* value becomes the new member in the cells of the cube ([Measures].[member]), whether an aggregation or a binary operation. The rest of the operations are included within the "AS" part of the clause and will define the calculus for the member.

2. All the dimensions added by means of *addDimension(Dimension,additivity)* are included as active dimensions on the cube. Each active dimension can only have one active level. Rolling up or drilling down to another level implies changing the current active level.

3. For each condition specified, the set of dimensions involved is added to a set of filtered dimension levels together with the corresponding condition. Each dimension can have more than one level filtered and the same level can be filtered multiple times. If multiple levels of one or more dimensions have been filtered, when we require to include them into a statement, we first create a "WITH SET" clause in MDX. This clause defines the set of members filtered with the first condition. Then, for each successive condition specified, we filter those members not included in the previous condition by using the "EXISTS" clause.

4. Unary operators make use of the current active and filtered dimensions to perform the aggregation. In the case of *sum*, *average*, *minimum*, and *maximum* operations the last *dimensionalProject(Source::Attribute)* value is aggregated, whereas in the case of *count* only the set of dimensions is used.

5. Whenever an aggregation is executed, or when the final "SELECT" statement is created, the set of active dimensions is added to the dimension list of the clause and cleared from the active list. The same procedure is applied to filtered dimensions specified by conditions.

Following these rules, we can create complex MDX statements that correspond to the set of operations specified in OCL for OLAP. Furthermore, we are able to represent subcubes by using filtered sets, even when the underlying technology does not support subcube operations. An example of an MDX generated by following these rules is the MDX query for indicator xg1 which is as follows:

```
WITH SET [depth1-0] AS FILTER([Time].[Years].Members,
[Time].[Years].CurrentMember.Name = '2004')
MEMBER [Measures].[result1] AS SUM([depth1-0],[Measures].[Sales])
SET [depth1-1] AS FILTER([Time].[Years].Members,
[Time].[Years].CurrentMember.Name = '2003')
MEMBER [Measures].[result2] AS SUM([depth1-1],[Measures].[Sales])
MEMBER [Measures].[result3] AS [Measures].[result1]-[Measures].[result2],
FORMAT\_STRING = '\#.00'
SELECT [Measures].[result3] ON 0 FROM [SteelWheelsSales]
```

At this point, we have transformed a KPI defined in Structured English for KPI definition into an executable MDX query that can retrieve the value of the KPI from the data warehouse. In the next section, we exploit this process not only to retrieve the value of each KPI, but also to navigate the data while maintaining a strategic point of view and transforming the business strategy into a dynamic strategic dashboard.

5. Data Extraction and Navigation

Once we have obtained the MDX representation for each KPI, we execute the queries against the OLAP server. The results from these queries are loaded into the atomic KPIs, and normalized according to the values specified [17] for each KPI. After atomic KPIs have been loaded, composite indicators are calculated using their values, resulting in a comprehensive, global view, of the business strategy, including the performance associated to each element. The result can be seen in Figure 6. For the interested reader, please refer to Appendix A for more information on the dataset, a comprehensive list of KPIs, their definitions, values, and generated queries.

According to the results obtained, the Steel Wheels company is meeting its main goal (green light), increasing its revenue. As expected, since the company is focusing on the "Fancy Designs created" course of action, the KPIs show that the "Low-Cost Designs" approach is providing mediocre results, as shown by the three indicators failing to meet their mark, two of them (xg4 and xg6) with yellow status, and one (xg3) with red status. On the other hand, the "Fancy Designs created" approach is obtaining average results. The approach is exceeding the target amount of sales (xg10), but although the "# of complaints" is low (s1 is active), the "Number of cancellations" is higher than its mark (xg14 presents a yellow light), thus it is expected that customer satisfaction decreases, hurting at the same image of the company.

As shown, our approach allows the decision maker to analyze his business strategy by using real data, as opposed to estimations only, thereby enabling strategic monitoring through the defined KPIs. This allows the decision maker to identify potential problems in the business processes of the company, e.g. there may be a potential problem in the distribution and delivery processes, as well as in the business strategy itself, e.g. despite average results in the delivery process we are meeting our goals in sales and revenue increase, were the target values too low? Have the problems in our delivery process not impacted our revenue yet? Using this information, the decision maker may use what-if analysis to evaluate the best course of action, by manually overwriting atomic KPI values to

Figure 6: Steel Wheels business strategy loaded with data

simulate the company taking action to improve them and analyzing how composite indicators would change according to these actions.

Furthermore, the decision maker can navigate the data using the business strategy model. In order to navigate the data, the decision maker needs to specify a global constraint. Global constraints are specified in the same terms as KPI definitions, by using the **restricted to** keyword followed by a condition statement as presented in Section 3.3. For example, the global constraint **restricted to** Country **equal to** USA will restrict the values of all the indicators in the model to the data subspace corresponding to USA. A global constraint essentially defines a partial view of the underlying data warehouse from a strategic perspective, forcing all indicators present in the model to be calculated from this new subspace instead of the original one. By redefining the active constraint and moving from one data subspace to another, decision makers can compare the performance of each subspace with regards to the objectives defined at the business strategy level. This way, one could move across data subspaces comparing different regions, different products, or different customer segments, detecting different problems in each of them. Conceptually, the result is similar to navigating ad-hoc an OLAP cube that not only includes the values for all the objectives, but also their relationships (between goals in the model) and factors affecting them (situations), while interpreting the results in a visual way.

While powerful, data navigation using constraints has a number of limitations to be taken into account. First of all, while a single global constraint can be a complex condition including multiple boolean conditions and aggregated calculus for defining the data space, only one global constraint can be active at a time. This is a limitation of the view, not the grammar, for the sake of consistency. For example, it would not make sense to restrict sales to USA while analyzing the customer satisfaction in Europe, and then interpreting the influence that European customer satisfaction has on USA sales.

Second, global constraints may define a data subspace that does not intersect with the space defined in some KPIs. In these cases, the MDX query generated will still be valid syntax-wise but will not return any value. To address these cases, the corresponding KPI will be disabled in the model, since it does not have a value or it is unknown, and thus, it cannot be used to make decisions or identify problems in the business strategy. For example, an alternative for KPI definition would be to define generic KPIs not bound by time in their definition but, instead, create a global restriction that captures this aspect, such as **restricted to** year **equal to** 2004. Unfortunately, it is not applicable to our scenario because the "Quadrant Analysis" cube that provides information for half the KPIs in the strategy lacks any time dimension, and thus, cannot be restricted to a specific year, since there is simply no data available.

For every other KPI included in the business strategy, the application of a global constraint allows the decision maker to analyze certain areas in detail in order to evaluate what KPIs are succeeding or failing in them. As a result, she can make specific decisions corresponding to each area individually, enabling her to address the specific needs of each data space.

After the application of global constraints, we have closed the proposed cycle depicted in Figure 1, starting from the definition of KPIs up to the extraction of their values and navigating the data warehouse using a strategic perspective. In the following sections, we will describe our architecture and implementation for supporting this process, and we will discuss the related work in the area. Afterwards, we will provide some discussion about the proposed approach and its limitations and will sketch the future works.

6. Architecture and Implementation

Our architecture is composed by several components that provide support for the different tasks in the process, starting from the initial business strategy modeling up to executing and retrieving the values from the different MDX queries generated. An overview of the different components and their relationships can be seen in Figure 7, and are detailed as follows:

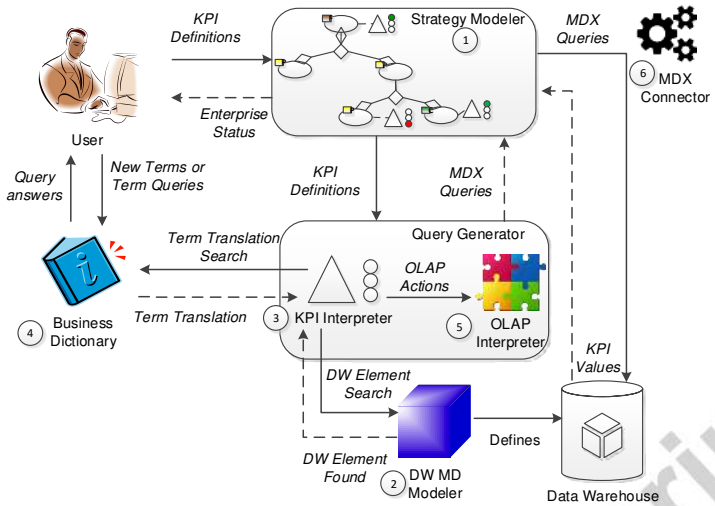

Figure 7: Components in the architecture of our approach

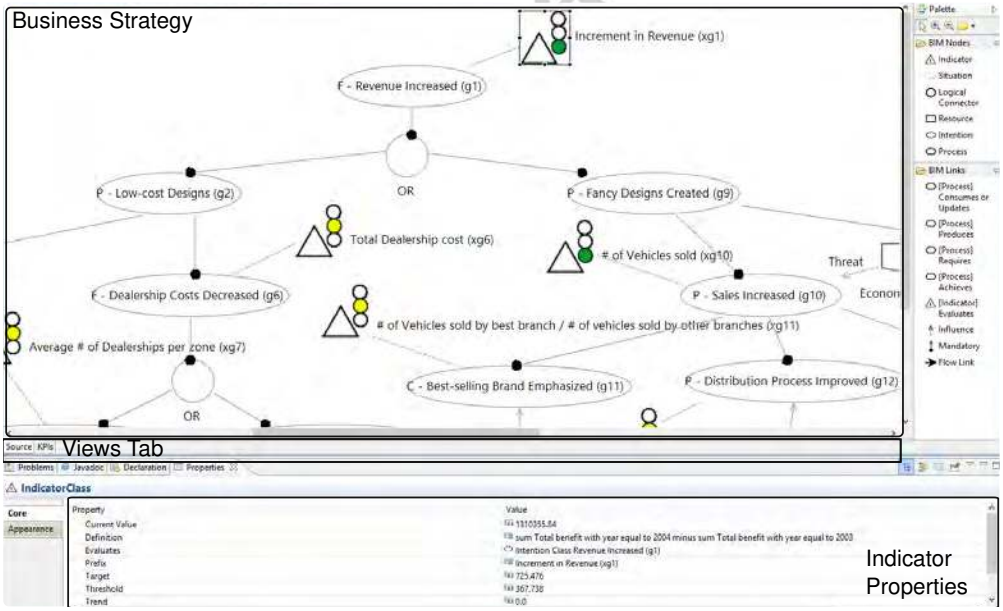

Figure 8: Implementation of the Business Strategy Editor acting as interface in our approach

Figure 8: Implementation of the Business Strategy Editor acting as interface in our approach

- First, the **Business Strategy Modeler**, seen in Figure 8 enables the modeling of key concepts within the business plan that were discussed in Section 2. This component acts as an interface for the decision maker, allowing her to model the business strategy in the Source tab and to define KPIs in the KPI tab. It also provides the means to present the information back to the user once the data has been retrieved. Our current implementation is based on the BIM metamodel [15] and it is implemented in Eclipse by means of the Ecore Modeling Framework (EMF) [37]. Strategic modeling is done through the palette on the right side, whereas KPI are currently defined by inputting formatted strings in modified TextBox components in the KPI tab, generated according to the list of KPIs included in the business strategy. The Source tab contains the Steel Wheels Sales business strategy of our case study and the values of the different KPIs already loaded. By analyzing the strategic model and comparing the indicator results obtained, the user can estimate the actual importance of external and internal factors modeled. For example, although there is a threat of an economic crisis, the company is succeeding to meet its sales mark (g10). Furthermore, It is important to note that, although visually we only represent the status of each KPI (depicted as a traffic light), the model also stores the quantitative values which can be seen through the properties pane.

- Second, the **Data Warehouse Modeler** enables the data warehouse designer to conceptually model the structure of the data warehouse and OLAP cubes [40, 4] that have been defined. These conceptual models play a key role when deriving executable queries from KPI definitions. As in the previous case, our current implementation is based on the UML profile for data warehouses defined in [20] and it is implemented in Eclipse by means of EMF.

- Third, the **KPI Interpreter** performs the process of interpreting and translating KPI definitions created by the decision maker into OLAP actions, such as adding dimensions, drilling down, etc., represented in our case using the formal language OCL for OLAP [32]. In order to perform this task, the Interpreter takes as input the KPI definitions, one or more data warehouse models that support these KPIs, and a Business Dictionary. The KPI Interpreter is implemented by using ANTLR [34] for the generation of the code associated with the grammar. However, the KPI Interpreter only makes use of the parser generated, as the lexer is substituted by a custom component that makes use of the Business Dictionary to interpret definitions in SBVR terms.

- Fourth, the **Business Dictionary** stores the formal definitions of business concepts and KPIs in order to aid with KPI definitions and translations. It captures and represents business concepts according to the Semantics of Business Vocabulary and Business Rules (SBVR) [30] specification. The concepts stored include key concepts related to the language used, business terms, KPIs and data warehouse elements. The Business Dictionary is part of a custom Java library that implements the basic concepts required from SBVR for implementing our approach. By using the Business Dictionary, we can make quick transformations of plain strings into SBVR typed strings, by searching the concepts in the dictionary, therefore saving time when specifying KPI definitions. For example, all the KPIs definitions created in this paper were automatically transformed from plain strings after having loaded all the business concepts in the dictionary and specified the mappings.

- Fifth, the **OLAP Actions Compiler** translates the set of OLAP actions generated by the KPI Interpreter in OCL for OLAP into MDX queries by following the approach presented in Section 4. This compiler is also written using ANTLR 3, like the KPI Interpreter. However, in this case no substitution of the lexer is required, as we are interpreting OCL which can be defined in terms of traditional tokens.

- Sixth, the **OLAP Connector** takes as input the MDX query generated and executes it against the OLAP server. Afterwards, it retrieves the results and passes them back to the Business

Table 1: Domain requirements satisfied by features in our solution

Requirement	Description	Proposed method features
R1 Interpretation	A method should allow users to link and visualize their performance indicators within their strategic context, in order to facilitate the identification of cause-effect relationships and to understand the status of their business	Strategic modeling language designed to facilitate the representation of internal and external strategic information, as well as linking indicators to goals and establishing relationships between them
R2.1 Traceability	A method should allow users to verify what sources are being used to calculate their performance indicators, effectively allowing them to detect incomplete information or incorrect strategic assumptions	Business dictionary linking business concepts back to facts and dimensions in the data warehouse and ultimately to their data sources through them
R2.2 Consistency	Given a single input (data, context, and performance indicator definition), a method should always provide the same result for any given execution	Deterministic derivation and calculus of performance indicators by means of a grammar, providing at most a single possible result for any given indicator definition. This requirement has also been tested during our empiric evaluation
R3 Encapsulation	A method should abstract users from unnecessary details, facilitating sharing and reuse of performance indicators	Definition language that allows decision makers to automatically derive indicator formulae from business terminology. Business Dictionary to store and recover KPI definitions allowing their reuse by other decision makers

Strategy Modeler. It is written in Java using OLAP4J 1.1 [16], an open API for querying OLAP engines. Version 2.0 of OLAP4J is being developed at the moment. For this case study, the OLAP Connector connects and executes queries against Mondrian OLAP Engine 3.7.

7. Evaluation

Different aspects in our proposal can be evaluated, ranging from its usability to its performance and precision. A thorough evaluation requires a dedicated paper covering a family of experiments, analyzing their results and insights. This evaluation is planned as a future work to identify ways to improve the effectiveness of the approach. Nevertheless, at the moment we can perform an initial theoretical and empirical evaluation to validate our solution.

On the one hand, we evaluate the suitability of our proposal to tackle the initial problems posed. In order to do this, we follow the method described in [38], where the authors formulate the objectives of their approach as requirements to be met. In our case, we reformulate our motivating problems into four requirements that must be satisfied. Then, for each requirement, we discuss how the components and features in our solution satisfy it. We summarize this information in Table 1.

On the other hand, using our implementation we perform an initial evaluation including performance, precision, and recall of the queries generated. Performance is the most traditional way of evaluating OLAP systems, such as by means of the TPC-H [39] benchmark for decision support

systems. This benchmark compares the performance of different BI systems by generating artificial databases over which queries are executed. In our case, instead of comparing multiple BI systems, our objective is to evaluate if there is any performance impact on the queries generated by our approach compared to manually designed ones. Since KPIs are tailored for each industrial sector and company, there is no standard, and thus we have used the ones in our case study as a sample. In order to evaluate the performance of the queries generated, we tested them against those coded by developers directly. We tasked two developers with coding all the KPIs in our case study using the SteelWheels data warehouse and compared their execution times with those automatically generated by the approach. Results shown that there was no significant difference in the execution time regardless if the queries were generated automatically or manually designed.

Some factors may influence this comparison, such as the planner included in OLAP engines which optimizes MDX queries and may reduce any notable differences between queries, or the limited volume of data in the data warehouse used. We plan to perform this comparison again on larger sized (at least gigabytes of information) data warehouses as part of our family of experiments to see if any significant differences arise.

Regarding precision and recall, the process of query derivation that we have defined includes the intermediate OCL for OLAP language, which serves to validate the set of actions executed. Furthermore, the approach generates an exact translation of Structured English for KPIs definitions into executable queries. In this sense, we could say that the precision and recall of queries generated can be considered 100% with regards to their definitions. However, we are aware that the main point of failure is writing KPI definitions themselves (usability). Therefore, we also ran an initial test by having both developers use the language to define KPIs. Their use of the language led to some invalid definitions and errors due to their inexperience and the lack of prioritization operators in the language, which were solved with the aid of language designers. A more thorough evaluation of the usability of the language is planned as part of the family of experiments, which will help to determine the precision and recall of the queries compared to what users have in mind. We expect this evaluation to help in locating the main usability issues, and determining whether these may be solved through a computer-aided solution, such as by means of software-driven suggestions and autocomplete features added to the tool.

The conclusion of our initial evaluation is that the proposal provides a significant speedup in the time required to prototype KPIs, which can be explained thanks to the focus of our approach on the specification of KPIs. This reduces the need for the user to invest considerable time in defining complex multidimensional operations and learning syntax. We should note, however, there is a one-time overhead cost introduced by building and evolving the strategic models. This task may require several hours if there is no clear business plan defined, but the resulting model can be re-used and maintained up-to-date as new ad-hoc queries arise, allowing users to define queries at the BIM level instead of at the warehouse level, improving the interpretability of the results and also enabling other interesting applications (e.g. SWOT Analysis).

8. Related Work

In this section, we present related work in KPI definition and business modeling. Initial works presented in [18] and [19] introduce the concepts of Balanced Scorecard and Strategy Maps. The Balanced Scorecard [18] has been one of the cornerstones in decision making for a long time. Its great advantage is that it maintains a summary of the business objectives along with KPIs. However, these objectives and KPIs are not modeled nor formally defined, thus the relationships between strategies, goals and indicators are unknown. Therefore, Strategy Maps [19] were proposed to deal with cases were the business strategy was apparently not working despite all the indicators being in place. Strategy Maps describe how the enterprise creates value combining the different perspectives present in the Balanced Scorecard. However, they do not address the lack of formalism of the constructs used. Furthermore, they do not provide any mechanism to analyze and assess the effectiveness of

the strategy modeled. As a result, the current practice is to identify and test KPIs in iteratively [33] according to the envisioned business objectives. Then, these KPIs are manually implemented into dashboards [10] in order to provide a detailed view of certain KPIs and enable some level of analysis.

In recent years, some works have addressed the lack of formalization in enterprise modeling [35, 15, 36]. In [35] the authors formalize the attributes of Performance Indicators and integrate them within a modeling framework that captures their relationships in order to enable the evaluation of their influences. Similarly, in [15], the authors propose the Business Intelligence Model metamodel (BIM) that includes a formalization of Performance Indicators as well as their existing relationships and a number of formal analysis techniques. Additionally, the BIM model includes additional elements that we have presented in previous sections, such as Situations, and even Business Processes. Among the differences between both works, one especially relevant is that while in [35] goals exist only in terms of Performance Indicators, i.e. "it is required that the state is achieved in which $PI27 \leq 48h$", [15] defines goals independently of the existence of Performance Indicators, which are used only to monitor the degree of achievement of an associated goal. Finally, in [36] the authors specify a series of Awareness Requirements over a requirements elicitation model of the DW, in order to model constraints which should be monitored. While all these works address the formalization of KPIs and other elements involved in enterprise modeling, all of them focus on design time and do not allow the decision maker to conceptualize KPIs, derive them into executable queries, and retrieve their value once the model is built.

In parallel, some works [42, 6] have explored the applicability of semantics and ontologies for defining KPIs for business activity monitoring. In [42], the authors define an ontology to model KPIs and mapping them to metrics related to the execution of a business process. Then, using these concepts, they define several formulas for calculating the KPIs, relating the KPIs to events within the event log generated by the business process execution and allowing them to show the values of the KPIs to the users. In [6], the authors propose an improved KPI ontology for defining KPIs that covers certain shortcomings of the work in [42], including also the definition of the analysis periods, data measures and derived measures. Unfortunately, these approaches cannot be applied in general enterprise modeling scenarios, since (i) the definition of KPIs is based on data repositories rather than on event logs, thus a different data query and extraction process is required, and (ii) the personnel involved is different, since the proposals focus on business process managers, whereas in general enterprise modeling users can range from local managers interested in monitoring their branch to high level executives monitoring the whole enterprise.

Nevertheless, more recent works using ontologies have moved further up in the enterprise layers, reaching enterprise-wide KPIs. These works are the most similar to ours. They highlight the importance of managing and maintaining up-to-date knowledge about enterprise KPIs. In [7, 9] the authors propose a collaborative framework based on KPIOnto for sharing PIs and KPIs to track the performance of collaborations at the strategic level. The information is stored in the form of an online dictionary that can be continuously updated, much similar to our proposal, which highlights the importance of always maintaining up-to-date KPI information. However, while these works focus on the consistency of the definition of KPIs over the ontology, we focus instead on their automatic derivation. Therefore, we can consider these works as complementary to ours.

Finally, it is worth highlighting that the main purpose of KPIs is to monitor objectives. While our approach enables users to understand the strategic context for each KPI, such as situations affecting their associated goal, or the status of other goals, it is not designed to perform systematic KPI monitoring, as it does not address their aggregated nature, in turn overlooking problems that are hidden within certain parts of the KPI. Instead, we advocate to make use of automatic and guided approaches for this task in order to locate latent problems. In this sense, drill-down techniques that perform an automatic systematic analysis, such as [24], or those that provide semantic-aware drill down operators for KPIs [8] and data analysis [28] can aid in monitoring latent problems within KPIs, and are complemented by the strategic context provided to the user thanks to our proposal.

9. Discussion and Future Work

In this paper, we have presented a novel approach to relate KPIs included in the business plan and the Balanced Scorecard [18] with business strategies and goals. Our proposal presents several advantages. First, all indicators are related to their respective goals, thus the decision maker can precisely identify which goals are having problems. Second, our approach not only allows the decision maker to model the business goals and indicators, but also allows her to analyze the business strategy using all the information in the underlying Data Warehouse, thus transforming the business strategy into a powerful dashboard. Third, KPIs are defined by using our SBVR for KPI Specification language, allowing the decision maker to perform quick modifications, without requiring knowledge of how the Data Warehouse is structured at logical level. Finally, our approach supports a combination of real data and what-if analysis, allowing analysts to compare expectations with reported results, thereby helping them identifying existing problems.

As we have shown, our framework allows both decision makers and data warehouse designers to extend the knowledge base of the enterprise by adding new concepts and mappings to the Business Dictionary artifact. Using this knowledge base, KPI definitions without semantics can be automatically transformed into semantic definitions using SBVR, and then mapped into multidimensional queries that retrieve the values of the corresponding KPIs. This is specially relevant because concepts can be reused across tasks and processes. Furthermore, the framework is capable of reading the data warehouse representation and load the corresponding concepts in the dictionary, thus avoiding that data warehouse designers spend considerable time on a repetitive task.

The main limitations of the approach concern ambiguity management and expressivity. On the one hand, regarding ambiguity management, as the number of concepts rise, so does the number of synonyms. In this scenario, if concepts are not separated in different namespaces, the decision maker will be asked to clarify a higher number of translations in order to avoid misinterpretation. A similar situation occurs as the grammar is relaxed in order to allow more variability in the way that KPIs can be expressed. Therefore, as the complexity of potential mappings across concepts rises, the overhead introduced by ambiguity also rises.

On the other hand, regarding expressivity, although concepts can be added to the Business Dictionary to extend the language, adding new functions requires more effort, since some specific functions that are supported in MDX are not directly included in OCL for OLAP [32]. For example, there is no direct counterpart for interpolation and other statistical functions. Additionally, while the intermediate language could be extended to support additional functions, not all vendors provide support for all MDX functions, thus, in these cases a different type of query should be generated depending on the target platform. This is especially relevant for defining templates and UDFs (User Defined Functions) that can simplify the definition of KPIs. For example, at the moment it is necessary to either explicitly state the years that take part in the calculus of a KPI or introduce in the dictionary the concepts of current and previous year as direct translations to the years they represent. A more powerful solution is to allow the user to define UDFs and substitute current and previous concepts by their dynamic, complex MDX functions, avoiding the need to update neither KPIs that represent increments nor the dictionary as time passes.

In practice, it is easier to use the language for rapid prototyping and obtaining the initial MDX query. Then, modifying this query with any additional functions that the user wishes to include from the MDX specification. Nevertheless, as shown in our case study, the language is sufficiently powerful to derive executable MDX queries for traditional definitions of KPIs without requiring any modification.

As part of the future work, we plan to run a family of experiments to further evaluate the usability and flexibility of the approach not only for KPI definition, but also to simplify general multidimensional query generation, for example during OLAP analysis. To this aim, we plan to extend the use of the Business Dictionary to allow for macro expansion and UDFs, by being able to define and reuse concepts that expand into other semantic concepts, thus further simplifying

the specification of general OLAP queries. Additionally, we are also considering introducing explicit operator preference by means of parenthesis, thus further reducing the usage of composite definitions and simplifying the strategic models.

Acknowledgments.

This research has been supported by the European Research Council (ERC) through advanced grant 267856, titled"Lucretius: Foundations for Software Evolution" (04/201103/2016) and the national project GEODAS-BI (TIN2012-37493-C03-03) from the Spanish Ministry of Economy and Competitiveness (MINECO). Alejandro Maté is funded by the Generalitat Valenciana under an APOSTD grant (APOSTD/2014/064).

[1] D. Barone, L. Jiang, D. Amyot, and J. Mylopoulos. Composite Indicators for Business Intelligence. *Conceptual Modeling–ER 2011*, pages 448–458, 2011.

[2] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[3] J. Cabot, R. Pau, and R. Raventós. From UML/OCL to SBVR specifications: A challenging transformation. *Information Systems*, 35(4):417–440, 2010.

[4] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997.

[5] A. Dardenne, A. Van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1-2):3–50, 1993.

[6] Adela Del-Río-Ortega, Manuel Resinas, and Antonio Ruiz-Cortés. Defining process performance indicators: an ontological approach. In *On the Move to Meaningful Internet Systems: OTM 2010*, pages 555–572. Springer, 2010.

[7] Claudia Diamantini, Laura Genga, Domenico Potena, and Emanuele Storti. Collaborative building of an ontology of key performance indicators. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*, pages 148–165. Springer, 2014.

[8] Claudia Diamantini, Domenico Potena, and Emanuele Storti. Extended drill-down operator: Digging into the structure of performance indicators. *Concurrency and Computation: Practice and Experience*, 2015.

[9] Claudia Diamantini, Domenico Potena, and Emanuele Storti. Sempi: a semantic framework for the collaborative construction and maintenance of a shared dictionary of performance indicators. *Future Generation Computer Systems*, 54:352–365, 2016.

[10] W.W. Eckerson. *Performance dashboards: measuring, monitoring, and managing your business*. Wiley, 2010.

[11] Fiorenzo Franceschini, Maurizio Galetto, and Domenico Maisano. *Management by measurement: Designing key indicators and performance measurement systems*. Springer, 2007.

[12] P. Giorgini, S. Rizzi, and M. Garzetti. Goal-oriented requirement analysis for data warehouse design. *DOLAP'05*, pages 47–56, 2005.

[13] P. Giorgini, S. Rizzi, and M. Garzetti. Grand: A goal-oriented approach to requirement analysis in data warehouses. *Decision Support Systems*, 45(1):4–21, 2008.

[14] T. Hill and R. Westbrook. SWOT analysis: it's time for a product recall. *Long Range Planning*, 30(1):46–52, 1997.

[15] Jennifer Horkoff, Daniele Barone, Lei Jiang, Eric Yu, Daniel Amyot, Alex Borgida, and John Mylopoulos. Strategic business modeling: representation and reasoning. *Software & Systems Modeling*, pages 1–27, 2012.

[16] Julian Hyde. OLAP4J 1.1, Open Java API for OLAP. `http://www.olap4j.org/` (04/09/2014), 2014.

[17] L. Jiang, D. Barone, D. Amyot, and J. Mylopoulos. Strategic Models for Business Intelligence. *Conceptual Modeling–ER 2011*, pages 429–439, 2011.

[18] R.S. Kaplan and D.P. Norton. The balanced scorecard–measures that drive performance. *Harvard business review*, 70(1), 1992.

[19] R.S. Kaplan and D.P. Norton. *Strategy maps: Converting intangible assets into tangible outcomes*. Harvard Business Press, 2004.

[20] S. Luján-Mora, J. Trujillo, and I.Y. Song. A UML profile for multidimensional modeling in data warehouses. *Data & Knowledge Engineering*, 59(3):725–769, 2006.

[21] B. Marr, G. Schiuma, and A. Neely. Intellectual capital–defining key performance indicators for organizational knowledge assets. *Business Process Management Journal*, 10(5):551–569, 2004.

[22] A. Maté and J. Trujillo. A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses. *Information Systems*, 37(8):753 – 766, 2012.

[23] Alejandro Maté, Juan Trujillo, and John Mylopoulos. Conceptualizing and specifying key performance indicators in business strategy models. In *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*, pages 102–115. IBM Corp., 2012.

[24] Alejandro Maté, Kostas Zoumpatianos, Themis Palpanas, Juan Trujillo, John Mylopoulos, and Elvis Koci. A systematic approach for dynamic targeted monitoring of kpis. In *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, pages 192–206. IBM Corp., 2014.

[25] J.N. Mazón, J. Pardillo, and J. Trujillo. A model-driven goal-oriented requirement engineering approach for data warehouses. *Advances in Conceptual Modeling–Foundations and Applications*, pages 255–264, 2007.

[26] J.N. Mazón, J. Trujillo, and J. Lechtenbörger. Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. *Data & Knowledge Engineering*, 63(3):725–751, 2007.

[27] Microsoft. MultiDimensional Expressions Reference. `http://msdn.microsoft.com/en-us/library/ms145506.aspx` (04/09/2014), 2014.

[28] Thomas Neuböck, Bernd Neumayr, Michael Schrefl, and Christoph Schütz. Ontology-driven business intelligence for comparative data analysis. In *Business Intelligence*, pages 77–120. Springer, 2014.

[29] Object Management Group. Business Motivation Model (BMM). `http://www.omg.org/spec/BMM/1.1/`, 2010.

[30] Object Management Group. SBVR 1.3 specification. `http://www.omg.org/spec/SBVR/1.3/`, 2015.

[31] Object Management Group (OMG). Common Warehouse Metamodel. 2003.

[32] J. Pardillo, J-N. Mazón, and J. Trujillo. Extending OCL for OLAP querying on conceptual multidimensional models of data warehouses. *Information Sciences*, 180(5):584 – 601, 2010.

[33] D. Parmenter. *Key performance indicators: developing, implementing, and using winning KPIs*. Wiley, 2009.

[34] Terence Parr. *The definitive ANTLR reference: building domain-specific languages*. Pragmatic Bookshelf, 2007.

[35] Viara Popova and Alexei Sharpanskykh. Modeling organizational performance indicators. *Information Systems*, 35(4):505–527, 2010.

[36] V.E.S. Souza, J-N. Mazón, I. Garrigós, J. Trujillo, and J. Mylopoulos. Monitoring Strategic Goals in Data Warehouses with Awareness Requirements. In *Proceedings of the 2012 ACM Symposium on Applied Computing*. ACM, 2012.

[37] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.

[38] Stefan Strecker, Ulrich Frank, David Heise, and Heiko Kattenstroth. Metricm: a modeling method in support of the reflective design and use of performance measurement systems. *Information Systems and e-Business Management*, 10(2):241–276, 2012.

[39] TPC. TPC-H decision support benchmark. `http://www.tpc.org/tpch/default.asp`, 2016.

[40] Panos Vassiliadis. Modeling multidimensional databases, cubes and cube operations. In *Scientific and Statistical Database Management, 1998. Proceedings. Tenth International Conference on*, pages 53–62. IEEE, 1998.

[41] Jos B Warmer and Anneke G Kleppe. The object constraint language: Precise modeling with uml (addison-wesley object technology series). 1998.

[42] Branimir Wetzstein, Zhilei Ma, and Frank Leymann. Towards measuring key performance indicators of semantic business processes. In *Business Information Systems*, pages 227–238. Springer, 2008.

[43] E. Yu. Modeling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11, 2011.

Appendix A. KPI Specification

In this Appendix we provide a small description of the dataset as well as a comprehensive list of the indicators included in the Steel Wheels model and their derivation.

The Steel Wheels dataset is a sample dataset included with the Pentaho distribution for showing the OLAP, dashboard, and reporting capabilities of the platform. The data is stored in a single database schema, on top of which two multidimensional Mondrian schemata are built in order to provide analysis capabilities for the SteelWheelsSales and Quadrant Analysis facts. The data stored contains information regarding sales of vehicles (SteelWheelsSales) across several territories ranging from year 2003 up to the second quarter of 2005. Therefore, data for year 2005 is only partial, whereas the data for the previous two years is complete. The database also stores information regarding budget allocation and actual costs (Quadrant Analysis). This data however, is not related to the time dimension.

By default, the Mondrian schemata included with Pentaho do not contain any properties. Some of them, however, are necessary to calculate our KPIs and provide interesting capabilities for the KPI specification language. Therefore, we substituted the default Mondrian schemata provided by ones including the corresponding properties, as shown in the paper.

In the following, we present a comprehensive list of all the KPIs defined in Structured English for KPI specification using SBVR notation. Afterwards, we provide the list of generated queries for those KPIs that can be computed, and a table with the real values of the KPIs in our case study.

xg1: **sum** Total benefit **with** year **equal to** 2004 *minus* **sum** Total benefit **with** year **equal to** 2003

xg3: **sum** planned cost **with** Department **equal to** Product Development *minus* **sum** cost **with** Department **equal to** Product Development

xg4: **average** vehicle cost

xg6: **sum** cost **with** Department **equal to** Sales

xg7: **average** Fact Count *of* Region **with** Department **equal to** Sales

xg8: **sum** planned cost **with** Department **equal to** Sales *minus* **sum** cost **with** Department **equal to** Sales

xg10: **sum** amount *of* Vehicles sold **with** year **equal to** 2004

xg11: **maximum** amount *of* Line sold *divided by* **sum** amount **with** amount *of* Line **not equal to** **maximum** amount *of* Line sold

xg12: Fact Count **with** status **equal to** Failed **and** year **equal to** 2004

xg13: **average** score *of* Vehicles

xg14: **sum** Fact Count **with** status **equal to** Cancelled **and** year **equal to** 2004

xg15: defects **with** status **equal to** Shipped **divided by** defects **with** status **equal to** Shipped **or with** status **equal to** Disputed

xs1: **sum** amount **with** year **equal to** 2003 *minus* **sum** amount **with** year **equal to** 2004

xs2: Fact Count **with** status **equal to** Disputed **and** year **equal to** 2004

Among these definitions there are some interesting aspects that we can highlight. For example, Sales is an ambiguous concept because it may refer to the fact attribute Sales or it may refer to the Sales. However, thanks to SBVR constructs in this case we can avoid querying the decision maker. Also, notice that some definitions, such as xg14 and xs2 make use of the concept Fact Count. This is a concept that is intrinsic of the dictionary that allows the user to count the number of rows in the fact table that meet a certain condition. This is useful because, for example, a customer may have ordered several vehicles. Thus, the amount concept does not reflect the correct semantics for this KPI, i.e. the number of orders that the company has received, and cannot be used when analyzing the status of orders. Finally, the list of MDX queries derived is as follows:

```
[xg1:] WITH SET [depth1-0] AS FILTER([Time].[Years].Members,
[Time].[Years].CurrentMember.Name = '2004')
MEMBER [Measures].[result1] AS SUM([depth1-0],[Measures].[Sales])
SET [depth1-1] AS FILTER([Time].[Years].Members,
[Time].[Years].CurrentMember.Name = '2003')
MEMBER [Measures].[result2] AS SUM([depth1-1],[Measures].[Sales])
MEMBER [Measures].[result3] AS [Measures].[result1]-[Measures].[result2],
FORMAT\_STRING = '\#.00'
SELECT [Measures].[result3] ON 0 FROM [SteelWheelsSales]

[xg3:] WITH SET [depth1-0] AS FILTER([Department].[Department].Members,
[Department].[Department].CurrentMember.Name = 'Product Development')
MEMBER [Measures].[result1] AS SUM([depth1-0],[Measures].[Budget])
SET [depth1-1] AS FILTER([Department].[Department].Members,
[Department].[Department].CurrentMember.Name = 'Product Development')
MEMBER [Measures].[result2] AS SUM([depth1-1],[Measures].[Actual])
MEMBER [Measures].[result3] AS [Measures].[result1]-[Measures].[result2],
FORMAT\_STRING = '\#.00'
SELECT [Measures].[result3] ON 0 FROM [QuadrantAnalysis]

[xg4:] WITH MEMBER [Measures].[result1] AS AVG([Product].[Product].Members,
[Product].CurrentMember.Properties("buyprice"))
SELECT [Measures].[result1] ON 0 FROM [SteelWheelsSales]

[xg6:] WITH SET [depth1-0] AS FILTER([Department].[Department].Members,
[Department].[Department].CurrentMember.Name = 'Sales')
MEMBER [Measures].[result1] AS SUM([depth1-0],[Measures].[Actual])
SELECT [Measures].[result1] ON 0 FROM [QuadrantAnalysis]

[xg8:] WITH SET [depth1-0] AS FILTER([Department].[Department].Members,
[Department].[Department].CurrentMember.Name = 'Sales')
MEMBER [Measures].[result1] AS SUM([depth1-0],[Measures].[Budget])
SET [depth1-1] AS FILTER([Department].[Department].Members,
[Department].[Department].CurrentMember.Name = 'Sales')
MEMBER [Measures].[result2] AS SUM([depth1-1],[Measures].[Actual])
MEMBER [Measures].[result3] AS [Measures].[result1]-[Measures].[result2],
FORMAT\_STRING = '\#.00'
SELECT [Measures].[result3] ON 0 FROM [QuadrantAnalysis]

[xg10:] WITH SET [depth1-0] AS FILTER([Time].[Years].Members,
[Time].[Years].CurrentMember.Name = '2004')
MEMBER [Measures].[result1] AS SUM(EXISTS(
```

```
NONEMPTYCROSSJOIN([Product].[Product].Members,[depth1-0]),
[depth1-0]),[Measures].[Quantity])
SELECT [Measures].[result1] ON 0 FROM [SteelWheelsSales]

[xg11:] WITH MEMBER [Measures].[result1] AS MAX([Product].[Line].Members,
[Measures].[Quantity])
MEMBER [Measures].[result2] AS MAX([Product].[Line].Members,
[Measures].[Quantity])
SET [depth1-2] AS FILTER([Product].[Line].Members,[Measures].[Quantity]
<> [Measures].[result2])
MEMBER [Measures].[result3] AS SUM([depth1-2],[Measures].[Quantity])
MEMBER [Measures].[result4] AS [Measures].[result1]/[Measures].[result3],
FORMAT\_STRING = '\#.00'
SELECT [Measures].[result4] ON 0 FROM [SteelWheelsSales]

[xg14:] WITH SET [depth1-0] AS FILTER(NONEMPTYCROSSJOIN([OrderStatus].[Type].Members,
[Time].[Years].Members),[OrderStatus].[Type].CurrentMember.Name = 'Cancelled'
AND [Time].[Years].CurrentMember.Name = '2004')
MEMBER [Measures].[result1] AS SUM([depth1-0],[Measures].[Fact Count])
SELECT [Measures].[result1] ON 0 FROM [SteelWheelsSales]

[xs1:] WITH SET [depth1-0] AS FILTER([Time].[Years].Members,
[Time].[Years].CurrentMember.Name = '2003')
MEMBER [Measures].[result1] AS SUM([depth1-0],[Measures].[Quantity])
SET [depth1-1] AS FILTER([Time].[Years].Members,
[Time].[Years].CurrentMember.Name = '2004')
MEMBER [Measures].[result2] AS SUM([depth1-1],[Measures].[Quantity])
MEMBER [Measures].[result3] AS [Measures].[result1]-[Measures].[result2],
FORMAT\_STRING = '\#.00'
SELECT [Measures].[result3] ON 0 FROM [SteelWheelsSales]

[xs2:] WITH SET [depth1-0] AS FILTER(NONEMPTYCROSSJOIN(
[OrderStatus].[Type].Members,[Time].[Years].Members),
[OrderStatus].[Type].CurrentMember.Name = 'Disputed'
AND [Time].[Years].CurrentMember.Name = '2004')
SELECT [Measures].[Fact Count] ON 0,
[depth1-0] ON 1 FROM [SteelWheelsSales]
```

Table A2: Values for each KPI according to Steel Wheels data

Indicator	Worst	Threshold	Current	Target	Status
xg1	0	367,738 (+10 %)	1,310,355	725,476 (+20 %)	Green
xg3	-539,330 (-5 %)	269,665 (2,5 %)	142,509	539,330 (5 %)	Red
xg4	57.46 (5 %)	54.73 (0 %)	54.4	53.08 (-3 %)	Yellow
xg6	12,070,31 (+10 %)	11,522,061 (+5 %)	11,168,773	10,973,392 (0 %)	Yellow
xg7	- no data available -				
xg8	-1,097,338 (-10 %)	-548,669 (-5 %)	-195,381	0	Yellow
xg10	32,795 (-10 %)	36,439	49,417	40,082 (+10 %)	Green
xg11	$< 0{,}15$ or $> 0{,}7$	$< 0{,}25$ or $> 0{,}6$	0,51	$> 0{,}3$ and $< 0{,}45$	Yellow
xg12	- no data available -				
xg13	- no data available -				
xg14	142 (10 %)	71 (5 %)	54	28 (2 %)	Yellow
xg15	- no data available -				
xs1	0	3,643 (10 %)	-12,978	7,286 (20 %)	Red
xs2	142 (10 %)	71 (5 %)	1	28 (10 %)	Green