International Journal of Cooperative Information Systems
Vol. 25, No. 2 (2016) 1650006 (46 pages)
© The Author(s)
DOI: 10.1142/S0218843016500064

Reference Modeling for Data Analysis: The BIRD Approach

Christoph G. Schuetz*, Bernd Neumayr† and Michael Schrefl‡

Department of Business Informatics —
Data and Knowledge Engineering
Johannes Kepler University Linz
Altenberger Straße 69, 4040 Linz, Austria
**schuetz@dke.uni-linz.ac.at*
†neumayr@dke.uni-linz.ac.at
‡schrefl@dke.uni-linz.ac.at

Thomas Neuböck

Solvistas GmbH
Graben 18, 4020 Linz, Austria
thomas.neuboeck@solvistas.com

Received 13 July 2015
Revised 21 April 2016
Accepted 16 August 2016
Published 22 September 2016

Reference models for data analysis with data warehouses may consist of multidimensional reference models and analysis graphs. Multidimensional reference models are best-practice domain-specific data models for online analytical processing. Analysis graphs are reference models of analysis processes for event-driven data analysis. Small and medium-sized enterprises (SMEs) as well as large multinational companies may benefit from the use of reference models for data analysis. The availability of multidimensional reference models lowers the obstacles that inhibit SMEs from using business intelligence (BI) technology. Multinational companies may define multidimensional reference models for increased compliance among subsidiaries and departments. Furthermore, the definition of analysis graphs facilitates the handling of business events for both SMEs and large companies. Modelers may customize the chosen reference models, tailoring the models to the specific needs of the individual company or local subsidiary. Customizations may consist of additions, omissions, and modifications with respect to the reference model. In this paper, we propose a metamodel and customization approach for multidimensional reference models and analysis graphs. We specifically address the explicit modeling of key performance indicators as well as the definition of analysis situations and analysis graphs.

Keywords: Conceptual modeling; business intelligence; online analytical processing; dimensional fact model; metadata management; data warehouses.

*Corresponding author.

1. Introduction

Reference models may describe both structural and behavioral aspects of data analysis. Structural aspects of data analysis comprise fact classes and dimensions of a data warehouse, key performance indicators (KPIs), and business terms of interest for a particular domain. Multidimensional reference models represent common, best-practice instances of data analysis structures which may be customized to the specific needs of a particular use case. Behavioral aspects of data analysis, on the other hand, comprise the analysis processes followed by analysts in order to solve a particular analytical task. An analysis graph defines a common, best-practice course of action to solve an analytical task, modeling the succession of individual analysis situations. An analysis situation represents a specific view on the data that analysts employ to solve the analytical task at hand.

Many small and medium-sized enterprises (SMEs) refrain from adopting business intelligence (BI) technology but reference models allow these companies to overcome the obstacles associated with the introduction of BI solutions. The factors for nonproliferation of BI technology among SMEs include the nonexistence of a coherent definition of KPIs within the company, the mismatch between business needs and BI functionalities, the perceived complexity of handling and report building, and unqualified personnel as well as poorly structured data.[1] Other factors are a lack of BI-related know-how and the costs for implementation and deployment of a BI solution.[2] By providing a starting point for BI projects, reference models have the potential to reduce implementation and deployment costs for BI solutions and serve as a basis for requirements analysis together with business analysts.[3,4] Service providers may offer a set of preconfigured multidimensional reference models, including a catalog of KPIs, for different industries. These reference models formalize best-practice multidimensional models[5] which can be tailored to the specific needs of a company.[3–5] Furthermore, the use of reference models facilitates inter-company data analysis, thus increasing the potential benefits from BI for SMEs which individually might not generate sufficient amounts of data for insightful data analysis. Analysis graphs, on the other hand, externalize tacit knowledge of analysts about best-practice analysis processes. The explicit representation and documentation of such knowledge facilitates data exploration for unexperienced personnel.

Multinational companies also benefit from the employment of reference models for data analysis. Multidimensional reference models facilitate the integration of reporting systems of acquired branches into the company-wide data warehouse infrastructure. Multidimensional reference models also support the enforcement of corporate policies and legal requirements[6] across different subsidiaries and departments as well as the establishment of a shared understanding of KPIs and business terms.[7,8] Analysis graphs allow for a uniform handling of business situations that require specific information needs.

Following the guidelines for design-science research,[9] we present BIRD — Business Intelligence Reference Modeling for Data Analysis — a lightweight reference modeling approach for relational OLAP (ROLAP) that covers not only multidimensional reference models but also analysis graphs; the individual components of BIRD reference models are customizable. The outcome of this paper is a design artifact which proposes a solution to the resource-intensive, time-consuming task of implementing BI solutions with high demands for specific know-how. For multidimensional reference modeling, the BIRD approach adapts the dimensional fact model[10] (DFM) — a popular approach for conceptual data warehouse modeling; SQL serves as definition language for KPIs and business terms. The reliance on well-known methodologies and ubiquitous technology, such as DFM and SQL, has several advantages. In order to avoid a mismatch between business needs and available data, business analysts should be involved in the customization process.[5] Preferably, multidimensional reference models are simple, understandable, and thus easily customized by the average business analyst since the necessary customization effort determines the benefit of a reference model.[4] Both the DFM and SQL should be easy to understand for business analysts who are spared from learning a complex reference modeling language. The use of SQL and star schema also potentially facilitates integration into existing infrastructure since many companies employ relational databases.

The contribution of BIRD is not another approach to conceptual multidimensional modeling but rather represents a methodology for the definition of reference models for data analysis and their customization as well as the generation of logical schemas and queries. BIRD specifically focuses on the explicit representation of calculated measures and business concepts, which is otherwise often down to the intuition of the business analyst and thus elusive to unexperienced business analysts. BIRD also focuses on the representation of analysis processes, which assist unexperienced business analysts with satisfying their analysis needs. As opposed to mere multidimensional modeling approaches, BIRD also considers the possibility of customizations. The BIRD metamodel for multidimensional reference models condenses and simplifies existing conceptual modeling approaches for the purposes of reference modeling, thereby allowing business analysts who are nonexperts in BI technology to take part in the customization process. We refer to existing works for general issues in multidimensional modeling, for example, the study of summarizability[11,12] and the definition of general consistency criteria. In this regard, Romero and Abelló[13] offer a comprehensive overview on multidimensional modeling approaches.

We illustrate the BIRD approach with a use case inspired by one of our industry partners, KOTI Kobra, the Austrian subsidiary of the KOTI group. The KOTI group[a] is a European manufacturer of various sorts of brushes such as industrial and technical brushes, strip and sealing brushes, work tool brushes, sweeping and cleaning brushes, runway brushes for airports, hygiene brushes, as well as entrance brush

[a]https://www.koti-eu.com/en/home.

mats. Different departments have similar requirements regarding data analysis but due to differences in the manufacturing processes data models require customization. The evaluation section (Sec. 4), in addition, discusses how the BIRD approach can be employed in the agriProKnow project, an experimental development collaboration between industry and academia for data analysis in precision dairy farming. Proof-of-concept prototypes demonstrate feasibility of the BIRD approach.

This paper is a revised and extended version of a previous conference paper.[14] Whereas the original conference paper focused on structural reference modeling, this paper also concentrates on the behavioral aspects of reference modeling for data analysis by adding analysis graphs. Analysis graphs are the result of the Semantic Cockpit project[7] (semCockpit) from a previous industry collaboration. With respect to previous works on analysis graphs[7,15,16] we also consider the interdependencies of analysis graphs and multidimensional reference models, the customization of analysis graphs as well as the generation of parameterized queries for analysis situations and workflow models based on State Chart XML[17] (SCXML). We choose SCXML due to its simplicity and extensibility, allowing for the machine-readable representation of workflow models for analysis graphs, self-contained along with the analysis situation and auxiliary queries. We stress, though, that the concept of analysis graphs is independent of any particular representation format.

The remainder of this paper is organized as follows. In Sec. 2 we present the approach for the representation of multidimensional reference models and their customization as well as the generation of corresponding star schema tables. In Sec. 3 we present the approach for the modeling of analysis situations and analysis graphs based on multidimensional reference models as well as the generation of parameterized queries and workflow models. In Sec. 4 we evaluate the BIRD approach. In Sec. 5 we review related works and discuss the relationship between BIRD and related works. In Sec. 6 we conclude the paper.

2. Multidimensional Models

In this section, we discuss multidimensional reference modeling, the customization of multidimensional reference models, and the derivation of star schema tables for specific customizations of reference models.

2.1. *Reference modeling*

We adapt existing multidimensional modeling approaches in data warehousing for the needs of reference modeling. A multidimensional reference model consists of fact classes and dimensions, calculated measures, and predicates for the definition of business terms. Reference models must be customizable to the specific needs of a particular business and ideally allow automatic generation of executable code which BIRD achieves through injection of SQL fragments in reference models.

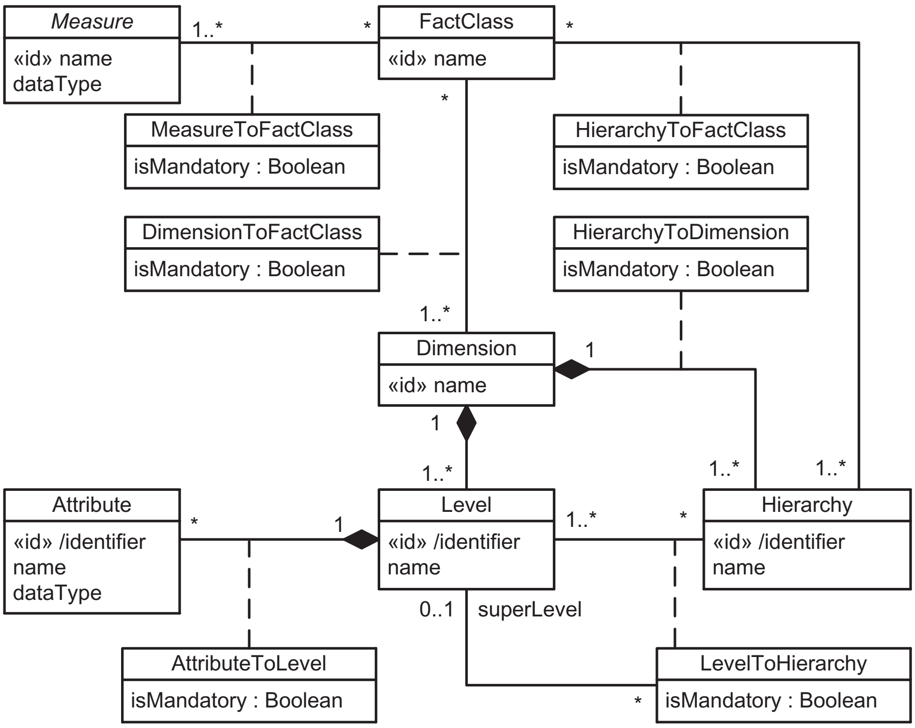

Fig. 1. The metamodel of multidimensional reference models as UML class diagram.

2.1.1. *Dimensions and facts*

We derive the metamodel for multidimensional reference models mainly from the dimensional fact model,[10] although the metamodel elements can be considered a common denominator of existing multidimensional models.[13] Figure 1 defines as UML class diagram the elements of a fact schema in a reference model. The central element of a multidimensional reference model is the fact class (class FactClass). A fact class refers to one or more dimensions (Dimension) each consisting of several levels (Level); attributes (Attribute) further describe a level. An attribute belongs to a single level and, thus, to a single dimension. Attribute names are unique within a dimension. Each level has a name which is unique within the dimension. Levels are organized in hierarchies (Hierarchy). Within a hierarchy, a level has either exactly one superlevel or is the hierarchy's top level. Parallel aggregation paths within a dimension are realized as different hierarchies. Fact classes may also employ only a subset of the hierarchies that the dimensions of the fact class define. To this end, the association between FactClass and Hierarchy explicitly determines the employment of hierarchies by a fact class. A fact class must, however, employ at least one hierarchy of each dimension of the fact class. At least one measure (Measure) quantifies the facts that the fact class represents.

A consistent multidimensional reference model must satisfy the integrity constraints commonly associated with multidimensional models. In particular, level hierarchies must be acyclic and level hierarchies of the same dimension must not contradict each other regarding level order. All levels in a hierarchy must belong to the same dimension that the hierarchy belongs to, a level must only have super levels from the same hierarchy. A fact class may only have hierarchies which belong to one of the dimensions of the fact class. These integrity constraints are not distinctive to reference modeling but correspond to those constraints proposed by academia for simple hierarchies[18] or implemented by database vendors.[19]

Nonstrict hierarchies, represented as multiple arcs in the DFM,[20] may be incorporated into the BIRD metamodel, thus allowing many-to-many relationships between levels or between the fact class and a base level. Then, the LevelToHierarchy association class would have a hasMultipleArc attribute that indicates whether the outgoing superLevel attribute represents a multiple arc. Similarly, the Hierarchy-ToFactClass association class would have a hasMultipleArc attribute that indicates whether the fact class references multiple instances of the respective hierarchy's base level. In order to avoid introducing additional complexity, however, we try to model separate dimensions[18] in our industry projects (see Sec. 4.2) instead of multiple arcs, which generally facilitates analysis for the user at the cost of redundancy.

Example 2.1 (Fact classes dimensions). Figure 2 illustrates, using a DFM-based notation, fact classes and dimensions of an example multidimensional reference model for manufacturing companies; the running example adapts and extends an example from previous work.[16] Boxes denote fact classes, circles denote levels, and arrows between circles denote roll-up relationships. The first compartment of a box contains the name of the fact class, the second compartment contains measures. Due to space considerations the graphical representation in Fig. 2 does not include hierarchy names. Dashed boxes attached to levels represent dimension predicates (see Sec. 2.1.3). The mandatory stereotype attached to an arrow from a fact class to a level denotes a mandatory dimension for the particular fact class. The mandatory stereotype next to a level name denotes a mandatory level for a particular hierarchy. Note, however, that the DFM-based notation is for illustration purposes only. The MaterialUsedForProduct fact class has mandatory Time, Product, and Material dimensions as well as optional Customer and Factory dimensions. The Time dimension consists of two hierarchies, the first with mandatory day level, optional week level, and mandatory year level, the second with mandatory day, month, and year levels as well as an optional quarter level. The Product dimension consists of a single hierarchy with mandatory productOrder level and optional productCategory level. The productOrder level has minTemperature, maxTemperature, and minLifeTime attributes. The Customer dimension consists of three hierarchies, the first with customer and consumerGroup levels, the second with customer and industry levels, and the third with customer and customerRegion levels. The Factory dimension consists of a single hierarchy with building, site, and country levels.

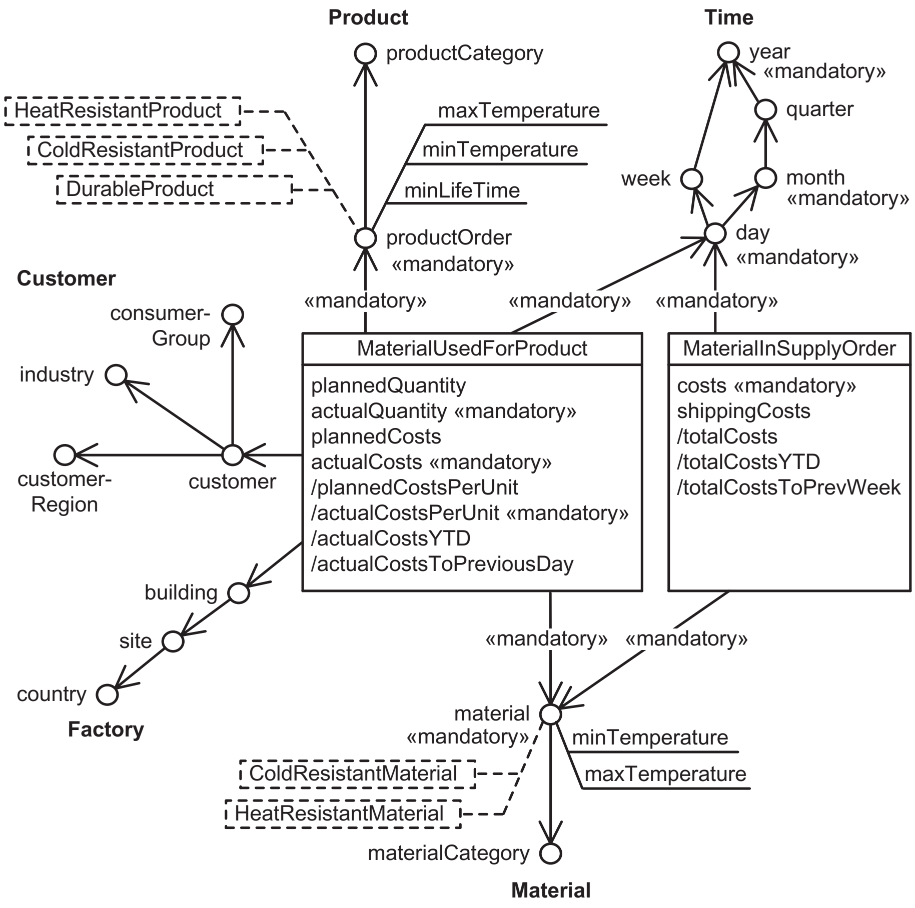

Fig. 2. An example multidimensional reference model for manufacturing companies.

The Material dimension consists of a single hierarchy with mandatory material and optional materialCategory levels. The material level has minTemperature and maxTemperature attributes. The MaterialInSupplyOrder fact class has mandatory Time and Material dimensions.

As opposed to elements of ordinary multidimensional models, reference model elements may be flagged mandatory, rendering illegal the deselection of the thus flagged elements during customization (see Sec. 2.2). The isMandatory attribute of association classes in the metamodel (Fig. 1) indicates classes of model elements possibly flagged mandatory. Measures, dimensions, and hierarchies of a fact class may be flagged mandatory, meaning that a link in the reference model between a particular measure, dimension, or hierarchy to a fact class must be present in the finally generated, derived model as well. Furthermore, hierarchies of a dimension, levels of a hierarchy, and attributes of a level may be marked mandatory, too. The possibility of declaring certain model elements mandatory allows for the definition of least common semantics for these elements, a requirement for ensuring the capability of reference models to enforce intercompany and interdepartmental compliance with regulations while allowing for the customization of reference models.

Reference model elements are named. For FactClass, Measure, and Dimension the name attribute is identifier. Instances of these classes are independent entities. The Hierarchy, Level, and Attribute classes have a derived identifier. Instances of these classes exist only as a component of other entities. The identifiers of hierarchies, levels, and attributes consist of the name of the dimension and the name of the respective model element, separated by a dot. The sets of level names and attribute names of a dimension must be disjoint, owing to the translation of dimensions in the multidimensional reference model into dimension tables in the generated logical model, with each level and attribute of a dimension becoming a column in the respective dimension table.

Measures quantify facts and are the actual focus of interest in a data warehouse. A measure (class Measure in Fig. 1) may belong to multiple fact classes; a measure has a unique name and a data type. In its most basic form a measure is just a value obtained from an operational data source during the extract, transform, and load (ETL) process. Such a measure is referred to as base measure, as opposed to a calculated measure which derives from other measures (see Sec. 2.1.2).

Example 2.2 (Base measures). The MaterialUsedForProduct fact class (Fig. 2) has base measures plannedQuantity, actualQuantity, plannedCosts, and actualCosts. Measures actualQuantity and actualCosts are mandatory base measures of MaterialUsedForProduct. The MaterialInSupplyOrder fact class has base measures costs and shippingCosts, with costs being mandatory. The graphical representation in Fig. 2 does not show data types. The measures are numbers.

An additivity matrix[21] may indicate sensible aggregation of measures along the individual dimensions' level hierarchies. The additivity matrix allows for the detection of logically invalid queries with summarizability problems, a feature that could be incorporated in a CASE tool for the modeling of analysis situations (see Sec. 3.1.1). An additivity matrix could be incorporated in the BIRD metamodel, indicating the aggregation operators that can be applied to the measure of a fact class. We do not specifically address summarizability problems but refer to related works[11,12] for the study of summarizability.

2.1.2. *Calculated measures*

Figure 3 defines the different kinds of measures available for reference modeling. A measure is either base (class BaseMeasure) or calculated (CalculatedMeasure), and never both. A base measure is an asserted value that is directly obtained from the operational databases during the ETL process, possibly after cleansing. A calculated measure, on the other hand, is ultimately derived from base measures. The

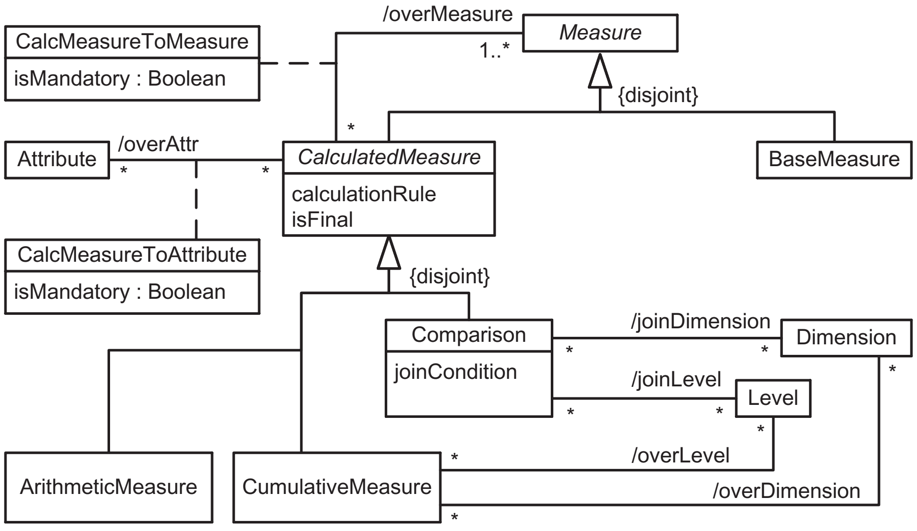

Fig. 3. Kinds of measures in the metamodel of multidimensional reference models.

most basic form of calculated measure is arithmetic (ArithmeticMeasure) which combines base measures using arithmetic operations such as addition/subtraction and multiplication/division. Cumulative measures (CumulativeMeasure) and comparative measures (Comparison) are special cases of calculated measures. Calculated measures may also be marked final, preventing the redefinition of the calculation rule during customization (see Sec. 2.2.2).

A calculated measure has a calculation rule (attribute calculationRule) defined as an SQL expression. Individual components of a calculation rule may be flagged mandatory, allowing for the definition of common semantics across different customizations, which may redefine calculated measures (see Sec. 2.2.2) The calculation rule refers to measures by their names in conjunction with the Fact qualifier as generic alias for the respective fact table in the ultimately generated star schema.

Example 2.3 (Arithmetic measures). The plannedCostsPerUnit and actualCostsPerUnit measures as employed by the MaterialUsedForProduct fact class in Fig. 2 are arithmetic. The SQL expression `Fact.plannedCosts/Fact.plannedQuantity` is the calculation rule for the plannedCostsPerUnit measure. The SQL expression `Fact.actualCosts/Fact.actualQuantity` is the calculation rule for the actualCostsPerUnit measure. Furthermore, the totalCosts measure as employed by the MaterialInSupplyOrder fact class is arithmetic. The SQL expression `Fact.costs + Fact.shippingCosts` is the calculation rule for the totalCosts measure.

The Fact qualifier in the calculation rule of a calculated measure refers to the measure's fact class which may change depending on the context. Since a measure may feature in multiple fact classes, at query time the Fact qualifier refers to different fact tables, depending on the context the measure value is actually calculated in. The reasons for the importance of the Fact qualifier in the definitions of calculation rules are twofold. First, SQL expressions are intended to be used directly for the generation of SQL queries (see Sec. 2.3). Second, besides referring to other measures, the calculation rule of a calculated measure may also contain references to attributes of levels. It is thus necessary to distinguish measures from attributes through the use of different qualifiers since the names of these model elements may overlap. In order to reference an attribute, the name of the attribute's dimension is used as a qualifier, which is possible since the attribute names are unique within a dimension. In other words, attributes in calculation rules are referred to by attribute identifier. Note that a measure that is based on an attribute may only be used in fact classes that actually use the hierarchy of this attribute's level.

Example 2.4 (Arithmetic measure based on attribute). The SQL expression `Fact.plannedCosts/Product.minLifetime` defines planned costs of material per year of the final product's (minimum) lifetime based on plannedCosts and the productOrder level's minLifetime attribute.

A link between a calculated measure and a fact class signifies, in general, that the calculated measure is available for facts at the base granularity of the fact class. In this sense calculated measures are not unlike base measures.

Cumulative measures represent aggregated values that are not obtained through roll-up; analysis situations define aggregation through roll-up (see Sec. 3.1.1). A cumulative measure partitions facts into groups. Within these groups the facts are ordered and their measure values accumulated. Year-to-date measures are typical examples of cumulative measures, for example, a company's revenue within a year accumulated until a particular month. In SQL the `PARTITION BY` clause allows for the definition of cumulative measures, providing `GROUP BY` functionality without actually altering the granularity of the table.

Example 2.5 (Cumulative measure). The actualCostsYTD measure as employed by the MaterialUsedForProduct fact class in Fig. 2 is cumulative. The SQL expression `SUM(Fact.actualCosts) OVER (PARTITION BY Time.year, Fact.Product, Fact.Customer, Fact.Factory, Fact.Material ORDER BY Time.day)` defines the year-to-date actual costs of material used in manufacturing of products by day. Furthermore, the totalCostsYTD measure as employed by the MaterialInSupplyOrder fact class is cumulative. The SQL expression `SUM(Fact.costs + Fact.shippingCosts) OVER (PARTITION BY Time.year, Fact.Material ORDER BY Time.day)` defines the year-to-date total costs of material in supply orders by day. The actualCostsYTD and totalCostsYTD measures are not aggregable.

Even though cumulative measures feature an aggregation operation, this aggregation does not correspond to the roll-up of facts. A cumulative measure that is associated with a fact class is available for facts at some granularity level defined by the attributes in the calculation expression's `OVER` clause. Commonly, cumulative measures cannot be further aggregated in a sensible way.

The third class of calculated measures are comparisons or comparative measures. Measure values may be compared with measure values from a different time period, geographic location, etc., of comparison. A comparative measure corresponds to a join of a cube with itself in order to compute ratios between measure values of individual facts and the respective facts of comparison. A comparative measure also has a join condition expressed in SQL over dimensions (association end joinDimension to Dimension from Comparison) and levels (association end joinLevel to Level). A comparative measure may only be associated with a fact class that references the join dimensions and the hierarchies that contain the join levels. In the calculation rule and join condition of comparative measures, the ComparisonFact qualifier refers to the fact of comparison that is obtained when applying the comparative measure's join condition. The ComparisonFact qualifier, just like the Fact qualifier, is context-dependent. Additional context-dependent qualifiers, for example, ComparisonTime, ComparisonMaterial, and ComparisonSupplier, refer to the dimensions of the fact of comparison.

Example 2.6 (Comparison). The actualCostsToPreviousDay measure as employed by the MaterialUsedForProduct fact class in Fig. 2 is a comparison. The SQL expression `Fact.actualCosts/ComparisonFact.actualCosts` is the calculation rule for the actualCostsToPreviousDay measure. The SQL expression `Time.day = DATEADD(day, 1, ComparisonTime.day) AND Fact.Product = ComparisonFact. Product AND Fact.Customer = ComparisonFact.Customer AND Fact.Factory = ComparisonFact.Factory AND Fact.Material = Comparison-Fact.Material` is the join condition of the actualCostsToPreviousDay measure. Furthermore, the totalCostsToPrevWeek measure as employed by the MaterialIn `SupplyOrder` fact class is a comparison. The SQL expression `(Fact.costs + Fact. shippingCosts)/(ComparisonFact.costs + ComparisonFact.shipping Costs)` is the calculation rule for the totalCostsToPrevWeek measure. The SQL expression `Time.day = DATEADD(week, 1, ComparisonTime.day) AND Fact.Material = ComparisonFact.Material` is the join condition of the totalCostsToPrevWeek measure. Note that the `DATEADD` function is specific to Microsoft SQL Server but other database vendors provide similar functions; dialect-specific expressions may be incorporated into BIRD reference models.

In the metamodel for calculated measures (Fig. 3), a derived association represents the referencing of measures and attributes in calculation rules of calculated measures. The overMeasure and overAttribute association ends may be automatically derived from the SQL expression that defines the calculation rule. Similarly, the overDimension and overLevel association ends from CumulativeMeasure to Dimension and Level, respectively, may be automatically derived from the calculation rule. The joinDimension and joinLevel association ends from Comparison to Dimension and

Level, respectively, may be automatically derived from the SQL expression that defines the join condition of the comparative measure.

A multidimensional reference model contains a catalog of calculated measures where each measure may be referenced by a number of fact classes. A calculated measure represents a KPI and the catalog of calculated measures provides uniform definitions and promotes a shared understanding of KPIs. Standard catalogs of KPIs from industry associations[22] and academia[23,24] may serve as a starting point for the development of reference models for a particular industry which may then be customized for individual companies.

2.1.3. *Predicates*

Predicates formalize and unambiguously define business terms. Rather than leaving the definition of business terms to the business analyst, who usually defines these terms in an *ad hoc* manner during the analysis, the multidimensional reference model may provide a shared conceptualization of business terms. In this sense, predicates are similar to concepts of a multidimensional ontology.[7]

A dimension predicate (class DimensionPredicate in Fig. 4) is defined by a Boolean SQL expression over a single dimension. A dimension predicate, however, may refer to several attributes of the same dimension. A dimension predicate may also refer to the implicit name attribute of levels, for example, use Product.productCategory to express a condition over the name of the product category that the predicate applies to. A multidimensional predicate (class MultidimensionalPredicate, not shown in figures) is defined by an SQL expression over multiple dimensions. A measure predicate (MeasurePredicate, not shown in figures), in addition to attributes, allows for the definition of Boolean SQL expressions over measures. Referenced attributes and measures in predicate expressions may be flagged mandatory, similarly to measures and attributes in calculation rules of calculated measures. Predicates may also be marked final for preventing redefinitions during customization.

Example 2.7 (Dimension predicates). The example multidimensional reference model in Fig. 2 contains the HeatResistantProduct, ColdResistantProduct, and DurableProduct predicates for the Product dimension as well as the Heat

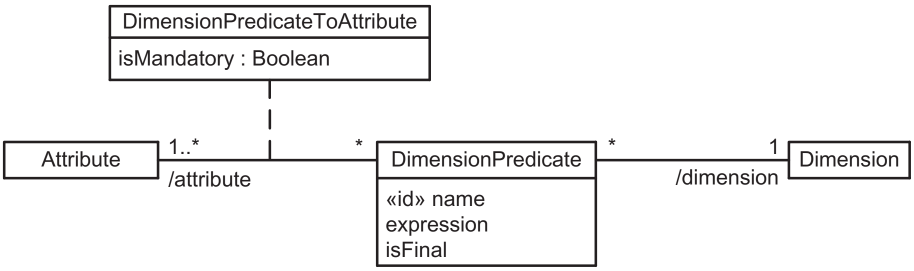

Fig. 4. Dimension predicates in the metamodel of multidimensional reference models. Multidimensional and measure predicates are defined analogously.

ResistantMaterial and ColdResistantMaterial predicates for the Material dimension. The SQL expression `Product.maxTemperature > 50` defines HeatResistantProduct, `Product.minTemperature < −10` defines ColdResistantProduct, and `Product.minLifeTime > 5` defines DurableProduct. The SQL expressions for HeatResistantMaterial and ColdResistantMaterial are similarly defined. A predicate HeatResistantBristleMaterial may define the heat-resistance property for a particular material category as `Material.maxTemperature > 60 AND Material.materialCategory = 'Bristle Material'`.

Example 2.8 (Multidimensional predicate). Heat-resistant materials in durable products may be defined by the SQL expression `Material.maxTemperature > 60 AND Product.minLifeTime > 5`.

Example 2.9 (Measure predicate). Expensive heat-resistant materials in durable products may be defined by the SQL expression `Material.maxTemperature > 60 AND Product.minLifeTime > 5 AND (Fact.actualCosts/Fact.actualQuantity) > 100`.

For better manageability, predicates could be organized in subsumption hierarchies. Previous work[7] has employed semantic technologies and automated reasoners for the organization of (multi)dimensional concepts into subsumption hierarchies. Future work could adapt this approach for multidimensional reference modeling with SQL-defined predicates, thereby eliminating the need for a separate multidimensional ontology language and instead relying on a standard query language.

2.2. *Customization*

An important characteristic of reference models is the possibility of customization for specific requirements imposed by a particular IT infrastructure and business environment. BIRD provides facilities for deselecting and adding model elements as well as for the redefinition of calculated measures and business terms.

2.2.1. *Dimensions and facts*

Customizations of a multidimensional reference model's fact classes and dimensions consist of additions and omissions. Figure 5 illustrates the elements of customizations of fact classes and their dimensions. A customization (class FactClassCustomization) applies to a single fact class. A customization may deselect measures (association end deselectedMeasure to Measure), dimensions (deselectedDimension to Dimension), and hierarchies (deselectedHierarchy to Hierarchy) that are associated with the customization's fact class. A customization may also deselect individual levels (deselectedLevel to Level) — as well as attributes (deselectedAttribute to Attribute) of these levels — of the dimensions of the fact class. Besides omissions, a customization may also add measures (addedMeasure), dimensions (addedDimension), and hierarchies (addedHierarchy) to a fact class.

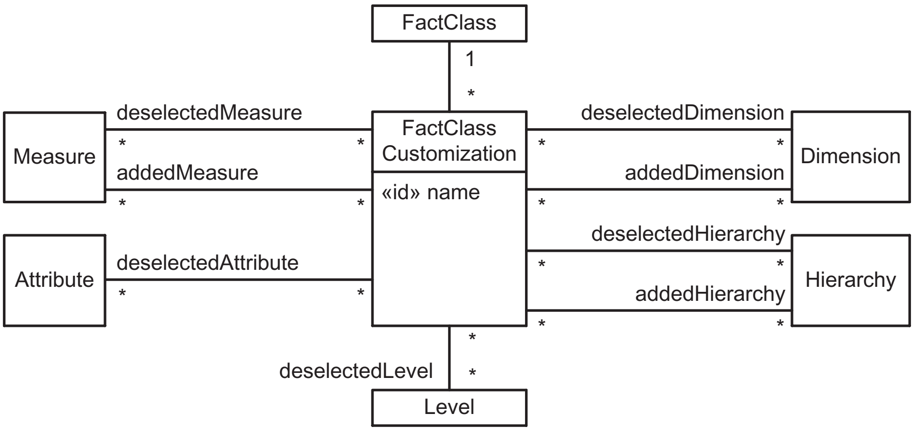

Fig. 5. The definition of customizations in the metamodel of multidimensional reference models.

When adding measures, dimensions, and hierarchies to a fact class, a customization may add existing model elements that previously had not been associated with the customization's fact class. On the one hand, a customization may add suitable elements from other fact classes. On the other hand, the multidimensional reference model may contain a multitude of measures, dimensions, and hierarchies that are not associated with any fact class, thereby providing additional options for customization. When adding a hierarchy of a dimension that is not already associated with the fact class, the customization must also add the hierarchy's dimension. For simplicity, and in order to avoid conflicts with contradicting level hierarchies, a customization cannot add individual levels. A new level could be falsely added between two levels while it should actually be higher up or lower down the hierarchy, for example, when adding region above country rather than below country and above site. The addition of another dimension is usually orthogonal and thus less error prone. Similarly, a customization cannot introduce additional attributes.

Example 2.10 (Customization of fact classes and dimensions). Figure 6 illustrates a customization of the MaterialUsedForProduct and MaterialInSupplyOrder fact classes as well as the associated dimensions from the multidimensional reference model in Fig. 2. In the Time dimension the customizations of the fact classes deselect the quarter level. In the Product dimension the customizations deselect the productCategory level. The Material dimension remains unchanged. In the Customer dimension the customization of the MaterialUsedForProduct fact class deselects the consumerGroup level and the corresponding hierarchy. The Factory dimension is deselected altogether. The customization of the MaterialInSupplyOrder fact class adds the Supplier dimension as well as the measures orderedQuantity, deliveredQuantity, and totalCostsPerUnit while deselecting the shippingCosts measure.

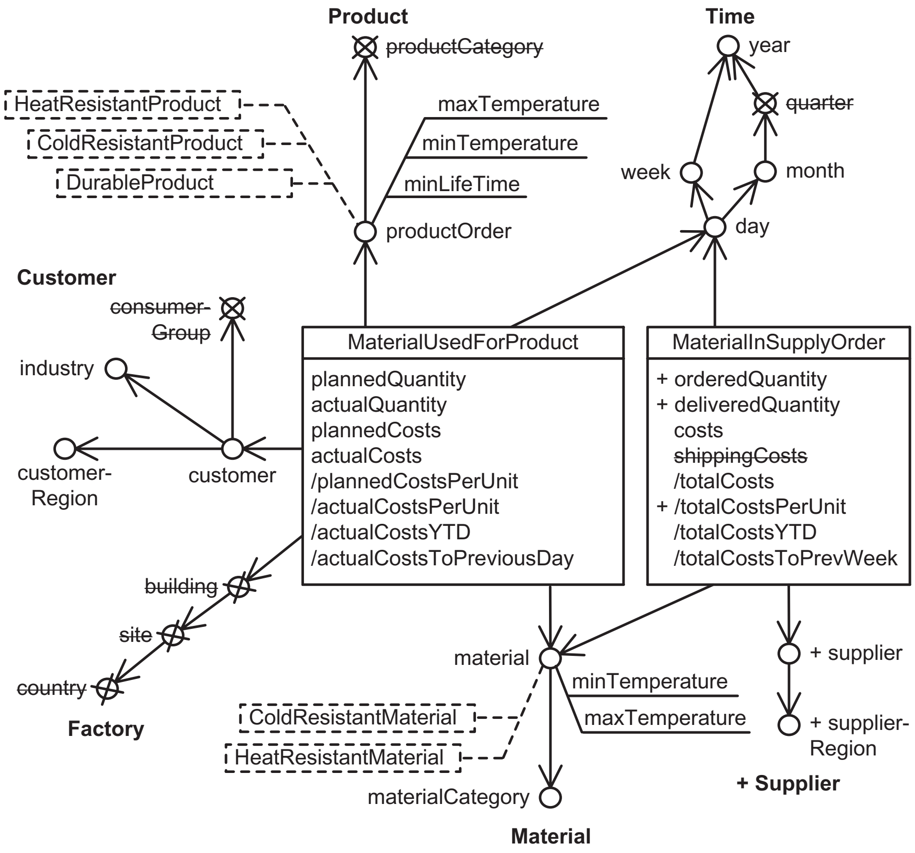

Fig. 6. An example customization of the multidimensional reference model in Fig. 2.

A customization may only deselect nonmandatory elements. The levels of a mandatory dimension do not inherit the mandatory feature and thus, by default, a customization may deselect levels of mandatory dimensions if not explicitly stated otherwise. At least one level of a mandatory dimension must remain in the customization. The levels of a mandatory hierarchy cannot be deselected by a customization. In this sense, the isMandatory property of hierarchies is much stronger than the isMandatory property of dimensions. This interpretation makes sense since a dimension is a rather generic construct. The association of a fact class with a dimension defines a certain dimensionality. A hierarchy, however, is a much more specialized construct, its main trait being the definition of level order.

In some cases, the isMandatory property of one model element prevents the deselection of other, *a priori* nonmandatory, model elements. A mandatory hierarchy prevents an otherwise nonmandatory dimension from being deselected. The isMandatory property of an individual level, being an attribute of the association between a level and its hierarchy, should not constrain the deselection of dimensions

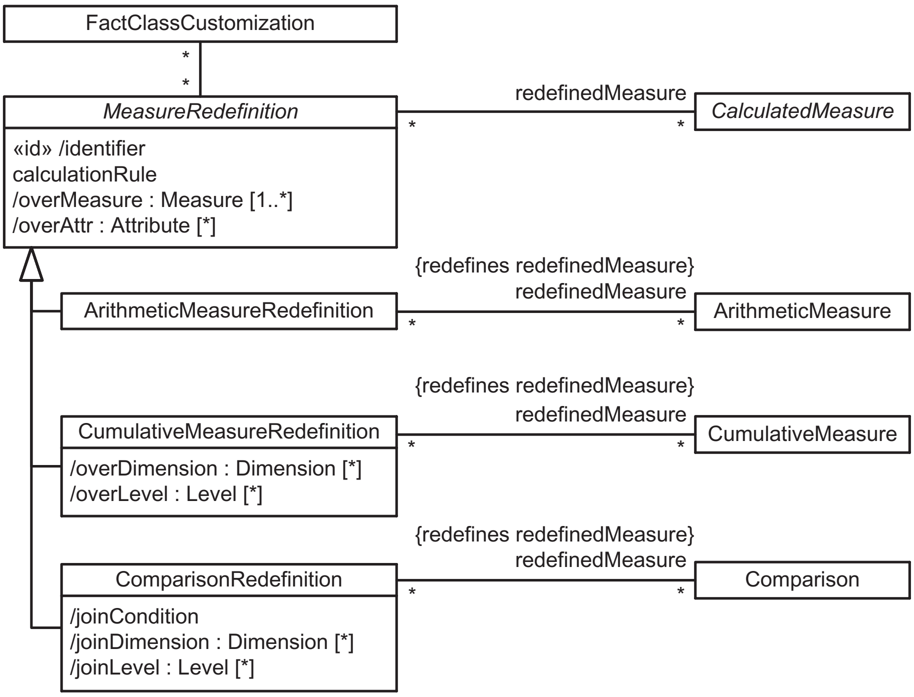

Fig. 7. Measure redefinitions in the metamodel of multidimensional reference models.

and hierarchies. A mandatory level is mandatory only within the hierarchy and deselection of the hierarchy or dimension results in the level being not mandatory for the fact class anymore. Similarly, the isMandatory property of an attribute does not constrain the deselection of levels.

2.2.2. *Calculated measures*

Customizations of multidimensional reference models may provide redefinitions of calculated measures. Figure 7 defines the metamodel for measure redefinitions. A measure redefinition (class MeasureRedefinition) overrides, in the context of customizations, the original calculation rule of the redefined measure. A redefinition of a comparison (ComparisonRedefinition) also overrides the join condition. A final measure cannot be redefined.

Each dependency of a calculated measure on another measure or attribute may be marked mandatory. Thus marked measures or attributes must be referred to in redefinitions of the calculated measure. Full compliance checking, mandated by legal regulations and policies, is outside the scope of this paper.

Redefinitions of measures can be necessary in case of deselection. When a measure m is deselected, the customization must either deselect any dependent calculated measure m' as well, or provide a redefinition of m' that does not reference m.

If the deselected measure m is a mandatory component of m', or m' is final, there exists no valid redefinition of m'; the customization must deselect m'. If the deselected measure m is a mandatory component of a mandatory calculated measure m' the deselection of m is invalid.

Example 2.11 (Redefined measure). The multidimensional reference model may define the totalCosts measure as `Fact.costs + Fact.shippingCosts`, the totalCostsPerUnit measure as `(Fact.costs + Fact.shippingCosts)/Fact.orderedQuantity`. The customization in Fig. 6, due to the deselection of the shippingCosts measure, must redefine the totalCosts and totalCostsPerUnit measures, for example, by selecting only the costs measure or by adding a constant amount, say 40, as handling fee instead of the shippingCosts measure, `Fact.costs + 40`.

2.2.3. *Predicates*

A customization may redefine predicates. Figure 8 defines the metamodel for the redefinition of dimension predicates; other types of predicates are redefined analogously. The feature of predicate redefinition is convenient since business terms are interpreted slightly differently across enterprises, local subsidiaries, and departments. Predicates are used for the definition of reference analysis situations (see Sec. 3.1.1) which represent condensed views of interest on the data warehouse. Queries that employ particular predicates might be useful in multiple companies, local subsidiaries, and departments. For example, a reference analysis situation that returns the costs of heat-resistant materials might be considered useful by several local subsidiaries producing different types of brushes for different application domains. The exact definition of "heat-resistant material" may differ, however, depending on whether brushes are used on airfields, in industrial production, or for hygiene purposes. A redefinition of the corresponding predicate allows for the use of the same analysis situation in different contexts.

Just like redefinitions of measures, redefinitions of predicates may become necessary in case of deselections. In case of an attribute's deselection — or the corresponding level's, dimension's, or hierarchy's deselection — the customization must either

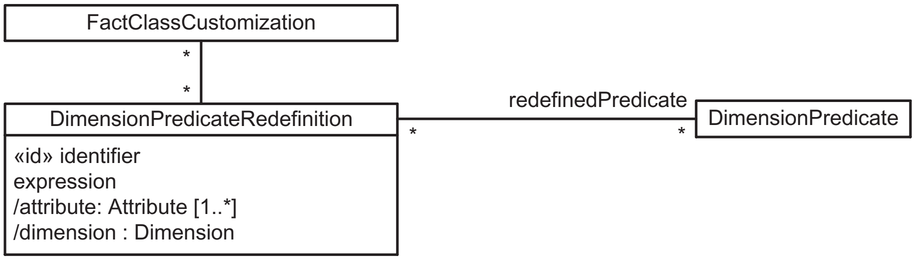

Fig. 8. Dimension predicate redefinitions in the metamodel of multidimensional reference models. Redefinitions of multidimensional and measure predicates are defined analogously.

deselect all dependent predicates as well or provide a redefinition that does not feature the deselected attribute. If the deselected attribute, however, is a mandatory component of a predicate, or the predicate is final, there exists no valid redefinition and the customization must deselect the predicate.

2.3. *Star schema generation*

Conceptual data warehouse models are often translated into relational data models.[10,18] The translation process from conceptual data warehouse model to logical, relational data model can be automated. The Indyco Builder,[b] for example, allows for the automatic generation of star schemas from dimensional fact models. The resulting relational models may either be queried directly using SQL or serve as the basis for other dedicated reporting tools such as Microsoft SQL Server Analysis Services or Saiku Business Analytics.[c]

After taking into account the customizations we translate the multidimensional reference model's fact class and dimensions into a star schema with denormalized, flat dimension tables that each have a surrogate key.[25] The star schema data model is arguably the most popular relational form of organization for data warehouses. The star schema data model is easy to implement and understand. A star schema consists of a single fact table, which corresponds to a fact class, the tuples of which represent the facts. The fact table has a column for each measure and for each dimension of the fact class. The fact table references the dimension tables which store the roll-up hierarchies and the attributes of the levels. Note that only base measures translate into columns in the fact tables. Calculated measures correspond to combinations of base measure columns. Allowing nonstrict hierarchies (or multiple arcs) in the multidimensional reference model would result in a generated logical model with bridge tables and weight attributes.

Example 2.12 (Star schema). The generated star schema for the customization of the MaterialUsedForProduct and MaterialInSupplyOrder fact classes as defined in Fig. 6 consists of the following tables; primary keys are underlined, names of tables are system-generated:

$$\begin{gathered}
\text{Time\#1}(\underline{\text{id}}, \text{day, week, month, year});\\
\text{Product\#1}(\underline{\text{id}}, \text{productOrder, minTemperature, maxTemperature, minLifeTime});\\
\text{Customer\#1}(\underline{\text{id}}, \text{customer, customerRegion, industry});\\
\text{Material\#1}(\underline{\text{id}}, \text{material, minTemperature, maxTemperature, materialCategory});\\
\text{Supplier\#1}(\underline{\text{id}}, \text{supplier, supplierRegion});\\
\text{MaterialUsedForProduct\#1}(\underline{\text{time, product, customer, material}},\\
\text{plannedQuantity, actualQuantity, plannedCosts, actualCosts});\\
\text{MaterialInSupplyOrder\#1}(\underline{\text{time, material, supplier}},\\
\text{orderedQuantity, deliveredQuantity, costs}).
\end{gathered}$$

[b]http://www.indyco.com/builder/
[c]http://www.meteorite.bi/products/saiku

In this example the star schema tables derive from two customizations at once, the customizations of the MaterialUsedForProduct fact class and the MaterialInSupply-Order fact class. Shared dimensions, in this case, do not translate into different dimension tables. The simultaneous generation of dimension tables from two customizations only works if the shared dimensions apply the same customizations to these dimensions. Otherwise separate dimension tables must be generated during customization. The tables Time#1, Product#1, Customer#1, Material#1, and Supplier#1 are dimension tables. The fact tables MaterialUsedForProduct#1 and MaterialInSupplyOrder#1 reference the id column of the dimension tables as defined by the following inclusion dependencies which represent foreign key constraints:

$$\begin{gathered}
\mathsf{MaterialUsedForProduct\#1(time)} \subseteq \mathsf{Time\#1(id)},\\
\mathsf{MaterialUsedForProduct\#1(product)} \subseteq \mathsf{Product\#1(id)},\\
\mathsf{MaterialUsedForProduct\#1(customer)} \subseteq \mathsf{Customer\#1(id)},\\
\mathsf{MaterialUsedForProduct\#1(material)} \subseteq \mathsf{Material\#1(id)},\\
\mathsf{MaterialInSupplyOrder\#1(time)} \subseteq \mathsf{Time\#1(id)},\\
\mathsf{MaterialInSupplyOrder\#1(material)} \subseteq \mathsf{Material\#1(id)},\\
\mathsf{MaterialInSupplyOrder\#1(supplier)} \subseteq \mathsf{Supplier\#1(id)}.
\end{gathered}$$

3. Analysis Processes

In this section we discuss the modeling and customization of analysis processes, using a revised and extended example from previous work.[16] Analysis process models externalize knowledge about how to obtain the information required for an appropriate reaction to a given business situation, sparing business analysts from *ad hoc* query formulation in cases of urgent information needs.

3.1. *Reference modeling*

An analysis graph models the succession of analysis situations during the analysis. Transitions in the analysis graph apply OLAP operations in order to transform a source analysis situation into a target analysis situation.

3.1.1. *Analysis situations*

An analysis situation is a view of interest over a fact class which analysts obtain through the application of slice and dice as well as roll-up operations on the base data. Typically, the specification of analysis situations contains a number of variable elements that are to be provided by the analyst during the actual analysis. For example, an analysis situation may see the retrieval of quantity and cost of material used in the production process of products during a particular time period. This analysis situation may occur frequently, with analysts providing concrete values for the products and time period of interest. Thus, the specification of an analysis

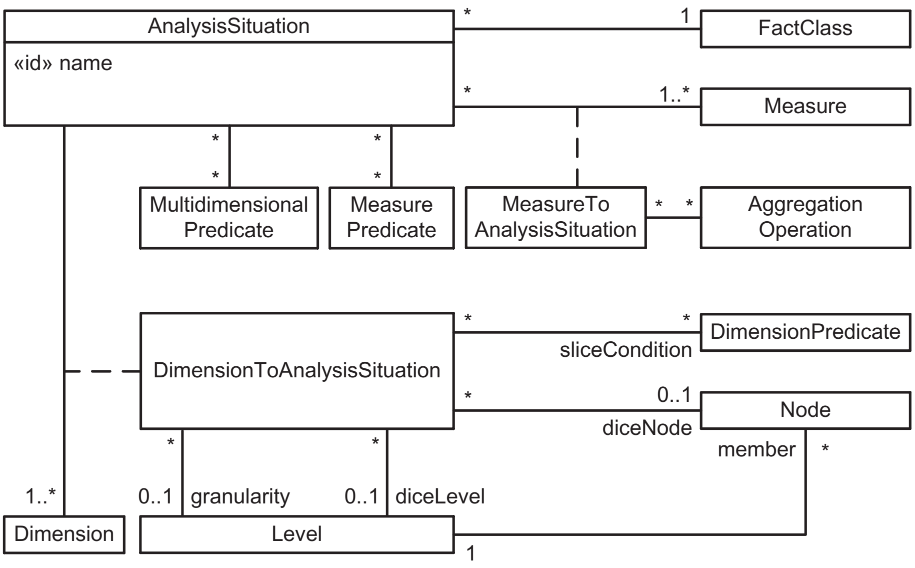

Fig. 9. The metamodel of analysis situations as UML class diagram.

situation corresponds to a generic query; the analyst may provide concrete values for an analysis situation's variables in order to obtain an executable query.

Figure 9 illustrates the metamodel for the specification of analysis situations. An analysis situation (class AnalysisSituation) has a unique name and refers to a single fact class. An analysis situation refers to at least one measure of this fact class. For each dimension of the fact class the analysis situation may restrict the facts to be included in the query result by associating a number of dimension predicates and dice nodes. The DimensionToAnalysisSituation association class attaches to the analysis situation the slice conditions as well as granularity and dice parameters. Multidimensional and measure predicates represent slice conditions of an analysis situation. An arbitrary number of dimension predicates (association end sliceCondition from DimensionToAnalysisSituation to DimensionPredicate) further governs the selection of facts based on attributes of the dimension. An analysis situation may also refer to only a region (or subcube) defined by nodes from the different dimensions of the fact class (diceNode to Node). A node is member of a level, for example, *Bristle Material* is a materialCategory-level node in the Material dimension. Furthermore, an analysis situation may represent the result of a roll-up operation to some granularity level (granularity to Level). In that case, the query result applies a set of given aggregation functions, as defined individually for each measure by the MeasureToAnalysisSituation association class, to the measures of interest.

Elements in the specification of an analysis situation may also be variables the concrete values of which are to be provided by the analyst while conducting the actual analysis. A variable has a name which by convention starts with a question mark. Formally, for each of the classes Measure, AggregationOperation, DimensionPredicate, MultidimensionalPredicate, MeasurePredicate, Node, and Level there exists a corresponding Variable subclass with a name attribute. In order to provide guidance to the analyst, an analysis situation in a reference model may specify the level of an expected dice node (association end diceLevel from DimensionToAnalysisSituation to Level in Fig. 9), thereby constraining possible instantiations of variables for the diceNode association end.

Example 3.1 (Analysis situation). The CostsInPeriodOfMaterialOfCategoryWithPropertyInSupplyOrder analysis situation in Fig. 10 refers to the MaterialInSupplyOrder fact class and from this fact class selects the costs measure. The analysis situation requires the analyst to provide as dice coordinates a node from the Time dimension (variable ?tm) as well as a materialCategory-level node from the Material dimension (?mat). The analyst must specify the desired granularity level in the Time dimension (?tmGranularity) whereas the granularity level for the Material dimension is fixed to material. The analyst must provide a predicate in the Material dimension as selection criteria (?prop) to select only data about materials with a specified property.

Through the inclusion of variables, an analysis situation becomes a reusable query schema. Analysts provide specific values in order to reuse this analysis situation. For example, by providing specific values for the variables in the CostsInPeriodOfMaterialOfCategoryWithPropertyInSupplyOrder analysis situation in Fig. 10, an analyst may obtain the monthly costs in the year 2015 — with month as the Time dimension's granularity, and 2015 as the dice node — for orders of heat-resistant materials in the *Bristle Material* category — with the HeatResistantMaterial predicate as the Material dimension's slice condition, and *Bristle Material* as the dice node.

3.1.2. *Analysis graphs*

Analysis graphs define default analysis processes triggered by business events, for example, a delayed supply order. An adequate reaction to business events requires information. Analysis graphs indicate how the analyst may obtain required information for considered reactions to business events. An analysis graph consists of analysis situations linked by OLAP operations which transform one analysis situation into another. Figure 11 shows an unrefined, bird's-eye view of an analysis graph that represents an analysis process triggered in the event of a material supply order's cancellation. Each rounded box corresponds to an analysis situation whereas the arrows between boxes correspond to OLAP operations. Supply orders may be canceled due to various reasons, for example, the inability of a supplier to deliver required material within a given time frame. The cancellation of a supply order necessitates the replacement of the undelivered material in the manufacturing processes that depend on the canceled order. In order to react adequately to the cancellation the

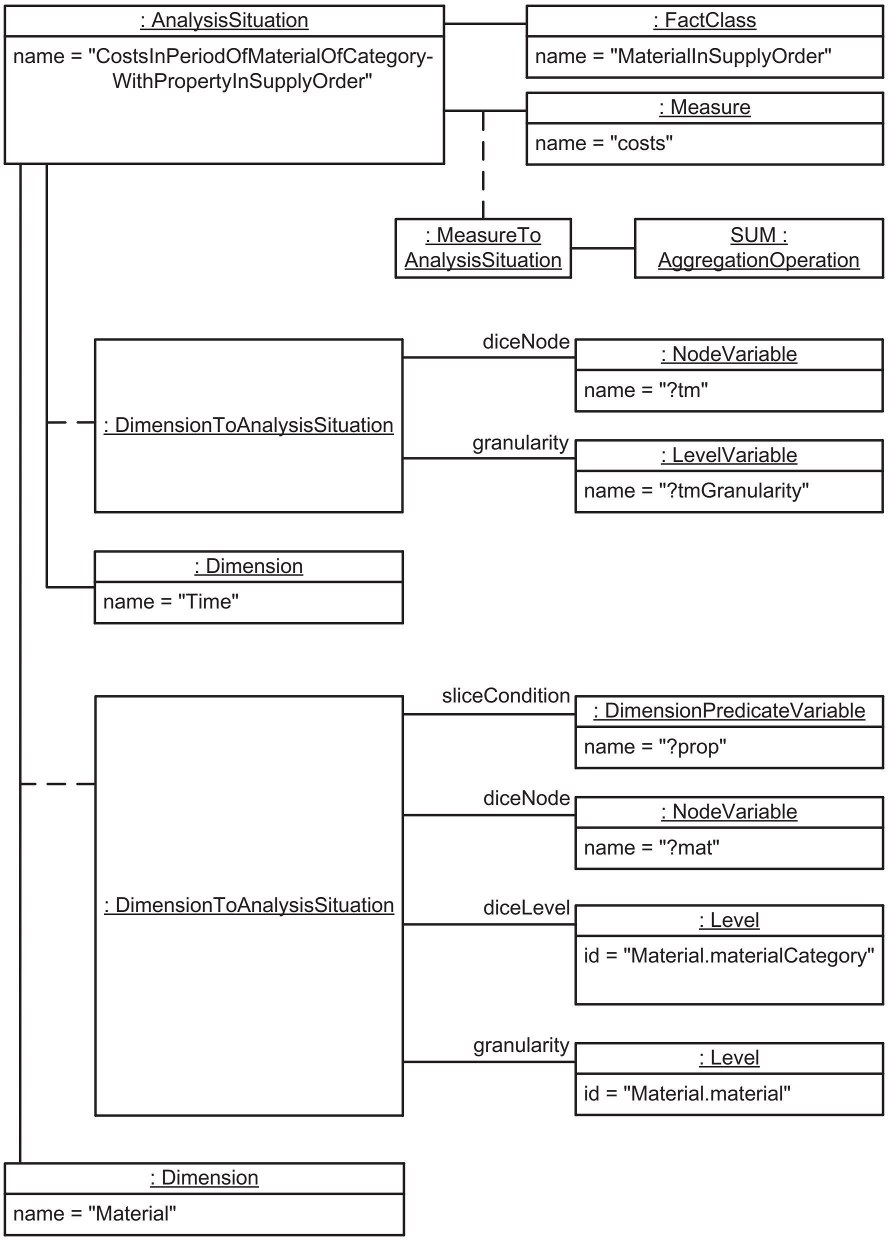

Fig. 10. The specification of an example analysis situation as UML object diagram.

analyst must first identify quantity and expected delivery time of the material from the canceled supply order. The identification of other supply orders for the same material may already resolve the shortage of material. Otherwise, the analyst must identify products affected by the cancellation. The analyst may then either identify

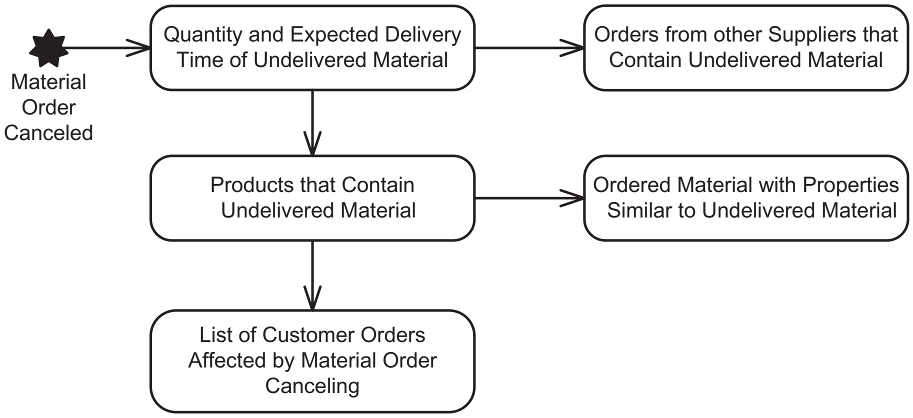
Material Order Canceled
Quantity and Expected Delivery Time of Undelivered Material
Orders from other Suppliers that Contain Undelivered Material
Products that Contain Undelivered Material
Ordered Material with Properties Similar to Undelivered Material
List of Customer Orders Affected by Material Order Canceling

Fig. 11. An unrefined analysis graph for analysis in the event of order cancellation.

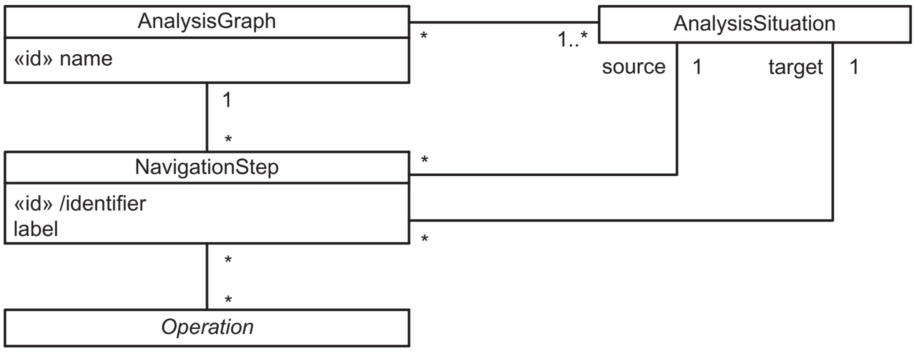

Fig. 12. The metamodel of analysis graphs as UML class diagram.

alternative materials with similar properties in other orders or compile a list of affected customers that the company cannot serve due to material shortage.

Figure 12 defines, using UML notation, the metamodel for the specification of analysis graphs. An analysis graph, which has a unique name, consists of analysis situations and navigation steps (class NavigationStep). Each navigation step has a label as well as a source analysis situation and a target analysis situation. A navigation step applies a number of OLAP operations to the source analysis situation in order to obtain the target situation. Conceptually, the analysis graph consists of specifications of analysis situations with variables, the bindings of which are changed at runtime by navigation steps. An analysis graph is similar to a state chart (or UML state machine diagram), the states being analysis situations and the transitions being invocations of OLAP operations. Technically, an analysis graph moves from one cube to another, the cubes being results of queries defined by the analysis situations.

Table 1. Navigation operators in analysis graphs.

Dice	
moveToNode(D,L,N)	Replace the dice node in dimension D with node N at level L.
Slice	
addSliceCondition(D,P)	Add dimension predicate P to the slice conditions of dimension D.
removeSliceCondition(D,P)	Remove dimension predicate P from the slice conditions of dimension D.
addSliceCondition(P)	Add predicate P to the slice conditions of the analysis situation.
removeSliceCondition(P)	Remove predicate P from the slice conditions of the analysis situation.
Granularity	
changeGranularity(D,L)	Replace the granularity in dimension D with level L. This operation corresponds to roll-up or drill-down if L is coarser or finer, respectively, than the original granularity.
setAggregationFunction(M,F)	Set aggregation function F for the roll-up of measure M.
Projection	
addMeasure(M)	Select measure M for display.
removeMeasure(M)	Project away measure M.
changeFactClass(F)	Select fact class F for display. For dimensions shared by source and target situations slice and dice conditions remain; common measures remain.

The navigation steps between analysis situations model the transformation of one analysis situation into another, defining the "delta" in terms of OLAP operations between the connected analysis situations. Table 1 provides an overview of OLAP operations for the modeling of navigation steps, grouped into the four broad categories *Dice*, *Slice*, *Granularity*, and *Projection*. An additional operator for selection of groups, similar to the `HAVING` clause in SQL, may also be incorporated. The moveToNode operation replaces the dice node of an analysis situation in a given dimension with another node, possibly reducing or expanding the number of facts that are considered in the query result. The addSliceCondition and removeSliceCondition operations add and remove predicates, respectively. Depending on the parameters, the predicate is either a dimension predicate or multidimensional/measure predicate. The changeGranularity operation changes the level of granularity of an analysis situation. The setAggregationFunction operation sets an aggregation function to be used for the roll-up of a particular measure. The addMeasure and removeMeasure operations change the measures that are returned by the query. The changeFactClass operation allows for changing the fact class that serves as the basis for the analysis. The list of operators may be extended further and refined.

In the specification of analysis graphs the parameters of OLAP operations may be either concrete values or variables. Analysts must provide concrete values for variables upon actually taking a navigation step. Concrete values may constrain the variables in the specification, for example, a concrete value for the level in the specification of a dice operation restricts possible values for the node.

Example 3.2 (Analysis graph). Figure 13 illustrates an analysis graph that consists of four analysis situations. The first analysis situation, CostsOfMaterial-SupplyOrder, returns the totalCosts base measure as well as the calculated measures totalCostsPerUnit, totalCostsYTD, and totalCostsToPreviousWeek from the Material-InSupplyOrder fact class. The CostsOfMaterialSupplyOrder analysis situation restricts facts to particular material-level nodes and allows for an arbitrary restriction of the time period. The CostsOfMaterialSupplyOrder analysis situation performs no aggregation. The navigation step labeled DisplayMonthlyCosts transforms CostsOf-MaterialSupplyOrder into MonthlyCostsOfMaterialSupplyOrder by removing measures totalCostsPerUnit, totalCostsYTD, and totalCostsToPreviousWeek, performing a roll-up operation to the month level, and selecting the totalCostsYTDRollUpByMonth measure, a cumulative measure specifically defined for the selected granularity. The navigation step labeled FocusOnMaterialUse subsequently changes the analyzed fact class to MaterialUsedForProduct. Parameters for common dimensions of MaterialIn-SupplyOrder and MaterialUsedForProduct remain. The navigation step labeled Focus-OnProperty narrows the slice condition in the Material dimension and restricts the product category.

3.2. *Customization*

Analysis situations and graphs are reference models that may be customized. Customizations of analysis situations and graphs are not independent from the customization of the underlying multidimensional reference model.

3.2.1. *Analysis situations*

The customization of an analysis situation may deselect or add measures, deselect or add predicates, and change the analysis situation's granularity and dice properties. In particular, a customization may replace variables in the specification of an analysis situation with a concrete value. The deselection of a measure from an analysis situation upon which other measures depend has no consequences on the dependent measures. An analysis situation is merely a view over the fact class. The values of the measures in the analysis situation are obtained from the original facts. The deselection of a measure from the analysis situation does not remove the measure from the fact class.

Analysis situations referencing particular predicates might be useful in different companies even though business terms are interpreted slightly differently across enterprises (see predicate redefinition in Sec. 2.2). For example, a reference analysis situation on the order costs for heat-resistant material might be considered useful by several companies. Nevertheless, the exact definition of "heat-resistant material" may differ, depending on the application scenario. A redefinition of the corresponding predicate allows for the use of the same analysis situation in different contexts.

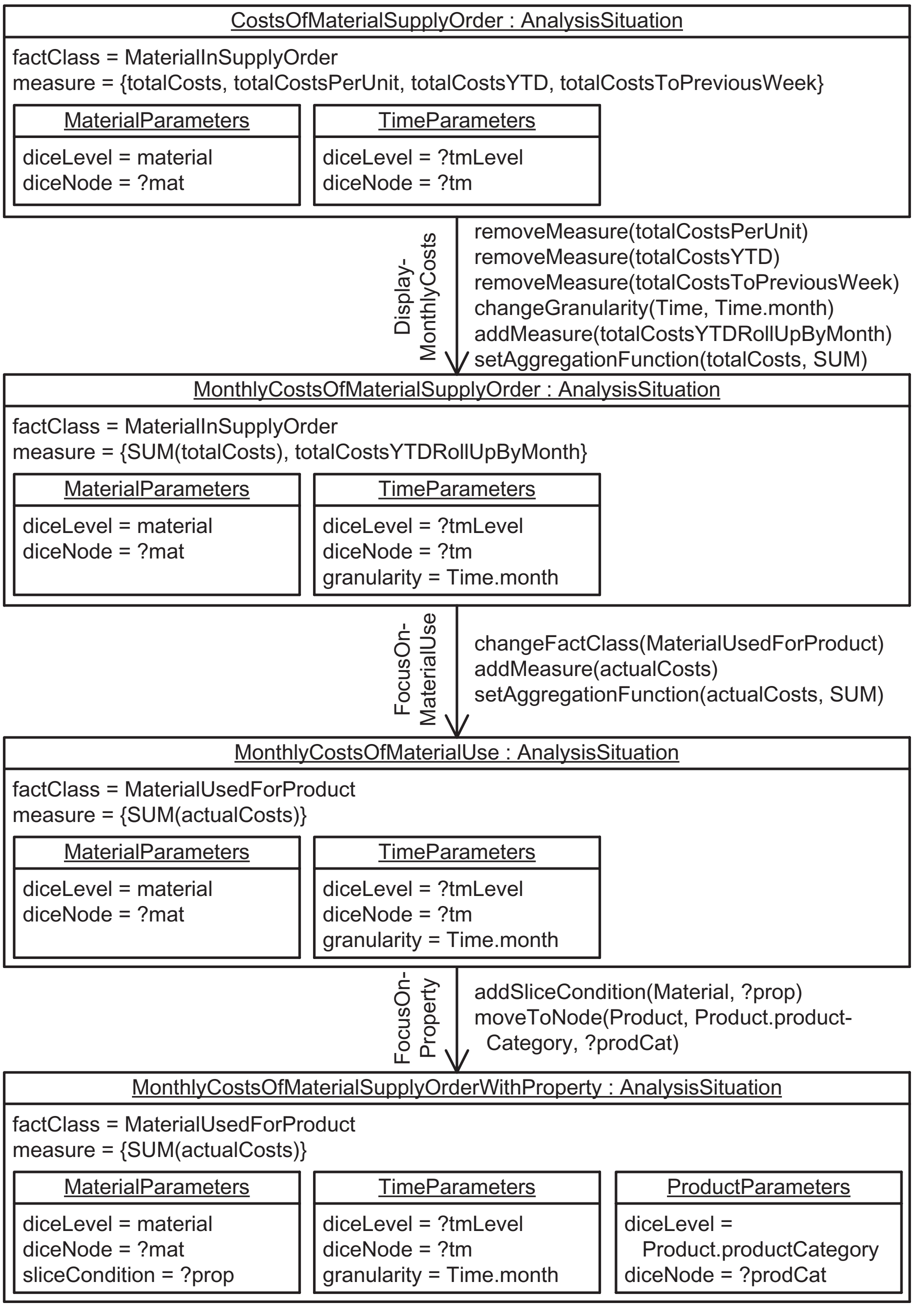
CostsOfMaterialSupplyOrder : AnalysisSituation
factClass = MaterialInSupplyOrder
measure = {totalCosts, totalCostsPerUnit, totalCostsYTD, totalCostsToPreviousWeek}
MaterialParameters
diceLevel = material
diceNode = ?mat
TimeParameters
diceLevel = ?tmLevel
diceNode = ?tm
Display-MonthlyCosts
removeMeasure(totalCostsPerUnit)
removeMeasure(totalCostsYTD)
removeMeasure(totalCostsToPreviousWeek)
changeGranularity(Time, Time.month)
addMeasure(totalCostsYTDRollUpByMonth)
setAggregationFunction(totalCosts, SUM)
MonthlyCostsOfMaterialSupplyOrder : AnalysisSituation
factClass = MaterialInSupplyOrder
measure = {SUM(totalCosts), totalCostsYTDRollUpByMonth}
MaterialParameters
diceLevel = material
diceNode = ?mat
TimeParameters
diceLevel = ?tmLevel
diceNode = ?tm
granularity = Time.month
FocusOn-MaterialUse
changeFactClass(MaterialUsedForProduct)
addMeasure(actualCosts)
setAggregationFunction(actualCosts, SUM)
MonthlyCostsOfMaterialUse : AnalysisSituation
factClass = MaterialUsedForProduct
measure = {SUM(actualCosts)}
MaterialParameters
diceLevel = material
diceNode = ?mat
TimeParameters
diceLevel = ?tmLevel
diceNode = ?tm
granularity = Time.month
FocusOn-Property
addSliceCondition(Material, ?prop)
moveToNode(Product, Product.product-Category, ?prodCat)
MonthlyCostsOfMaterialSupplyOrderWithProperty : AnalysisSituation
factClass = MaterialUsedForProduct
measure = {SUM(actualCosts)}
MaterialParameters
diceLevel = material
diceNode = ?mat
sliceCondition = ?prop
TimeParameters
diceLevel = ?tmLevel
diceNode = ?tm
granularity = Time.month
ProductParameters
diceLevel = Product.productCategory
diceNode = ?prodCat

Fig. 13. An example analysis graph.

3.2.2. *Analysis graphs*

An analysis graph customization may add or deselect analysis situations and navigation steps. On the one hand, this allows for the introduction of additional paths in the specification of an analysis graph. On the other hand, existing paths from the reference model may be refined by introducing additional steps in the analysis graph. Likewise, existing paths may be shortened by removing analysis situations and redefining the navigation steps.

Figure 14 shows the metamodel for the customization of analysis graphs. The customization of an analysis graph (class AnalysisGraphCustomization) refers to a single analysis graph and may have a number of step redefinitions (StepRedefinition) which refer to existing navigation steps, redefining source and target situations as well as the performed operations. A redefinition of a navigation step may become necessary when customizing analysis situations. In that case navigation steps must be kept consistent with source and target analysis situations as navigation steps specify the "delta" between these situations. An analysis graph customization requires a customization for each fact class referenced by any of the analysis situation in the analysis graph (customization).

An analysis graph customization also associates each fact class used in one of the analysis graph's analysis situations with at most one fact class customization. Thus, an analysis situation customization must also take into account a customization of the analysis situation's fact class. An analysis graph customization does not associate multiple fact class customizations with a fact class.

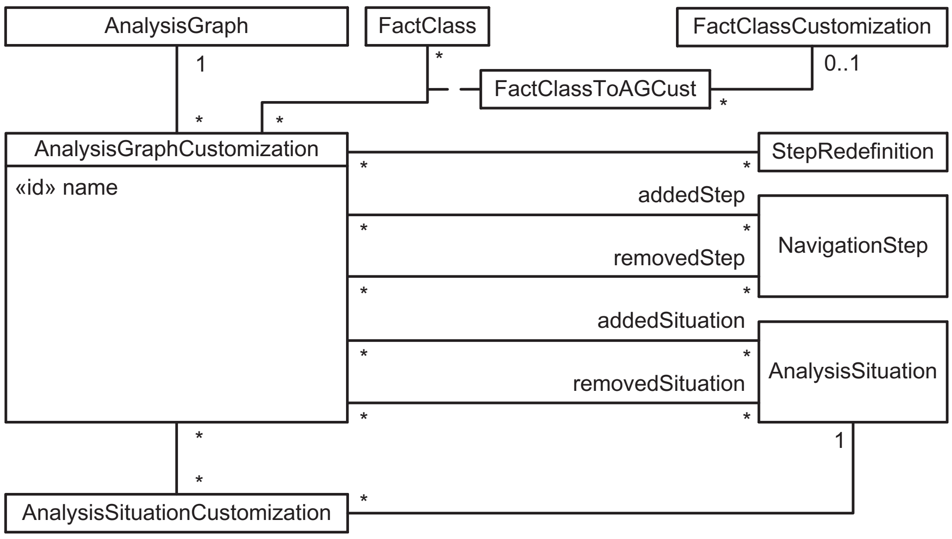

Fig. 14. The metamodel of analysis graph customizations as UML class diagram.

3.3. *Schema generation*

Analysis situations translate into (parameterized) queries that represent views over the data warehouse. Analysis graphs translate into State Chart XML documents.

3.3.1. *Analysis views*

An analysis situation translates into an OLAP query over the star schema tables. The use of surrogate keys in the star schema tables facilitates the automated generation of analysis views. In all dimension tables, the fact table references the id attribute. The calculation rules of calculated measures are in the `SELECT` clause; references to other calculated measures must be resolved prior to code generation. When translating an analysis situation with more than a single comparative measure, nested queries must be used to define the view. Otherwise, due to the use of the ComparisonFact qualifier in join conditions, the query would have faulty joins. Each comparative measure then has its own subquery. Base, arithmetic, and cumulative measures are in another subquery. An outer query then performs a natural join over the dimension attributes before grouping the result according to the specified roll-up levels with the specified aggregation operators.

Example 3.3 (Analysis view). The query in Listing 1 is an instantiation of the CostsOfMaterialSupplyOrder analysis situation in Fig. 13. The example illustrates the use of cumulative measures and comparisons. Notice the inclusion of the calculation rules in the `SELECT` clause of the query. The comparison features a special join condition. Since only one comparative measure is present there is no need for subqueries. The `WHERE` clause shows the selection of particular dice nodes.

Listing 1. Generated query for the CostsOfMaterialSupplyOrder analysis situation in Fig. 13.

```
SELECT (Fact.costs + 40) AS totalCosts,
       Fact.costs/Fact.deliveredQuantity
         AS totalCostsPerUnit,
       SUM(Fact.costs) OVER (
         PARTITION BY
           Time.year, Fact.material, Fact.supplier
         ORDER BY Time.day
       ) AS totalCostsYTD,
       Fact.costs/ComparisonFact.costs
         AS totalCostsToPreviousWeek,
       Fact.time, Material.id, Supplier.id
FROM MaterialInSupplyOrder#1 Fact JOIN
     Time#1 Time ON Fact.time = Time.id JOIN
     Material#1 Material ON
       Fact.material = Material.id JOIN
     Supplier#1 Supplier ON
     Fact.supplier = Supplier.id LEFT OUTER JOIN (
       MaterialInSupplyOrder#1 ComparisonFact JOIN
       Time#1 ComparisonTime ON
         ComparisonFact.time =
           ComparisonTime.id JOIN
       Material#1 ComparisonMaterial ON
         ComparisonFact.material =
           ComparisonMaterial.id JOIN
       Supplier#1 ComparisonSupplier ON
         ComparisonFact.supplier =
           ComparisonSupplier.id
     ) ON Time.day =
            DATEADD(week, 1, ComparisonTime.day) AND
     Fact.Material = ComparisonFact.Material AND
     Fact.Supplier = ComparisonFact.Supplier
WHERE Time.year = 2015 AND
      Material.material = 'Bristle Material'
```

Since cumulative measures cannot be sensibly aggregated, special variants of cumulative measures may be defined to be displayed at different roll-up levels, with only an additional aggregation operator added to the calculation rule and a slightly changed `ORDER BY` clause. For example, the totalCostsYTDRollUpByMonth measure could define year-to-date total costs for each month. In case the code generator generates an outer query which performs the roll-up, with several subqueries in the `FROM` clause for selecting base and comparative measures, the cumulative measure for this roll-up level becomes part of the outer query instead of the subqueries.

Example 3.4 (Analysis view). The query in Listing 2 is a possible instantiation of the MonthlyCostsOfMaterialSupplyOrder analysis situation in Fig. 13. The query performs a roll-up to the month level, requiring the definition of an aggregation operation for the base and calculated measures. The totalCostsYTDRollUpByMonth has no aggregation operator specified, instead being a cumulative measure specifically defined to be displayed at this roll-up granularity.

Listing 2. Generated query for the MonthlyCostsOfMaterialSupplyOrder analysis situation in Fig. 13.

```
SELECT SUM(Fact.costs + 40) AS totalCosts,
       SUM(SUM(Fact.costs + 40)) OVER (
         PARTITION BY
           Time.year, Fact.material, Fact.supplier
         ORDER BY Time.month
       ) AS totalCostsYTDRollUpByMonth,
       Time.month, Time.year,
       Fact.material, Fact.supplier
FROM MaterialInSupplyOrder#1 Fact JOIN
     Time#1 Time ON Fact.time = Time.id JOIN
     Material#1 Material ON
       Fact.material = Material.id JOIN
     Supplier#1 Supplier ON
       Fact.supplier = Supplier.id
WHERE Time.year = 2015 AND
      Material.material = ‘Bristle Material’
GROUP BY Time.month, Time.year,
         Fact.material, Fact.supplier
```

Multiple arcs in the multidimensional model could result in additional bridge tables in the star schema, which must be incorporated accordingly in the generated queries. Instead of referencing a single table for each dimension, a subquery for each dimension must first join the dimension and bridge tables. Roll-up queries and calculated measures must take into account the weight attribute or assume an equal split of weights. We refer to general literature[18] for handling bridge tables.

Note that Listings 1 and 2 contain specific instantiations of analysis situations. In case of variables being present in the definition of analysis situations, the queries are parameterized. These parameterized queries contain placeholders to be replaced by specific string values that are selected at runtime.

3.3.2. *Workflow model*

Analysis graphs must be translated into workflow models that are self-contained, executable models with all the necessary information to perform the analysis. We propose a representation of workflow models of analysis graphs in State Chart XML,[17] a W3C recommendation for the representation of state charts (or state machine diagrams) in XML. State charts are a natural fit for the representation of analysis graphs, each analysis situation being considered a state. SCXML is a lightweight language for representing state charts in a machine-readable format. The logical representation of analysis graphs, however, may also employ another representation format.

The basic idea of an analysis graph workflow model is that each analysis situation becomes a state in the state chart. Transitions represent navigation steps, the event that triggers these transitions is the label of the corresponding navigation step in the reference model. The operations are realized as SCXML custom action elements. These custom action elements also hold the definitions of calculated measures and predicates as well as allowed nodes and levels for the selection as argument by the analyst, in order to make the workflow model self-contained.

The operations may be implemented in XQuery or another programming language, for example, Java.

Example 3.5 (Workflow model). The workflow model in Listing 3 corresponds to the analysis graph in Fig. 13. Analysis situations translate into `state` elements in the SCXML workflow model, the `id` attribute containing the name of the analysis situation. The transition elements under the state elements represent navigation steps of the analysis graph. The child elements of the transition elements represent the operations associated with the navigation steps. For example, the `addSliceCondition` element represents the addSliceCondition operation, the `moveToNode` element represents the moveToNode operation. The `var` elements under these action elements contain the possible values of bind variables or queries on the instance data that allow to retrieve the possible values. For example, the `addSliceCondition` element, in this case, provides inline definitions of the predicates, taking into account possible customizations, that can be selected as value for the ?prop variable. The `moveToNode` element provides a query for the retrieval of productCategory-level nodes in the product dimension, which can be selected as value for the ?prodCat variable. The `datamodel` element of the SCXML workflow model contains a `data` element for each bind variable, each initially empty, and a `data` element for each analysis situation that contains a parameterized query.

4. Rationale and Evaluation

In this section, we discuss how BIRD meets certain requirements for lightweight reference modeling approaches. We discuss applicability of BIRD to real-world projects. We conclude with a short presentation of proof-of-concept prototypes.

4.1. *Requirements*

The BIRD approach aims at lowering the obstacles that inhibit SMEs from employing BI technology by allowing for the use of industry-specific best practices represented as reference models. From the general aim of BIRD we derive the following design goals:

(i) A lightweight approach for domain-specific reference modeling of static and behavioral aspects of data analysis with data warehouses.
(ii) Support for the flexible adaptation of industry-specific best-practice reference models to the needs of individual companies.
(iii) Contribution to the implementation of BI solutions.
(iv) Adequacy for SMEs with regard to technological and methodological limitations of IT infrastructure and personnel of SMEs.
(v) Simplicity, understandability, and ease of application for business analysts.

Listing 3. Extract of the workflow model for a customization of the analysis graph in Fig. 13.

```
<scxml initial="MonthlyCostsOfMaterialUse"
 xmlns:sc="http://www.w3.org/TR/scxml"
 xmlns:ag="http://www.dke.jku.at/AnalysisGraph">
 <sc:datamodel>
  <sc:data id="?mat"/>
  <sc:data id="?tmLevel"/>
  <sc:data id="?tm"/>
  <sc:data id="?prop"/>
  <sc:data id="?prodCat"/>
  <data id="CostsOfMaterialSupplyOrder"> SELECT ...
  <data id="MonthlyCostsOfMaterialSupplyOrder"> SELECT ...
  <data id="MonthlyCostsOfMaterialUse"> SELECT ...
  <data id="MonthlyCostsOfMaterialSupplyOrderWithProper ...
 </sc:datamodel>
 <sc:state id="CostsOfMaterialSupplyOrder" ...
 <sc:state id="MonthlyCostsOfMaterialSupplyOrder" ...
 <sc:state id="MonthlyCostsOfMaterialUse">
  <sc:transition event="FocusOnProperty"
     target="MonthlyCostsOfMaterialSupply-
       OrderWithProperty">
   <ag:addSliceCondition dimension="Material"
      predicate="?prop">
    <ag:var name="?prop">
     <ag:predicate name="HeatResistantMaterial" expr="...
     <ag:predicate name="ColdResistantMaterial" expr="...
    </ag:var>
   </ag:addSliceCondition>
   <ag:moveToNode dimension="Product"
      level="Product.productCategory"
      node="?prodCat">
    <ag:var name="?prodCat">
    SELECT DISTINCT Product.productCategory
    FROM Product#1 Product
    </ag:var>
   </ag:moveToNode>
  </sc:transition>
 </sc:state>
 <state id="MonthlyCostsOfMaterialSupplyOrderWithProper ...
</sc:scxml>
```

The BIRD approach does not cover all aspects of data warehousing. ETL processes, which heavily depend on the company's operational source systems, are difficult to generalize with reference models and, therefore, the representation of ETL processes is out of BIRD's scope. Assuming that data visualization does not differ much between different industrial sectors and companies but depends heavily on the available visualization frontend, domain-specific reference modeling of data visualization is out of BIRD's scope.

In the following, we derive from design goals and scope the requirements for the development of the BIRD approach. We discuss the rationale behind each individual requirement and evaluate BIRD's compliance with the respective requirement.

Requirement 1 (Definition of business ratios). The proposed reference modeling approach should support the reuse, adaptation, and multidimensional analysis of business ratios.

Rationale: At the core of any BI solution are business ratios, calculated from data coming from operational systems, serving as KPIs. Typically, analysts inspect business ratios for different parts of a business at various levels of granularity (multidimensional analysis) in order to monitor business performance and assess success. Many business ratios are domain-specific, i.e. specific to an industrial sector. Companies may specify additional company-specific business ratios and may adapt the definition of business ratios to their specific setting, taking into account the data that is actually available in the operational systems. Thus, business ratios are well suited for representation in reference models as well as the subsequent reuse and customization.

Evaluation: BIRD represents business ratios as calculated measures, each with a calculation rule that can be adapted to the data available in the data warehouse of a specific company. The association of a calculated measure with multiple fact classes (association class MeasureToFactClass in Fig. 1) in a multidimensional reference model may be regarded as a family of business ratios with the same calculation rule but with different base data, possibly at different granularities, that go into the calculation. A calculated measure associated with a fact class serves as the core for the definition of a KPI. A KPI's full definition[26,27] additionally comprises the association of the calculated measure with the organizational context (responsibilities, goals, justifications, etc.) and is outside of BIRD's scope.

Requirement 2 (Definition of business terms). The proposed reference modeling approach should support the reuse and adaptation of business terms as well as their use in multidimensional analyses.

Rationale: Business analysts communicate with each other analysts, managers, and executives using business terms. The meaning of business terms typically is not precisely defined, let alone explicitly represented in BI systems. Business terms are typically translated into queries against the data warehouse in an *ad hoc* manner, again and again for every single analysis. The *ad hoc* translation of business terms into queries inhibits the understanding of the rationale behind analytical queries, hindering the reuse and interpretation of queries. Therefore, business terms should be explicitly represented in BI systems together with a specification that makes business terms usable for analytical queries. Business terms are domain-specific, i.e. specific to a particular industrial sector. Every company may have its own 'business dialect', with slightly different meanings of business terms. The translation of business terms into queries against the data warehouse must consider the available data which may differ across companies or departments. Business terms are predisposed for reference modeling as well as subsequent reuse and customization.

Evaluation: The BIRD approach has the reference modeler translate the meaning of business terms into query fragments against the multidimensional schema, constituting named predicates for reuse in multiple models. Predicates may be customized to the specific needs of a particular company or department. Predicates may be reused in different analysis situations and analysis graphs.

Requirement 3 (Definition of queries and analyses). The proposed reference modeling approach should support the reuse and adaptation of queries and analyses on an *ad hoc* basis.

Rationale: Different business situations impose different information needs. Often such information needs cannot be satisfied by a single multidimensional query but require a set of interrelated queries. Companies in the same industrial sector experience similar business situations with similar information needs, making analysis process modeling a case for reference modeling. Since it is impossible to foresee every information need during customization there should be an *ad hoc* reuse mechanism that provides patterns or templates that can be reused to satisfy information needs occurring in different, yet similar situations in day-to-day analysis work.

Evaluation: BIRD's analysis situations provide a very flexible query-reuse mechanism. Every part of an analysis situation may be a variable which can be set during customization but can also be left open to be set by the business analyst at query time. BIRD's analysis graphs allow for the representation of a set of analytical queries with relationships between each other.

Requirement 4 (Standard IT infrastructure). The proposed reference modeling approach should generate code, models, or configurations in standard languages deployable on IT infrastructure available in most SMEs and maintainable by IT personnel of SMEs.

Rationale: We assume that a typical SME has a small IT department with limited budget. IT personnel at SMEs typically consists of IT generalists, not BI experts, yet able to set up a (relational) database management system (DBMS). IT personnel should nevertheless be able to understand and maintain code as well as troubleshoot problems. IT management, however, is often reluctant to introduce new technology and will prefer systems that match the skills of existing personnel. Thus, nonexpert IT personnel should be able to deploy the generated code.

Standard and widely-known technologies facilitate the adoption of BI for IT generalists. The IT infrastructure necessary for running the code should be readily available in most SMEs or it should be easy for nonexpert IT personnel to set up the infrastructure.

Evaluation: BIRD is ROLAP approach with generation of SQL data definition and query statements, thus running on a relational DBMS. The use of a star schema logical model requires no specific hardware or specific DBMS but allows for the employment of readily available software solutions.

Requirement 5 (Intuitive modeling language). The proposed reference modeling approach should employ a modeling language based on standard and intuitive modeling language(s) and methodologies.

Rationale: Business analysts should be able to use the reference modeling language. We assume that a typical business analyst in an SME holds a bachelor's degree in business administration or similar. A business analyst is assumed to have basic skills in data and process modeling, SQL, as well as spreadsheet software, taught as part of many business administration curricula. With these basic IT skills a business analyst should be able to use the approach without much additional learning effort. Basing the approach on standard languages should facilitate the employment and extension of existing modeling and CASE tools for reference modeling.

Evaluation: The BIRD reference modeling language is based on UML, the DFM, and state charts interspersed with fragments of SQL. Existing modeling tools can be adapted for reference modeling as demonstrated by a proof-of-concept prototype that employs Indyco Builder for multidimensional reference modeling and star schema generation (see Sec. 4.3).

4.2. *Applicability to real world projects*

The agriProKnow project,[d] as a joint experimental research effort between industry and academia, investigates the possibilities of data analysis in precision dairy farming. Modern milk production generates vast amounts of data, tracking various indicators from animal movement to milk quality and quantity to animal health. The available data may be analyzed in order to improve animal wellbeing and, consequently, increase animal productivity. Farmers often diagnose sickness only when symptoms become apparent. Data analysis promises the timely diagnosis of sickness which then triggers appropriate countermeasures in order to find the source of the sickness and combat its effects.

An interfarm data warehouse integrates data from several dairy farms in order to allow for the generation of metaknowledge. Domain experts in veterinary medicine employ the interfarm data warehouse for discovering indicators and early warning

[d]http://www.agriProKnow.com/

signs for specific sicknesses. Based on these findings, the domain experts in collaboration with BI experts define queries of interest and analysis processes that guide dairy farm managers in managing animal wellbeing and optimize production output of the dairy herd.

Farm-specific data warehouses organize the data generated in dairy production in a representation format fit for day-to-day analysis work at individual dairy farms. Dairy farm managers operate on farm-specific data warehouses by executing the queries of interest and analysis processes discovered by domain experts using the interfarm data warehouse. When warning signs are detected in the data, for example, an indicator for ketosis, the system alerts the dairy farm manager who can follow a specified analysis process to discover the source of the problem.

The availability of BIRD reference models facilitates the development of farm-specific data warehouses. Although similar, the schemas of farm-specific data warehouses vary due to different machines, tracking devices, and screening programs in place. The data sources for milk quality are, on the one hand, milking parlors that automatically analyze milk contents; the accuracy and kinds of measures vary by vendor. On the other hand, Dairy Herd Improvement Associations conduct laboratory analyses of the produced milk. Food quantity, contents, and individual intake may be precisely tracked by feeding machines or estimated. Movement data are captured using earmarks and trackers of different vendors, some vendors being able to classify movement type and capture rumination activity. Some dairy farms also follow screening programs for different sicknesses.

The multidimensional models developed for the agriProKnow project consist of fact classes for body condition, blood samples, feeding, fresh cow parameters, climate data, milk production, calvings, and animal activity. The fact classes have the animal and time dimensions. The fact class for the tracking of activity and climate data also has a function area. The animal and time dimensions functionally determine the farm where a fact occurred as well as the lactation cycle and days in milk of the animal. In order to avoid multiple arcs — animals may change farms, have multiple lactation cycles — we represent these dependent attributes as separate dimensions, thus denormalizing the fact tables. The farm-specific data warehouses are then constructed by customization of multidimensional reference models. Individual farms may deselect specific measures, for example, milk quality data supplied by milking parlors, food quantity data supplied by automated feeding machines, blood test data for individual sicknesses, and activity tracking data. Individual farms may also deselect specific levels, for example, capturing climate data at a coarser time granularity. Furthermore, redefinitions of predicates are also common, with different farms potentially employing different definitions of the concept of *active cow* or division of a lactation cycle into the different lactation phases.

With dairy farm managers hardly being experts in data analysis, the definition of analysis situations and analysis graphs considerably facilitates the task of dairy farm managers. Dairy farm managers are spared the task of formulating complex SQL queries and may follow instead analysis graphs. Commercial dairy herd management software, for example, DairyComp,[e] provides related query features with a focus on operational data. The representation of many-to-many relationships as separate dimensions in the multidimensional models facilitates the definition of analysis situations. The agriProKnow project extends BIRD analysis situations with parameterized predicates — subqueries, essentially — and a `HAVING` clause. Both extensions neatly fit into the BIRD approach and are minor extensions.

4.3. *Proof-of-concept prototypes*

A proof-of-concept prototype[28] demonstrates how modelers may use Indyco Builder, BaseX, and XQuery for the design, management, and customization of BIRD reference models. Indyco Builder, a CASE tool for DFM models, then serves as a modeling component for multidimensional reference models. Using comment fields, additional information such as the mandatory nature of levels is incorporated in the DFM models. Indyco Builder employs XML as serialization format for DFM models. Customizations are defined in separate catalogs stored in an XML database. XQuery functions apply these customizations to the DFM models. The resulting DFM files are opened in Indyco Builder which can then generate the logical schema from the customized DFM model. Similarly, analysis graphs and analysis graph customizations are defined in XML and transformed into SCXML by XQuery functions.

A visual editor,[29] implemented in MetaEdit+,[f] facilitates the design of analysis graphs. The visual editor autocompletes the target analysis situation of a navigation step in line with the associated operations. The corresponding execution environment, implemented in Java, allows for the execution of analysis processes modeled with analysis graphs. When executing the analysis processes, the execution environment sends SQL queries that correspond to the current analysis situation and the current bindings of the variables to the database. The execution environment visualizes the result of the queries using pie charts and bar charts.

5. Related Work

In this section we review related works and highlight the similarities, and differences, between BIRD and related works as well as our previous work. First, we discuss the related works in modeling for data warehousing and OLAP. Second, we discuss reference modeling in general followed by reference modeling for data warehousing and OLAP. Finally, we review further related works concerning schema adaptation and evolution in data warehousing and OLAP.

[e]http://web.vas.com/en/Products/Detail/4570
[f]http://www.metacase.com/products.html

5.1. *Modeling for data warehousing and OLAP*

Apart from the reference modeling features, BIRD is a coherent set of simple techniques for conceptual modeling and model-driven development for data warehousing and relational OLAP, with a focus on representing calculated measures and business concepts. In this regard, BIRD builds on our experience from the Semantic Cockpit project[7] which developed an ontology-based framework for interactive data analysis. Apart from fostering reuse through reference modeling, the overall goal in the design of BIRD is a simplification of the semCockpit approach, both with regard to employed technology and with regard to the complexity of the modeling language and models.

Multidimensional modeling in BIRD largely builds on the dimensional fact model,[10] although most features are present in other approaches as well. MultiDimER[18] supports different kinds of hierarchies, for example, nonstrict hierarchies with many-to-many relationships between levels, a feature also present in later elaborations of the DFM.[20] Trujillo *et al.*[30] present an object-oriented modeling approach for data warehouses based on UML; our paper employs UML for the representation of the metamodel rather than the actual reference models themselves. The Common Warehouse Metamodel[31] (CWM) is a comprehensive, but also heavyweight, standard for the specification of data warehouse schemas. The CWM could serve as the fundamental for the representation of BIRD's dimensions, fact classes, and analysis situations especially for exporting them to OLAP engines such as Mondrian.[g]

Procedures for the derivation of relational implementations from conceptual models complement data warehouse modeling approaches. Golfarelli and Rizzi[32] propose a method for logical schema design based on DFM models, Vaisman and Zimányi[33] describe a method for MultiDimER models. Based on these procedures, CASE tools may automate parts of the implementation. Battaglia *et al.*[34] describe QBX, a CASE tool with a graphical DFM editor and automatic generation of relational star and snowflake schemas from DFM models. Indyco Builder, a commercial CASE tool, provides similar features. The BIRD approach may serve as basis for extending such CASE tools with analysis process and reference modeling. The BIRD approach, similar to QBX and Indyco Builder, focuses on relational OLAP. Some BIRD elements contain SQL fragments which are used in the generated SQL queries, which are related to UML/P[35] where UML diagrams are enriched with Java code.

The semCockpit approach[7] introduces dimensional and multidimensional concepts[36] specified in a proprietary language then translated simultaneously into SQL for querying and into OWL for subsumption checking. Concepts in semCockpit are similar to predicates in BIRD where SQL query fragments represent predicate expressions, which are then employed in `WHERE` clauses of generated SQL

[g]http://mondrian.pentaho.com

queries, thus avoiding additional complexity arising from the introduction of a new language.

The conceptual modeling of measures and KPIs has only recently received increased research interest. Maté *et al.*[26] employ Semantics of Business Vocabulary and Business Rules (SBVR) for the definition of atomic and composite KPIs which resemble (aggregations of) base and calculated measures, respectively, in BIRD; SBVR representation of KPIs are then translated[26] into OCL4OLAP (see below). The conceptual modeling of KPIs[26,27] also concerns their organizational context (e.g. goals and organizational roles associated with the KPI), which is outside the scope of the core BIRD approach. The semCockpit approach[7] comes with its own language constructs for the definition of complex measures, where the definition of a complex measure also incorporates a single aggregation operation (e.g. having aggregated measures totalPlannedCosts and avgPlannedCosts). With respect to the semCockpit approach, BIRD simplifies the definition of complex measures, first, by employing SQL for the specification of the calculation expression and, second, by leaving the specification of the aggregation operation to the definition of the analysis situation.

5.2. *Modeling of analytical queries and processes*

Analysis situations and graphs for the conceptual modeling of reusable and parameterizable analytical queries and processes have been introduced in the semCockpit project, in both a simple variant[15] and a fully-elaborated[7] that focuses on comparative data analysis. BIRD builds on a consolidated lightweight variant[16] of analysis situations and graphs that is well-suited for modeling more traditional OLAP sessions, with a specific focus on customization.

Conceptual modeling of analytical queries or views has not received much attention so far. The OCL4OLAP approach[37] employs OCL for the definition of OLAP queries as part of conceptual multidimensional models; the OCL expressions can subsequently be translated to SQL queries or MDX expressions. MetaMIS for Reporting[38] has investigated the conceptual modeling of reports. DFM[10] comes with a simple textual notation for expressing queries over a multidimensional schema. BIRD's analysis situations generalize DFM query expressions by allowing, for each element of the conceptual query specification, variables that are assigned specific values at runtime.

Analysis graphs are inspired by WebML[39] (meanwhile promoted OMG standard as IFML[40]) which describes how data and their relationships may be traversed. The navigation model of WebML is a graph of units and links. A unit represents an object or set of objects retrieved by a parameterized SQL query associated with the unit. A link relates the objects of source and target units. Changes in the parameters of the source unit and, thus, the represented object(s), propagate to the target unit through information transported by the link, binding parameters of target unit to the source unit, thereby leading to changes of the objects represented in the target unit; Changes are either automatic or dependent on user input. Similarly, an analysis situation in analysis graphs represents a parameterized SQL query. A navigation step between analysis situations binds parameters of the target situation to the source situation, potentially depending on user input.

Trujillo *et al.*[41] employ UML state machine diagrams to represent valid execution orders of operations on OLAP cubes. Trujillo *et al.* propose the modeling of a separate state machine diagram for each dimension of an OLAP cube. In these diagrams the states represent the individual dimensions at different roll-up granularities, transitions represent changes in granularity and slice conditions. The BIRD approach also employs a state machine variant, namely State Chart XML, for the representation of executable analysis graphs. Analysis graphs, however, model transitions between states that represent analysis situations. Each analysis situation is characterized by slice and dice conditions as well as a roll-up granularity for each dimension of the corresponding OLAP cube. Rather than specifying OLAP operations for each dimension independently, analysis graphs specify OLAP operations for analysis situations characterized by parameters affecting several dimensions. Analysis graphs aim at the modeling of best-practice analysis processes as opposed to a focus on requirements specification for the conception of OLAP systems.

As opposed to analysis graphs, which support proactive modeling of best practice and useful analysis processes, other works[42,43] follow a sort of process mining approach, analyzing previous analysis sessions in order to recommend an analysis process. Analysis graphs may also be complemented with a representation of the interactive dynamics for visual analysis.[44] Other work[45] has proposed a multidimensional algebra which allows for the description of analytical sessions, providing operators similar to the navigation operators in analysis graphs. Selection corresponds to the slice conditions in analysis situations that can be expressed by dimensional, multidimensional, and measure predicates. Analysis situations in BIRD also feature a roll-up and projection component. The changeFactClass in BIRD analysis situations is similar to a base change. Multidimensional algebras are orthogonal to analysis graphs: The set of navigation operators can be extended with existing operators from the literature. Note that in this paper we only present a simplified version of analysis graphs for reference modeling; future work will present more advanced concepts such as history, loops, backtracking, and drill-across operations.

Other work[46] employs BPMN for modeling ETL processes, in a similar way business intelligence tools employ proprietary languages, for example, Pentaho Data Integration,[h] for the model-driven development of ETL processes. Modeling ETL processes is outside the scope of this paper since we assume that ETL processes are highly dependent on the specific IT infrastructure and thus hard to generalize in reference models.

[h]http://community.pentaho.com/projects/data-integration/

5.3. *Reference modeling*

Reference modeling has received a certain amount of research interest as a reuse mechanism in business engineering and business process management, with the ARIS House of Business Engineering[47] as a notable outcome. Reference modeling is about utilizing "business knowledge contained in reference models for the construction of specific information models",[48] it is "motivated by the Design by Reuse paradigm".[49] Thomas provides an overview[50] of the different meanings of the term 'reference model' in the information systems literature. The common denominator of these meanings, following Fettke and vom Brocke,[51] is that reference models are models that are (intended to be) reused for the construction of specific models, and, as opposed to metamodeling, reference model and specific models are on the same modeling level.

Becker *et al.*[52] distinguish configuration and generic adaptation mechanisms for the derivation of specific models from reference models. In *configurable reference modeling*, the different possible configurations are already encoded in the reference model. Configurable event-driven process chains[49] are a notable example of configurable reference modeling languages. Generic adaptation mechanisms are *instantiation*, *specialization*, and *aggregation* of (parts of) reference models, as well as conclusion by analogy.[53] BIRD supports generic adaptation through instantiation, specialization, and aggregation. Analysis graphs are adapted through instantiation by replacement of variables with constants. Analysis graphs and multidimensional reference models are adapted through specialization by addition and omission of model elements, through aggregation by integration of model elements originally designed for other reference models.

There exist different approaches to utilize reference models for the development and deployment of company-specific software. First, requirements engineers and system modelers use reference models as starting point for the creation of company-specific conceptual models which may subsequently guide the design and implementation process.[3] Second, large enterprise software products, such as SAP ERP, implement reference models and are documented by reference models[54] which help in customizing the software for company-specific settings. Such software products can typically be customized to the specific needs of a company by setting parameters, deselecting features, and extending the software at predefined extension points. In a wider sense, the customizable software can itself be regarded as a kind of reference model (the term reference model subsumes software frameworks[51]). Third, in a model-driven setting, the implementation or the configuration of the system is generated from customized reference models.[55] BIRD follows the latter, model-driven-development approach.

5.4. *Reference modeling for data warehousing and OLAP*

In data warehousing, reference modeling aims at facilitating the conceptualization and development of BI solutions.[3–5] In particular, the H2 metamodeling toolset has been employed for configurative reference modeling for data warehousing.[56] A rule-based approach allows reference modelers to define different variants of conceptual domain models. Configuration parameters govern the selection of these variants in particular situations. Depending on the values of parameters at customization time, the tool generates the specific conceptual domain model. H2 for Reporting (H2fR) is a language based on the H2 toolset for the conceptual modeling of reporting requirements with a special focus on ensuring compliance with regulatory requirements.[6]

Orthogonal to the configurative approach of H2, BIRD builds on generic adaptation mechanisms, namely instantiation, specialization, and aggregation of (parts of) reference models. Rather than anticipating various situations, the reference modeler provides a set of model elements which, at customization time, are quickly adapted through additions, omissions, and redefinitions of model elements. Both reference model and customization are rapidly developed due to a comprehensible amount of customization possibilities, making it especially appropriate for the use in SMEs.

Another major difference between BIRD and the H2 approach to data warehouse reference modeling is the level of abstraction. H2 aims at conceptual domain modeling for data warehousing and reporting, which is a precursor to the concrete design and implementation of the data warehouse and reporting systems. As opposed to H2, BIRD aims primarily at model-driven development: SQL tables, dynamic SQL views, and XML-based workflow models are directly generated from customized multidimensional models and analysis graphs.

6. Conclusion

BIRD is a lightweight reference modeling approach for multidimensional models and analysis processes that builds on widely-available standard technology. Key performance indicators and predicates, which formalize business terms, are defined using SQL. Customized reference models translate into star schema tables, SQL queries, and State Chart XML documents.

Design-science research is an iterative search process,[9] each iteration leading to a design artifact which represents an approximate solution to the investigated research problem. We identify the following future research directions for the BIRD approach to multidimensional reference modeling:

- Reference modeling of guidance and judgment rules: In the semCockpit project, guidance and judgment rules[57] complement analysis graphs, providing hints to analysis in the course of the analysts. Guidance and judgment rules could be subject to redefinitions in the spirit of predicates and calculated measures.
- Reference modeling of events: The business events that trigger and instantiate specific analysis graphs could also be incorporated into the reference model. As opposed to active data warehouses, where events in analytical systems trigger

actions in operational systems,[58] analysis graphs are processes in analytical systems triggered by actions in operational or analytical systems.

- Techniques to associate BIRD's data analysis models with their organizational context in the line of Strecker[27] and Maté[26]: The organizational context could incorporate explanations of the semantics of customizations, indicating the rationale between specific deselections, additions, or redefinitions of model elements in the context of the organization.
- Compliance criteria checking: Techniques that allow reference modelers to specify more fine-grained compliance criteria could be useful in areas where compliance with legal regulations is of critical importance.

Acknowledgments

This work was partially conducted within the agriProKnow project funded by the Austrian Research Promotion Agency (FFG Grant 848610).

References

1. P. Scholz, C. Schieder, C. Kurze, P. Gluchowski and M. Böhringer, Benefits and challenges of business intelligence adoption in small and medium-sized enterprises, in *Proc. 18th European Conf. Information Systems* (2010).
2. C. M. Olszak and E. Ziemba, Critical success factors for implementing business intelligence systems in small and medium enterprises on the example of Upper Silesia, Poland, *Interdiscip. J. Inf. Knowl. Manage.* **7** (2012) 129–150.
3. J. Becker and R. Knackstedt, Referenzmodellierung im data-warehousing — State-of-the-art und konfigurative Ansätze für die Fachkonzeption, *Wirtschaftsinformatik* **46**(1) (2004) 39–49 [in English: Reference modeling in data warehousing — State-of-the-art and configurative approaches for information models].
4. R. Knackstedt and K. Klose, Configurative reference model-based development of data warehouse systems, in *Proc. 16th Information Resources Management Association Conf.* (2005), pp. 32–39.
5. M. Goeken and R. Knackstedt, Multidimensional reference models for data warehouse development, in *Proc. 9th Int. Conf. Enterprise Information Systems* (2007), pp. 347–354.
6. J. Becker, R. Knackstedt, M. Eggert and S. Fleischer, Fachkonzeptionelle Modellierung von Berichtspflichten in Finanzaufsicht und Verwaltung mit dem H2-Toolset, in *Auf dem Weg zu einer offenen, smarten und vernetzten Verwaltungskultur, Gemeinsame Fachtagung Verwaltungsinformatik: (FTVI) und Fachtagung Rechtsinformatik (FTRI)* 2012, LNI, eds. J. von Lucke, C. P. Geiger, S. Kaiser, E. Schweighofer and M. Wimmer, Vol. 197 (Köllen Druck + Verlag Gmbh, Benn, 2012), pp. 83–94 [in English: Conceptual model of reporting requirements in financial supervision and administration using the H2-toolset].
7. T. Neuböck, B. Neumayr, M. Schrefl and C. Schütz, Ontology-driven business intelligence for comparative data analysis, in *Business Intelligence*, LNBIP, ed. E. Zimányi, Vol. 172 (Springer, Heidelberg, 2014), pp. 77–120.
8. C. Diamantini, L. Genga, D. Potena and E. Storti, Collaborative building of an ontology of key performance indicators, in *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*, LNCS, eds. R. Meersman, H. Panetto, T. S. Dillon,

M. Missikoff, L. Liu, O. Pastor, A. Cuzzocrea and T. Sellis, Vol. 8841 (Springer, 2014), pp. 148–165.
9. A. R. Hevner, S. T. March, J. Park and S. Ram, Design science in information systems research, *MIS Q.* **28**(1) (2004) 75–105.
10. M. Golfarelli, D. Maio and S. Rizzi, The dimensional fact model: A conceptual model for data warehouses, *Int. J. Coop. Inf. Syst.* **7**(2–3) (1998) 215–247.
11. J. Mazón, J. Lechtenbörger and J. Trujillo, A survey on summarizability issues in multidimensional modeling, *Data Knowl. Eng.* **68**(12) (2009) 1452–1469.
12. C. A. Hurtado, C. Gutierrez and A. O. Mendelzon, Capturing summarizability with integrity constraints in OLAP, *ACM Trans. Database Syst.* **30**(3) (2005) 854–886.
13. O. Romero and A. Abelló, A survey of multidimensional modeling methodologies, *Int. J. Data Warehousing Mining* **5**(2) (2009) 1–23.
14. C. Schütz and M. Schrefl, Customization of domain-specific reference models for data warehouses, in *Proc. 18th IEEE Int. Enterprise Distributed Object Computing Conf.* eds. M. Reichert, S. Rinderle-Ma and G. Grossmann (IEEE, 2014), pp. 61–70.
15. T. Neuböck, B. Neumayr, T. Rossgatterer, S. Anderlik and M. Schrefl, Multidimensional navigation modeling using BI analysis graphs, in *Advances in Conceptual Modeling*, LNCS, eds. S. Castano, P. Vassiliadis, L. V. S. Lakshmanan and M. Lee, Vol. 7518 (Springer, 2012), pp. 162–171.
16. T. Neuböck and M. Schrefl, Modeling knowledge about data analysis processes in manufacturing, in *Proc. 15th IFAC/IEEE/IFIP/IFORS Symp. Information Control Problems in Manufacturing* (2015).
17. W3C, State Chart XML (SCXML): State Machine Notation for Control Abstraction — W3C Proposed Recommendation 30 April 2015 (2015), http://www.w3.org/TR/2015/PR-scxml-20150430/.
18. E. Malinowski and E. Zimányi, Hierarchies in a multidimensional model: From conceptual modeling to logical representation, *Data Knowl. Eng.* **59**(2) (2006) 348–377.
19. P. Lane and P. Potineni, *Oracle Database Data Warehousing Guide 12c Release 1* (12.1) (*E41670-08*) (Oracle Corporation, 2014).
20. M. Golfarelli and S. Rizzi, *Data Warehouse Design: Modern Principles and Methodologies* (McGraw-Hill, 2009).
21. Indyco/Iconsulting, Indyco Support Center: Additivity matrix (2014), http://indyco.freshdesk.com/support/solutions/articles/1000152606-additivity-matrix.
22. Zentralverband Elektrotechnik- und Elektronikindustrie, *ZVEI-Kennzahlensystem: ein Instrument zur Unternehmenssteuerung*, 4th edn. (ZVEI, Betriebswirtschaftlicher Ausschuss, 1989) [List of key performance indicators proposed by the German Association of Electrical and Electronic Manufacturers].
23. A. P. C. Chan and A. P. L. Chan, Key performance indicators for measuring construction success, *Benchmarking: An Int. J.* **11**(2) (2004) 203–221.
24. W. Leiderer, *Kennzahlen zur Steuerung von Hotel- und Gaststättenbetrieben*, 2nd edn. (Matthaes, Stuttgart, 1983) [List of key performance indicators for hotels and food and beverage companies].
25. E. Malinowski and E. Zimányi, *Advanced Data Warehouse Design — From Conventional to Spatial and Temporal Applications* (Springer, 2008).
26. A. Maté, J. Trujillo and J. Mylopoulos, Conceptualizing and specifying key performance indicators in business strategy models, in *Conceptual Modeling*, LNCS, eds. P. Atzeni, D. W. Cheung and S. Ram, Vol. 7532 (Springer, 2012), pp. 282–291.
27. S. Strecker, U. Frank, D. Heise and H. Kattenstroth, MetricM: A modeling method in support of the reflective design and use of performance measurement systems, *Inf. Syst. E-Bus. Manage.* **10**(2) (2012) 241–276.

28. C. G. Schuetz, I. Spörl and M. Schrefl, Design, management, and customization of data analysis reference models using Indyco Builder and XQuery, in *Proc. 20th IEEE Int. Enterprise Distributed Object Computing Conf.*: *EDOC Workshops* (2016).
29. C. Horschitz, Prototypische Implementierung eines Werkzeuges zur Modellierung und Ausfhrung von Business Intelligence Analysgraphen, thesis, Johannes Kepler University Linz (2016).
30. J. Trujillo, M. Palomar, J. Gomez and I.-Y. Song, Designing data warehouses with OO conceptual models, *Computer* **34**(12) (2001) 66–75.
31. J. Poole, D. Chang, D. Tolbert and D. Mellor, *Common Warehouse Metamodel Developer's Guide* (Wiley, 2003).
32. M. Golfarelli and S. Rizzi, Methodological framework for data warehouse design, in *Proc. ACM First Int. Workshop Data Warehousing and OLAP* (ACM, 1998), pp. 3–9.
33. A. A. Vaisman and E. Zimányi, *Data Warehouse Systems — Design and Implementation* (Springer, 2014).
34. A. Battaglia, M. Golfarelli and S. Rizzi, QBX: A case tool for data mart design, in *Advances in Conceptual Modeling*: *Recent Developments and New Directions*, LNCS, eds. O. De Troyer, C. Bauzer Medeiros, R. Billen, P. Hallot, A. Simitsis and H. Van Mingroot, Vol. 6999 (Springer, 2011), pp. 358–363.
35. B. Rumpe, *Modellierung mit UML*, Xpert.press Series, 2nd edn. (Springer, 2011) [in English: *Modeling with UML*].
36. B. Neumayr, C. Schütz and M. Schrefl, Semantic enrichment of OLAP cubes: Multidimensional ontologies and their representation in SQL and OWL, in *On the Move to Meaningful Internet Systems*: *OTM 2013 Conferences*, LNCS, eds. R. Meersman, H. Panetto, T. S. Dillon, J. Eder, Z. Bellahsene, N. Ritter, P. De Leenheer and D. Dou, Vol. 8185 (Springer, 2013), pp. 624–641.
37. J. Pardillo, J.-N. Mazón and J. Trujillo, Extending OCL for OLAP querying on conceptual multidimensional models of data warehouses, *Inf. Sci.* **180**(5) (2010) 584–601.
38. S. Seidel, R. Knackstedt and C. Janiesch, Procedure model for the analysis and design of reporting systems — A case study in conceptual modeling, in *17th Australasian Conf. Information Systems* (2006).
39. S. Ceri, M. Brambilla and P. Fraternali, The history of WebML: Lessons learned from 10 years of model-driven development of web applications, in *Conceptual Modeling*: *Foundations and Applications*, LNCS, eds. A. Borgida, V. K. Chaudhri, P. Giorgini and E. S. K. Yu, Vol. 5600 (Springer, 2009), pp. 273–292.
40. R. Acerbis, A. Bongio, M. Brambilla and S. Butti, Model-driven development based on OMG's IFML with webratio web and mobile platform, in *Proc. 15th Int. Conf. Web Engineering* (2015), pp. 605–608.
41. J. Trujillo, J. Gómez and M. Palomar, Modeling the behavior of OLAP applications using an UML compliant approach, in *Advances in Inofrmation Systems*, LNCS, ed. T. M. Yakhno, Vol. 1909 (Springer, 2000), pp. 14–23.
42. J. Aligon, E. Gallinucci, M. Golfarelli, P. Marcel and S. Rizzi, A collaborative filtering approach for recommending OLAP sessions, *Decis. Support Syst.* **69** (2015) 20–30.
43. C. Sapia, On modeling and predicting query behavior in OLAP systems, in *Proc. Int. Workshop DMDW 1999*, eds. S. Gatziu, M. A. Jeusfeld, M. Staudt and Y. Vassiliou, Vol. 19 (CEUR-WS.org, 1999).
44. J. Heer and B. Shneiderman, Interactive dynamics for visual analysis, *Commun. ACM* **55**(4) (2012) 45–54.

45. O. Romero, P. Marcel, A. Abelló, V. Peralta and L. Bellatreche, Describing analytical sessions using a multidimensional algebra, in *Data Warehousing and Knowledge Discovery*, LNCS, eds. A. Cuzzocrea and U. Dayal, Vol. 6862 (Springer, 2011), pp. 224–239.
46. Z. El Akkaoui, E. Zimányi, J. Mazón and J. Trujillo, A BPMN-based design and maintenance framework for ETL processes, *Int. J. Data Warehousing Mining* **9**(3) (2013) 46–72.
47. A. Scheer and M. Nüttgens, ARIS architecture and reference models for business process management, in *Business Process Management*: *Models*, *Techniques*, *and Empirical Studies*, LNCS, eds. W. M. P. van der Aalst, J. Desel and A. Oberweis, Vol. 1806 (Springer, 2000), pp. 376–389.
48. O. Thomas and A. Scheer, Tool support for the collaborative design of reference models — A business engineering perspective, in *Proc.* 39*th Hawaii Int. Int. Conf. Systems Science* (2006).
49. M. Rosemann and W. M. P. van der Aalst, A configurable reference modeling language, *Inf. Syst.* **32**(1) (2007) 1–23.
50. O. Thomas, Understanding the term reference model in information systems research: History, literature analysis and explanation, in *Business Process Mangement Workshops*, LNCS, eds. C. Bussler and A. Haller, Vol. 3812 (Springer, 2006), pp. 484–496.
51. P. Fettke and J. vom Brocke, Referenzmodell, in *Enzyklopdie der Wirtschaftsinformatik Online-Lexikon* (University of Potsdam, 2013), http://www.enzyklopaedie-der-wirtschaftsinformatik.de / wi-enzyklopaedie / lexikon / is-management / Systementwicklung / Softwarearchitektur / Wiederverwendung-von-Softwarebausteinen / Referenzmodell
52. J. Becker, P. Delfmann and R. Knackstedt, Adaptive reference modeling: Integrating configurative and generic adaptation techniques for information models, in *Reference Modeling*, eds. J. Becker and P. Delfmann (Springer, 2007), pp. 27–58.
53. J. vom Brocke, Design principles for reference modeling: Reusing information models by means of aggregation, specialisation, instantiation, and analogy, in *Innovation in Information Systems Modeling*: *Methods and Best Practices*, eds. T. Halpin, J. Krogstie and E. Proper (IGI Global, 2009), pp. 269–296.
54. T. Curran, G. Keller and A. Ladd, *SAP R/3 Business Blueprint*: *Understanding the Business Process Reference Model* (Prentice-Hall, 1998).
55. J. Recker, J. Mendling, W. M. P. van der Aalst and M. Rosemann, Model-driven enterprise systems configuration, in *Advanced Information Systems Engineering*, LNCS, eds. E. Dubois and K. Pohl, Vol. 4001 (Springer, 2006), pp. 369–383.
56. R. Knackstedt, S. Seidel and C. Janiesch, Konfigurative Referenzmodellierung zur Fachkonzeption von Data Warehouse-Systemen mit dem H2-Toolset, in *Data Warehousing*, LNI, eds. J. Schelp, R. Winter, U. Frank, B. Rieger and K. Turowski, Vol. 90 (GI, 2006), pp. 61–82 [in English: Configurative reference modeling for the conceptualization of data warehouse systems using the H2-Toolset].
57. D. Steiner, B. Neumayr and M. Schrefl, Judgement and analysis rules for ontology-driven comparative data analysis in data warehouses, in *Proc.* 11*th Asia-Pacific Conf. Conceptual Modeling*, CRPIT, eds. M. Saeki and H. Kohler, Vol. 165 (Australian Computer Society, 2015), pp. 71–80.
58. M. K. Mohania, U. Nambiar, M. Schrefl and M. W. Vincent, Active and real-time data warehousing, in *Encyclopedia of Database Systems*, eds. L. Liu and M. T. Özsu, (Springer, 2009), pp. 21–26.