Monitoring Strategic Goals in Data Warehouses with Awareness Requirements

Vítor E. Silva Souza
University of Trento, Italy
vitorsouza@disi.unitn.it

Jose-Norberto Mazón
University of Alicante, Spain
jnmazon@dlsi.ua.es

Irene Garrigós
University of Alicante, Spain
igarrigos@dlsi.ua.es

Juan Trujillo
University of Alicante, Spain
jtrujillo@dlsi.ua.es

John Mylopoulos
University of Trento, Italy
jm@disi.unitn.it

ABSTRACT

A data warehouse (DW) system stores data from multiple data sources in integrated form and provides capabilities for monitoring business operations to ensure compliance to strategic goals. As such, DWs constitute a fundamental building block for Business Intelligence (BI) operations. In this paper, we introduce the notion of Awareness Requirements (*AwReqs*) in the requirements analysis and elicitation phase for DWs. In this context, *AwReqs* provide analysts with the means for eliciting and modeling requirements over performance measures (indicators) to appraise the success or failure of strategic goals. To demonstrate the benefit of our approach, we present a typical business example throughout the paper and show how we can establish in the early stages of DW design the adequacy of the design for BI operations.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications; H.2.8 [**Database Management**]: Database Applications

General Terms

Data Warehouse; Requirements; Awareness; Goals; Monitoring; Key Performance Indicators

Keywords

Data Warehouse; Requirements; Awareness; Goals; Monitoring; Key Performance Indicators

1. INTRODUCTION

A data warehouse (DW) is commonly described as an integrated collection of historical data sources in support of decision making that structures information into facts (composed of measures) and dimensions (which are the contexts for analyzing facts) based on multidimensional (MD) modeling [7]. DWs constitute a fundamental bulding block for Business Intelligence (BI) operations. Although the development of the MD model has been traditionally guided by the data sources [6], several approaches [3, 11, 14, 18] advocate a requirements-driven DW design process in order to define a MD model that better agrees with user needs and expectations. For example, a UML profile based on $i^\star$ [20] has been defined in [9] in order to capture strategic goals of an organization, the decisions made to achieve them, and the required information to be analyzed for supporting these decisions. From these information requirements, the MD model of the DW is automatically obtained through model transformations, thus allowing us to structure data in a suitable manner to be analyzed for achieving strategic goals.

Unfortunately, current proposals for engineering DW requirements overlook how the DW should be queried to monitor the decision making process and evaluate the fulfillment of the strategic goals. This evaluation is a crucial issue for measuring the success of an organization and it is commonly carried out by analyzing metrics collected during the execution of business processes (BPs), such as Key Performance Indicators (KPIs), normally defined based on facts and on dimensions of the MD model. Therefore, queries for these measurements should be defined from the very beginning of the development in order to (i) prevent designers from deploying an entire DW which does not meet the decision makers data analysis needs, and (ii) implement them in a coherent and integrated manner with the rest of the DW in the subsequent design stages (as basic queries normally evolve in further analysis queries).

To this aim, in this paper we introduce the notion of Awareness Requirements (*AwReqs*) in the user's requirements analysis and elicitation phase for DWs. *AwReqs* have been proposed in [17], motivated by the use of feedback loops as a generic mechanism for self-adaptation. Here, we use them in the context of DW modeling in order to explicitly specify the different queries over KPIs that allow us to monitor if the strategic goals are being fulfilled.

In this paper, we make the following contributions: (i) include *AwReqs* in the DW's MD model and, for this purpose, we extend the UML profile for $i^\star$ proposed in [9] with new elements for *AwReqs*; (ii) show how *AwReqs* can be used to model monitoring requirements on KPIs in different levels of abstraction; (iii) illustrate a systematic process that starts with high-level, close to natural language requirements (business rules and vocabulary) and ends with low-level rules tailored to a specific DW's implementation; and (iv) provide some experimental evaluation by apply-

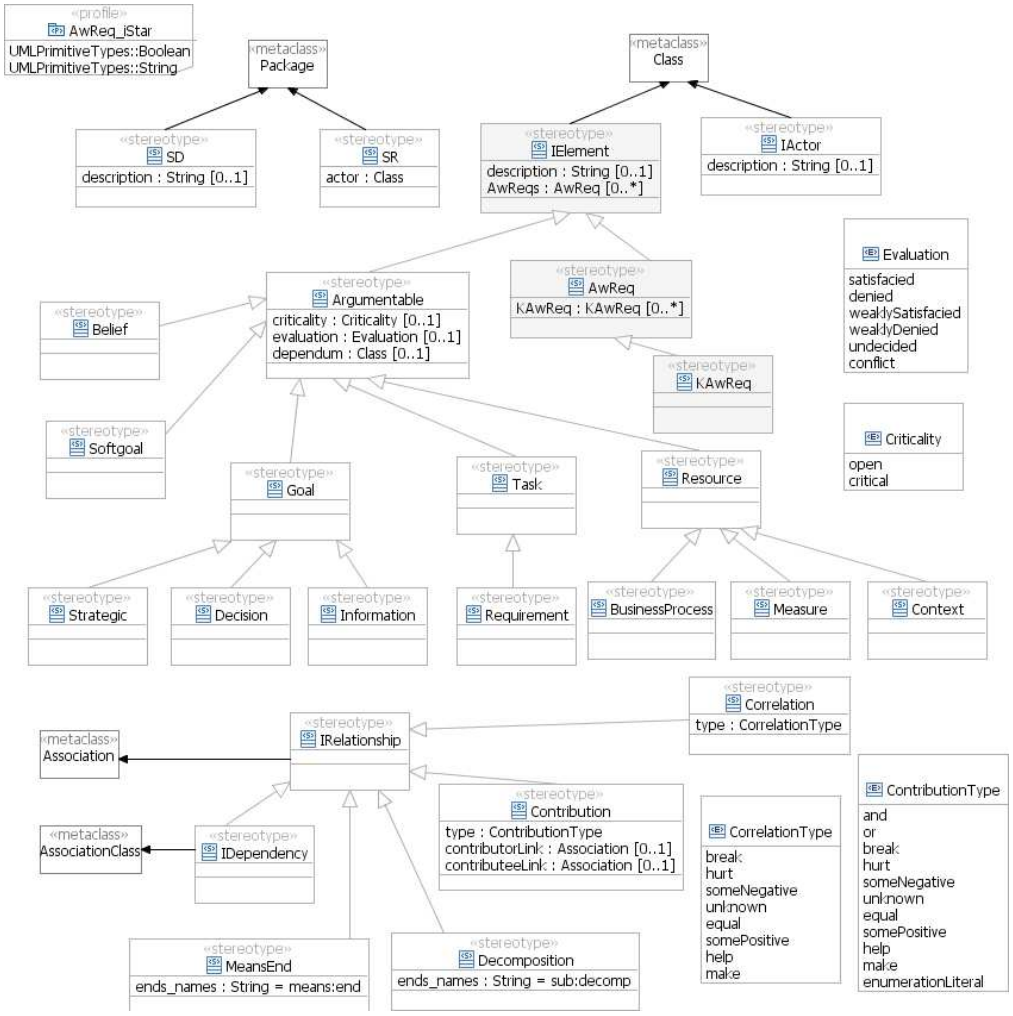

Figure 1: Extended profile for $i^\star$ modeling in the DW domain, including *AwReqs*.

ing the proposed ideas in an experiment, in which the DW requirements for a BP are augmented with monitoring requirements modeled as *AwReqs* and then implemented in the open source BI tool Pentaho[1]. The chosen BP is inspired by one of Pentaho's samples (*SteelWheelSales*) and the results of our experiments are presented throughout the paper as a running example.

The remainder of this paper is structured as follows: section 2 summarizes the work used as baseline for the proposals in this paper; section 3 presents our proposal for the specification of *AwReqs* in the requirements of DWs to monitor strategic goals; section 4 discusses the implementation of monitors based on such a specification; section 5 compares our proposal with the existing state-of-the-art in requirements-oriented DW modeling; finally, section 6 concludes and presents future work.

2. BASELINE

This section summarizes the main building blocks of our approach.

[1]http://www.pentaho.com/

2.1 User's Requirements for Data Warehouses

A requirement analysis stage for DWs aims at obtaining informational requirements of decision makers, which are related to interesting measures of business processes and the context for analyzing these measures. However, decision makers often ignore how to suitably describe information requirements, since they are instead concerned about goals which the DW should help satisfy. Therefore, a requirement analysis phase for DWs should start by discovering goals of decision makers. Afterwards, the information requirements will be more easily discovered from these goals. Finally, information requirements will be related to the required MD concepts, i.e., the measures of the business process or the context for analyzing these measures.

To ease the task of discovering and eliciting goals and requirements for DWs, we classify the different kind of goals that decision makers expect to fulfill with the envisaged DW according to [9]: strategic goals are the main objectives of the business process, such as "increase sales", "increase number of customers", "decrease cost", etc.; decision goals aim to take the appropriate actions to fulfill a strategic goal, for example "define some kind of promotion" or "open new

stores"; finally, information goals are related to the information required by a decision goal if they are to be achieved, examples of which might be "analyze customer purchases" or "examine stocks". Once these goals have been defined, information requirements can be obtained directly from the information goals. The various MD elements, such as facts or dimensions, will be discovered from these information requirements in order to specify the corresponding conceptual MD model of the DW.

Several concepts from DW design have been represented by matching and extending the most convenient concepts of the $i^\star$ framework [20] in order to model the aforementiones hierarchy of goals and their corresponding information requirements. This framework provides mechanisms with which to represent the various actors and their dependencies, and to structure the business goals that the organization wishes to achieve. In order to model goals and information requirements, a UML profile for the $i^\star$ modeling framework has been defined, adapting $i^\star$ to the DW domain. This profile is shown in Figure 1.

In order to define $i^\star$ models for DWs, goals, tasks, and resources are represented as intentional elements for each decision maker. Goals of decision makers are defined by using the `Strategic`, `Decision`, and `Information` stereotypes by specializing the defined `Goal` stereotype; and intentional means-end relationships between them. From information goals, information requirements (`Requirement`) are derived and represented as tasks. Furthermore, the requirement analysis for DWs needs some MD concepts to be added (in the sense of [3]). Thus, the following concepts are added as `Resource` stereotype extensions: business processes related to the goals of decision makers (`BusinessProcess` stereotype), relevant measures related to information requirements of decision makers (`Measure`), and contexts needed for analyzing these measures (`Context`). Also, foreseen relations between contexts of analysis are modeled. For instance, city and country contexts could be related because cities can be aggregated in countries. For modeling these relationships, we use UML's (shared) aggregation relationship.

As a running example for this paper, Figure 2 shows an $i^\star$ model — using the DW UML profile — for a DW of a company in the retail business that wishes to increase its sales each year. In the context of the *SteelWheelSales* business process, this objective is represented by the strategic goal *SG: Increase sales by 10% each year*. In order to accomplish this goal, decision makers included decision goal DG_1*: Launch adequate promotions*.

To fulfill DG_1, the DW will have to provide adequate information. Such objective for the DW is modeled first as information goal $IG_{1.1}$*: Analyze sales*, which is then further refined into information requirements tasks $R_{1.1.1}$*: Quantity of products sold by promotion* and $R_{1.1.2}$*: Quantity of products sold in general*. The latter are then associated with measures and contexts, allowing us to proceed with the implementation of a DW that provides the correct data to be successful in the decision making process, thus achieving *SG*.

Unfortunately, a major drawback of this model is that it only accounts for the static part of the DW[2], i.e., a plan for achieving strategic goals and the required information

[2]The static part of the DW refers to the MD-data structure while the dynamic one refers to the operations that can be done on it (e.g., obtaining some tables, OLAP cubes, reports, etc.).

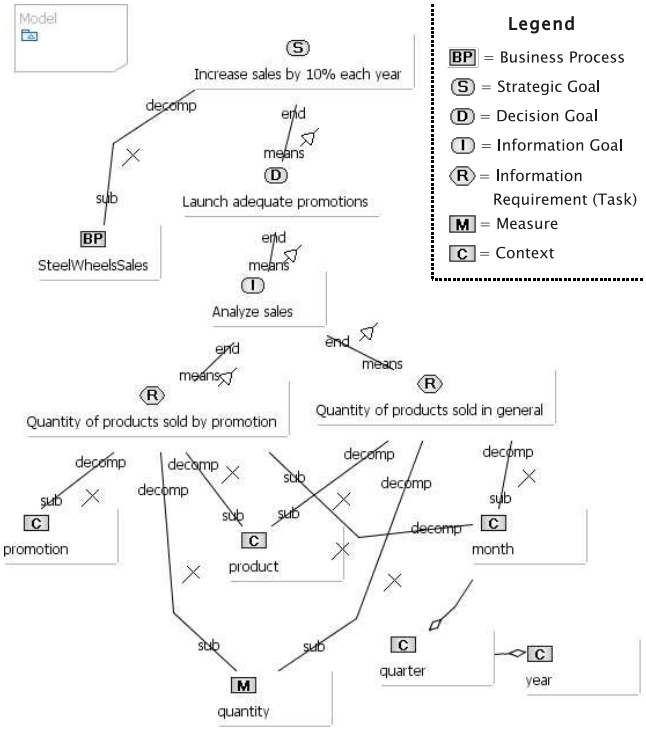

Figure 2: $i^\star$ model for a DW related to the *SteelWheelSales* business process of a retail company.

to be further structured in an MD model. However, requirements should also address how the decision making process could be monitored at runtime through the analysis of specific KPIs, allowing analysts to successfully evaluate the fulfillment of the strategic goals. In order to overcome this limitation, we propose the use of Awareness Requirements (*AwReqs*) to specify monitoring requirements in our DW models. The next subsection introduces the concept of *AwReq*.

2.2 Awareness Requirements

Awareness Requirements (*AwReqs*) have been proposed in [17], motivated by the use of feedback loops as a generic mechanism for self-adaptation. The purpose of such loops is to maintain properties of the system's output at or as close as possible to its reference input. In software systems, the reference input are the requirements, whereas the output is measured at runtime by monitoring it.

We therefore define *AwReqs* as requirements that refer to the success or failure of other requirements. In goal-oriented approaches, an *AwReq* could specify, for instance, that a goal of the system should never fail (i.e., it should always be satisfied). More complex types of *AwReqs* can refer to specific parameters (e.g., requirement R_1 should be satisfied within 10 minutes), aggregate many execution instances and refer to the success rate (e.g., requirement R_2 should be satisfied 95% of the time over one week periods) or even the trend of the success rate over a period of time (e.g., the success rate of requirement R_3 in a month should not be lower than the previous month for two months in a row).

Furthermore, since *AwReqs* are themselves requirements, we allow for the creation of meta-*AwReqs*, meta-meta-*AwReqs*

```
context R0
    def: start : LTL::OclMessage = receivedMessage('start')
    def: end : LTL::OclMessage = receivedMessage('end')
    def: fail : LTL::OclMessage = receivedMessage('fail')
    inv AR0: between(start <> null, end <> null, never(fail <> null))
context R2
    def: weekDurationM : Integer = 7 * 24 * 60;
    def: weekA : LTL::OclMessage = receivedMessage('newWeek')
    def: weekB : LTL::OclMessage = receivedMessage('newWeek')
    def: allS : Set(LTL::OclMessage) = receivedMessages('success')
    def: allF : Set(LTL::OclMessage) = receivedMessages('fail')
    def: wS : Integer = allS->select(m | weekA.timestamp <
            m.timestamp and m.timestamp < weekB.timestamp)->size()
    def: wF : Integer = allF->select(m | weekA.timestamp <
            m.timestamp and m.timestamp < weekB.timestamp)->size()
    inv AR2: between(weekA <> null, weekB <> null and
            ((weekB.timestamp - weekA.timestamp) / 6000 =
            weekDurationM), always(wS / (wS + wF) >= 0.95)
```

Figure 3: Formalization of simple "never fail" and aggregate *AwReqs* using OCL$_{TM}$.

and so forth, while disallowing the specification of circular references. To ease the task of modeling, we have also proposed patterns that simplify *AwReqs* description and a syntax for their graphical representation in the goal model.

After they are specified, *AwReqs* have to be formalized in order to be monitored at runtime. Any formal language could be used, as long as it: (i) allows references to requirements; (ii) can reason over periods of time; and (iii) is supported by the framework that implements the adaptivity. For example, Figure 3 shows the formalization of two of the above *AwReqs*, using OCL$_{TM}$ [16].

As described in [17], a monitoring framework represents requirements as UML classes which are instantiated at runtime and the target system to which these requirements belong is instrumented to send messages (such as `start()`, `end()` and `fail()`) to instances of these requirements representing changes in their states. Given this infrastructure, the *AwReqs* formalized in Figure 3 can be explained: `AR0` states that it should never be the case that requirement `R0` goes into the `failed` state at any point of its life cycle (between `start()` and `end()` messages). `AR2`, on the other hand, processes `success()` and `fail()` messages received in a week period (delimited by calls of method `newWeek()` performed by the framework) and assures that at least 95% of the times requirement `R2` has succeeded.

Therefore, once *AwReqs* are formalized, they can be fed into this monitoring framework that will indicate during system operation when they have succeeded or failed, assuming that the system provides the appropriate log messages. To this aim, in [17] we describe a monitoring framework implemented over the Event Engineering and Analysis Toolkit (EEAT[3]), formerly known as ReqMon [15].

3. MONITORING STRATEGIC GOALS

We would like to augment our DW models with requirements that help assuring the fulfillment of strategic goals. This can be accomplished by specifying monitoring requirements on KPIs that can influence decision and information goals. Such measures would then be monitored at runtime, leading to one of two possible outcomes: (a) through some dashboard-like user interface, decision makers could be informed if the decisions they have taken are indeed fulfilling the organization's strategic goals in time for them to change plans (new decision/information goals) if needed; or (b) the DW system itself could adopt new strategies that would better fulfill the organization's objectives if adaptivity requirements were also provided in the DW's design.

Our focus on this paper, however, is on monitoring only. We propose the use of *AwReqs* (see §2.2) to model such monitoring requirements in our DW models. Going back to the example of Figure 2, we start by stating we do not want the strategic goal *SG: Increase sales by 10% in each year* to fail – $\texttt{AR}_1$: `NeverFail(SG)`. However, adding such an *AwReq* to a strategic goal will only make the system monitor if, at the end of the year, this goal has been accomplished or not. To be useful, our monitors should tell decision makers if there are any problems with their decisions ahead of time. In other words, during the period in consideration (i.e., throughout the year in question), indicators (KPIs) that could tell if we are on the right path to satisfying *SG* should be regularly extracted from the DW in order to be checked.

We should, thus, break down *AwReq* `NeverFail(SG)` into KPI-related *AwReqs* (or *K-AwReqs* for short), which specify the monitoring requirements over given KPIs. The first step towards this, however, is eliciting from stakeholders and domain experts what these indicators are. In our running example, we have elicited four monitoring requirements, namely:

- $\texttt{AR}_{1.1}$ — after every month, the quantity of products sold in that month should be at least 10% greater than the quantity sold in the same month last year;
- $\texttt{AR}_{1.2}$ — after every quarter, the quantity of products sold so far in the year should be at least 10% greater than the quantity of products sold in the same period last year;
- $\texttt{AR}_{1.3}$ — for every month, the quantity of products sold in promotions should account for at least 50% of the total quantity of products sold in the month;
- $\texttt{AR}_{1.4}$ — for every month, the quantity of sales of a given product under a promotion should not decrease in 10% or more the quantity of sales of other products of the same type.

K-AwReqs $\texttt{AR}_{1.1}$ and $\texttt{AR}_{1.2}$ verify if we are indeed increasing the amount of sales, while $\texttt{AR}_{1.3}$ and $\texttt{AR}_{1.4}$ check that, if sales are increasing, this is most likely due to a good decision goal: making promotions to sell more. After eliciting *AwReqs* and *K-AwReqs*, the next step is to formalize them.

3.1 AwReq Formalization

As said in section 2.2, any formal language that allows us to reference the requirements and reason over periods of time can be used to formalize *AwReqs*. *K-AwReqs* refer to data that can be extracted from the DW, thus we need a model of these data before we can write constraints that reference them. For instance, Fig. 4 shows a conceptual MD model for our running example based on the notation of [8]. The model depicts the relationship between facts and dimensions in the DW: a *SteelWheelsSales* fact stores the *quantity* of product sold, which can be analyzed by using the *promotion*, *time*

[3]http://eeat.cis.gsu.edu:8080/

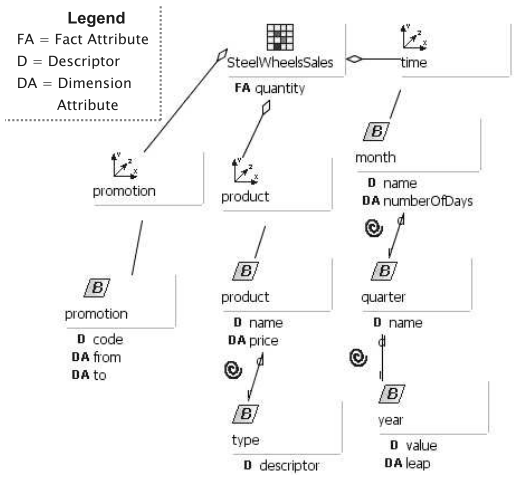

Figure 4: Conceptual MD model of a DW for our running example.

or *product* dimensions. The time dimension has a hierarchy consisting of *month*, *quarter* and *year* aggregation levels. This MD model can be obtained from the $i^\star$ model of Fig. 2 by using the approach proposed in [9].

Although we could use OCL_{TM} for the formalization of *K-AwReqs*, a more suitable language for formalizing *AwReqs* that refer to DW elements is an extension of OCL that includes a set of pre-defined OLAP (On-Line Analytical Processing) operators [13]. OLAP is one of the most popular kinds of applications that can be used to analyze data in the DW, since it allows human analysts to navigate through MD structures in order to access data in a more natural manner. Therefore, OLAP operators can be useful for specifying queries over KPIs according to the *AwReqs*. Although OCL for OLAP[4] does not include temporal logic operators like OCL_{TM}, time periods (years, months, days, etc.) are always considered as relevant dimensions when aggregating and disaggregating data in DWs. Hence, we can refer to information in the DW from different moments in time by slicing and dicing over these dimensions.

Based on the conceptual model of Figure 4, OCL for OLAP can be used to formalize our monitoring requirements. As an example, Figure 5 shows the formalization of $\texttt{AR}_{1.1}$ and $\texttt{AR}_{1.3}$. Unlike *AwReqs*, which refer to requirements and are triggered by log messages coming from the monitored system, *K-AwReqs* have predefined moments in which they need to be checked. In our examples, $\texttt{AR}_{1.1}$, $\texttt{AR}_{1.3}$ and $\texttt{AR}_{1.4}$ should be checked monthly, while $\texttt{AR}_{1.2}$ is checked at the end of every quarter. Therefore, we introduce a `@trigger` annotation in comments to specify when the *K-AwReq* is supposed to be verified, passing a Cron expression[5] as a parameter. Cron's predefined scheduling definitions (e.g., `@monthly`, `@weekly`, etc.) are also allowed. Both *K-AwReqs* shown in Figure 5 are supposed to be checked monthly (on the 1^{st} day of the month at hour 00:00).

3.2 AwReq Implementation

Since our MD models contain *K-AwReqs* written by using OCL for OLAP, they can be directly implemented in any final BI technology platform by using exactly the same existing methods and tools able to generate code from UML/OCL schemas. These methods do not need to be extended to cope with our *K-AwReqs* due to the fact that functions of our OCL for OLAP approach are defined by using standard OCL operations. Furthermore, *K-AwReqs* are defined at the model-level and thus they are technologically independent. A set of rules to deal with the transformation between OCL for OLAP and SQL is proposed in [13]. In this paper we exemplify this transformation by defining some ad-hoc rules for the MultiDimensional eXpressions (MDX) language[6]. This language provides a specialized syntax for managing MD data stored in an OLAP cube. Since we aim to query data, we are interested in the `SELECT` statement in order to retrieve data from a specified cube. A simplified generic `SELECT` statement template is described in Figure 6.

For example, $\texttt{AR}_{1.1}$ can be transformed into two MDX statements according to this template, as follows: `addDimension(e)` corresponds to a ⟨`SELECT query axis clause`⟩ containing an `ON ROWS` statement. Specifically, the expression `e` in this OCL function is mapped to the `Set_Expression` in the ⟨`SELECT query axis clause`⟩. As `sum()` is the default aggregation function in MDX, there is no need for map-

```
let currAmount = grain->addDimension(Product, sum())
    ->sliceAndDice(s | s.time.month = m and s.time.month.year
    = y)->dimensionalProject(SteelWheelsSales::Quantity)
let lstAmount = grain ->addDimension(Product, sum())
    ->sliceAndDice(s | s.time.month = m and s.time.month.year
    = y - 1)->dimensionalProject(SteelWheelsSales::Quantity)
-- @trigger("0 0 1 * *")
inv AR1_1: currAmount >= 1.1 * lstAmount

let promAmount = grain->addDimension(Product, sum())
    ->sliceAndDice(s | s.time.month = m and s.time.month.year
    = y and s.promotion=p)
    ->dimensionalProject(SteelWheelsSales::Quantity)
let totalAmount = grain->addDimension(Product, sum())
    ->sliceAndDice(s | s.time.month = m and s.time.month.year
    = y)->dimensionalProject(SteelWheelsSales::Quantity)
-- @monthly
inv AR1_3: promAmount >= 0.5 * totalAmount
```

Figure 5: *K-AwReqs* $\texttt{AR}_{1.1}$ and $\texttt{AR}_{1.3}$ in OCL for OLAP.

```
SELECT [( <SELECT query axis clause> [ , <SELECT query axis clause>,...n ] )]
FROM <SELECT subcube clause> [ <SELECT slicer axis clause> ]

<SELECT query axis clause> ::= [ NON EMPTY ] Set_Expression ON COLUMNS | ROWS

<SELECT subcube clause> ::= Cube_Name |
    (SELECT [( <SELECT query axis clause> [ ,<SELECT query axis clause>,...n ] )]
    FROM <SELECT subcube clause> <SELECT slicer axis clause>)

<SELECT slicer axis clause> ::= WHERE Tuple_Expression
```

Figure 6: Generic MDX `SELECT` statement template for retrieving data from an OLAP cube.

[4]Despite its name, OCL for OLAP is still applied over UML classes, but adds OLAP operators to "vanilla" OCL.
[5]http://en.wikipedia.org/wiki/CRON_expression

[6]http://www.microsoft.com/msj/0899/mdx/mdx.aspx, last visit: October 24^th, 2011

```
select NON EMPTY {[Measures].[Quantity]} ON COLUMNS, NON EMPTY
{([Product].[All Products])} ON ROWS from [SteelWheelsSales] where
([Time].[All Years].&[Y].&[Q].&[M])
select NON EMPTY {[Measures].[Quantity]} ON COLUMNS, NON EMPTY
{([Product].[All Products])} ON ROWS from [SteelWheelsSales] where
([Time].[All Years].&[Y-1].&[Q].&[M])
```

Figure 7: MDX query for *K-AwReq* $\texttt{AR}_{1.1}$

ping it; `sliceAndDice(e)` corresponds to a ⟨`SELECT slicer axis clause`⟩, and the expression `e` is mapped to the `Tuple_Expression` in MDX; `dimensionalProject(e1::e2)` corresponds, at the same time, to the ⟨`SELECT subcube clause`⟩ (since the expression `e1` in the OCL function is the `Cube_Name` in the MDX statement) and to a ⟨`SELECT query axis clause`⟩ containing an `ON COLUMNS` (since expression `e2` maps to the measure to be analyzed in the MDX statement). The resulting MDX implementation is shown in Figure 7.

3.3 SBVR formalizations of AwReqs

Our approach focuses on giving the decision makers the right mechanisms to specify their KPIs and giving the designer the right mechanisms for formalizing them in OCL using the concept of *AwReqs*. These *AwReqs* can be translated as queries over the MD model in some technology-dependent language as MDX. Monitoring requirements are elicited from stakeholder and domain experts and are then translated into OCL for OLAP by technical people in order to be used in the DW. This process has two shortcomings: (i) requirements are formalized solely in a language that Decision Makers do not understand and, therefore, cannot further contribute; (ii) if stakeholders present changes in the requirements, technical people must once again interpret the ideas presented by the domain experts and translate them into OCL for OLAP. It would be interesting to formalize these requirements in an intermediary language that both technical and business people could understand and that could be systematically or automatically translated into a more formal language. In this way, our approach could be used in the context of a model-driven development process.

The Object Management Group (OMG) has developed the Semantics of Business Vocabulary and Business Rules (SBVR) specification [1]. SBVR has been created to be useful for business purposes, independently of information systems designs. Specifically, it defines a meta-model for documenting the semantics of business vocabulary, facts and rules in a language close enough to a natural language to allow business experts to manage them, and at the same time formal enough (based on predicate logic) to be suitable for being used in a model-driven process. Thus, according to [2], it is specially suited for acting as an intermediate representation between the business users and the IT designers.

It is worth noting that within SBVR there is a user-friendly notation based on Structured English that allows us to express *AwReqs* in natural language. There are four font styles with formal meaning in this language: the term font is used for noun concepts that are part of a vocabulary being used or defined; the name font is used for individual concepts; the ***verb*** font is used for verbs, preposition, or combination thereof; and the *keyword* font is used for linguistic symbols used to construct statements (e.g., quantifications as "each" or "at least one").

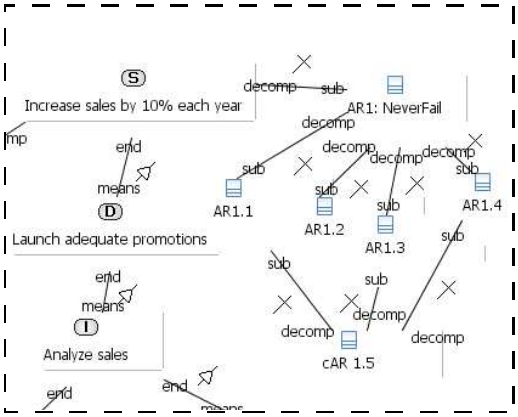

Figure 8: Section of the model shown in Figure 2 with graphical representation of *AwReqs*.

Therefore, in order to bridge the gap between business and IT perspectives, *K-AwReqs* would first be documented in SBVR before being formalized (as in Figure 5). Due to space constraints, we show here the SBVR version of only one *K-AwReq*, $\texttt{AR}_{1.1}$: *it is necessary that* the sum of the quantity of SteelWheelSales of *each* product in month M of year Y ***increases by*** 10% of the sum of the quantity of SteelWheelSales of *each* product in month M of year Y-1.

Once, the *AwReqs* are defined in SBVR the following step towards a model-driven process is to automatically derive the corresponding OCL for OLAP formalizations. However, this is a challenging research task [2] that is out of the scope of this paper.

3.4 Context K-AwReqs

Monitoring KPIs with *K-AwReqs* can alert decision makers of undesirable situations, such as not increasing sales in 10% in the previous month ($\texttt{AR}_{1.1}$) or in the cumulative result of the past months ($\texttt{AR}_{1.2}$). These warnings, however, could be more or less important given the current context of the organization. Such context can also be monitored through analysis of indicators and, therefore, could be included in the requirements for the DW using *K-AwReqs*.

In our running example, suppose that an analysis of the past 3 years shows us that, in average, 80% of the sales of our company happen in the months of November and December due to shopping for the holidays. In this case, if we increased sales in these two months by 12.5%, we would guarantee the 10% sales increase for the whole year even if sales did not increase (nor decrease) in all of the other months. Therefore, if a *K-AwReq* like $\texttt{AR}_{1.1}$ fails, it could be interesting to know if the period that has been analyzed is a critical one or not.

A context *K-AwReq* to monitor this information can be writte in SBVR as follows — $\texttt{cAR}_{1.5}$: *it is necessary that* the sum of the quantity of SteelWheelSales of *each* product in month M of years Y-3 to Y-1 ***is not lower than*** 5% of the sum of the quantity of SteelWheelSales of *each* product of years Y-3 to Y-1.

As it can be seen from this SBVR formalization, Context *K-AwReqs* are just like regular *K-AwReqs*, only differing in their purpose. Regular *K-AwReqs* failures should trigger warnings on dashboards or self-adaptive behavior of the system itself. Context *K-AwReqs* failures, on the other hand,

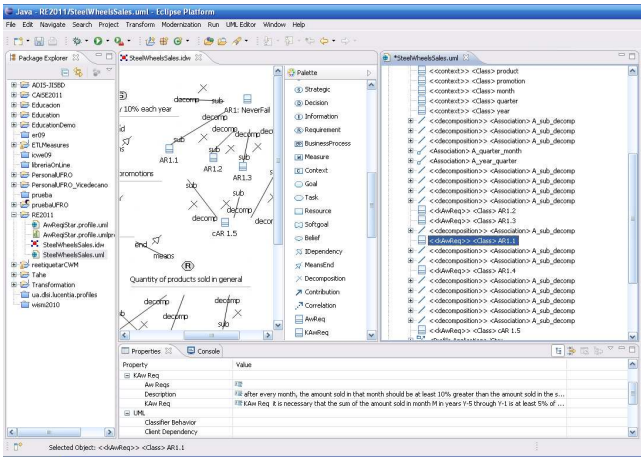

Figure 9: Screen shot of the CASE Tool being used to model *AwReqs* in an MD model.

just indicate that the current context for a given set of regular *K-AwReqs* is not a critical one (in the above example, a $\texttt{cAR}_{1.5}$ failure indicates the current month is traditionally not a high sales month). Decision makers are, thus, advised to consider a *K-AwReq* failure as less important if one or more of its associated Context *K-AwReqs* have failed.

3.5 K-AwReqs in the $i^\star$ profile for DW

In order to graphically represent *AwReqs* and *K-AwReqs* in requirement models for DW, such as the one in Figure 2, we need to extend the $i^\star$ profile for DW that we have previously explained. Figure 1 shows the extended profile (with new and changed meta-classes presented in gray), while an example of the graphical representation of *AwReqs* in the DW requirements model is depicted in Figure 8.

By extending the profile, we could benefit from the transformations proposed in [9] and the CASE tool developed in [5], facilitating the implementation of the DW. A screen shot of the CASE tool being used to create the model of Figure 8 is shown in Figure 9.

4. EXPERIMENTS USING A BI TOOL

Once *K-AwReqs* are defined using the MDX language (like $\texttt{AR}_{1.3}$ in Figure 7), it is possible to insert them into a specific business intelligence tool. It is worth noting that together with the MDX queries, the rest of the MD structures must be implemented. However, this is out of the scope of this paper and the reader should refer to [10] for a detailed explanation of this implementation.

To test the queries generated with our approach in the context of the running example presented in this paper, the Mondrian open-source OLAP server – which is part of the Pentaho BI Suite – was chosen, since it uses MDX as a query language. In this section, it is shown how the MDX derived from one of the *K-AwReqs* – $\texttt{cAR}_{1.5}$ – was validated.

The aim of $\texttt{cAR}_{1.5}$ is to *query* sales of products during different periods of time in order to know periods that are critical. Therefore, a couple of MDX *queries* are executed in Mondrian: the first one, shown in Figure 10a, retrieves sales of products in a certain month for three consecutive years, whilst the second one, in Figure 10b, considers these three years as a whole.

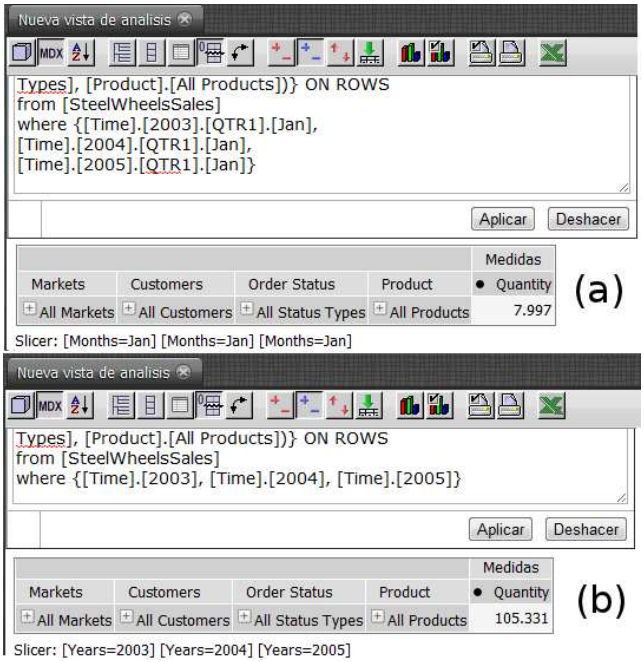

Figure 10: Executing MDX query for $\texttt{cAR}_{1.5}$.

Both figures show the Pentaho user console with the MDX *query* being executed. Shaded in the bottom we can see the result of the *queries*, namely the quantity of sales during the specified periods. Other MDX *queries* generated from the *K-AwReqs* presented in this paper were tested in a similar fashion. The idea is that these *queries* could be later integrated into a dashboard to inform decision makers how their decisions are affecting the business process.

5. RELATED WORK

Requirement analysis is a crucial task in early stages of the DW development. However, only few approaches in this field have considered this task. In [19], a method is proposed in order to both determine information requirements of DW users and match these requirements with the available data sources. The work in [12] presents the DW requirements definition (DWARF) approach that adapts the traditional requirements engineering process for requirements definition and management of DWs. The approach described in [14] focuses on describing a requirement elicitation process for DWs by identifying goals of the decision makers and the required information that supports the decision making process. Finally, in [4], the authors present a goal-oriented framework to model requirements for DWs, thus obtaining a conceptual MD model from them, in which data sources are used for shaping hierarchies, while user requirements are used to choose facts, dimensions and measures.

Unfortunately, these approaches present one common drawback, since they overlook business user's requirements about how the DW should be queried to monitor the decision making process and evaluate the fulfillment of the strategic goals. Consequently, metrics used for monitoring purposes, such as KPIs, are overlooked in these requirement engineering approaches for DWs. To overcome this situation, in this paper we have introduced the notion of *AwReqs* in the user's re-

quirements analysis and elicitation phase for DWs with the aim of explicitly specifying the different queries over KPIs that allow us to monitor if the strategic goals are being fulfilled. The approach is not a BI tool *per se*, but is useful for designing BI solutions on top of existing BI tools.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced the notion of Awareness Requirements in the requirements analysis and elicitation phase for Data Warehouses in order to model requirements over KPIs, with the purpose of generating monitors for the success of the organization's strategic goals and warning decision makers if their decisions may not have been adequate.

We proposed a model-driven approach, in which: (i) *AwReqs* are included in the DW's MD model, broken down into *K-AwReqs*; (ii) *K-AwReqs* are elicited from stakeholders and written in OCL for OLAP, which formalizes the *K-AwReqs*; (iii) MDX queries are derived from the *K-AwReqs*'s formalizations in order to be used in a specific BI tool. The approach has been validated with an experiment, presented throughout the paper as a running example, that used the OLAP server Mondrian and its sample business process SteelWheelSales.

As immediate future work, we plan to obtain better validation results by performing a case study (preferably using the aforementioned tools) in order to evaluate the benefits of strategic goal monitoring and the effectiveness of our approach in a real-world scenario. Another direction we find interesting is to further apply research on self-adaptive systems in the area of DW, developing a framework that implements a feedback controller that not only detects failures, but diagnoses and compensates them and reconciles the business process behavior to satisfy the strategic goals.

7. ACKNOWLEDGMENTS

This work has been partially supported by MESOLAP (TIN2010-14860), from the Spanish Ministry of Science and Innovation, by the DADS (PBC-05-012-2) project, from the Castilla-La Mancha Ministry of Education and Science (Spain) and by the ERC advanced grant 267856 "Lucretius: Foundations for Software Evolution" (unfolding during the period of April 2011 – March 2016, www.lucretius.eu).

8. REFERENCES

[1] OMG: Semantics of Business Vocabulary and Rules (SBVR) Specification, v. 1.0 (formal/08-01-02), 2008, `http://www.omg.org/spec/SBVR/1.0/`, 2008.

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

[3] P. Giorgini, S. Rizzi, and M. Garzetti. Goal-oriented requirement analysis for data warehouse design. In *DOLAP '05: 8^{th} international workshop on Data warehousing and OLAP*, pages 47–56. ACM, 2005.

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

[5] O. Glorio, J. Pardillo, J.-N. Mazon, and J. Trujillo. DaWaRA: An Eclipse Plugin for Using i* on Data Warehouse Requirement Analysis. In *RE '08: 16^{th} IEEE International Requirements Engineering Conference*, pages 317 –318, sept. 2008.

[6] W. H. Inmon. *Building the Data Warehouse, 4^{th} Edition*. Wiley, Oct. 2005.

[7] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2^{nd} Edition*. Wiley, Apr. 2002.

[8] S. Luján-Mora, J. Trujillo, and I.-Y. Song. A UML profile for multidimensional modeling in data warehouses. *Data & Knowledge Engineering - Special issue: ER 2003*, 59:725–769, December 2006.

[9] J.-N. Mazón, J. Pardillo, and J. Trujillo. A Model Driven RE Approach for Data Warehouses. In *RIGiM '07: First International Workshop on Requirements, Intentions and Goals in Conceptual Modeling*, 2007.

[10] J.-N. Mazón and J. Trujillo. An MDA approach for the development of data warehouses. *Decision Support Systems*, 45:41–58, April 2008.

[11] J.-N. Mazón, J. Trujillo, M. Serrano, and M. Piattini. Designing data warehouses: from business requirement analysis to multidimensional modeling. In *REBNITA '05: 1^{st} International Workshop on Requirements Engineering for Business Need and IT Alignment*, pages 44–53, Paris, France, 2005.

[12] F. R. S. Paim and J. F. B. de Castro. DWARF: An Approach for Requirements Definition and Management of Data Warehouse Systems. In *RE '03: 11^{th} IEEE International Conference on Requirements Engineering*, pages 75–84, 2003.

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

[14] N. Prakash, Y. Singh, and A. Gosain. Informational Scenarios for Data Warehouse Requirements Elicitation. In *ER '04: 23^{rd} International Conference on Conceptual Modeling*, pages 205–216. Springer Berlin / Heidelberg, 2004.

[15] W. N. Robinson. A requirements monitoring framework for enterprise systems. *Requirements Engineering*, 11(1):17–41, Mar. 2006.

[16] W. N. Robinson. Extended OCL for Goal Monitoring. In *Ocl4All '07: 7^{th} International Workshop on Ocl4All: Modelling Systems with OCL*. Springer, 2007.

[17] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos. Awareness Requirements for Adaptive Systems. In *SEAMS '11: 6^{th} International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, Honolulu, USA, 2011. ACM.

[18] R. Winter and B. Strauch. A Method for Demand-Driven Information Requirements Analysis in Data Warehousing Projects. In *HICSS '03: 36^{th} Annual Hawaii International Conference on System Sciences*, page 231a. IEEE, 2003.

[19] R. Winter and B. Strauch. Information requirements engineering for data warehouse systems. In *SAC '04: The 2004 ACM symposium on Applied computing*, SAC '04, pages 1359–1365. ACM, 2004.

[20] E. Yu. Social Modeling and i*. In A. Borgida, V. Chaudhri, P. Giorgini, and E. Yu, editors, *Conceptual Modeling: Foundations and Applications*, volume 5600 of *Lecture Notes in Computer Science*, pages 99–121. Springer Berlin / Heidelberg, 2009.