

A UML Profile for Representing Business Object States in a Data Warehouse^{*}

Veronika Stefanov and Beate List

Women's Postgraduate College for Internet Technologies
Vienna University of Technology
{stefanov, list}@wit.tuwien.ac.at

Abstract. Data Warehouse (DWH) systems allow to analyze business objects relevant to an enterprise organization (e.g., orders or customers). Analysts are interested in the states of these business objects: A customer is either a potential customer, a first time customer, a regular customer or a past customer; purchase orders may be pending or fulfilled. Business objects and their states can be distributed over many parts of the DWH, and appear in measures, dimension attributes, levels, etc. Surprisingly, this knowledge – how business objects and their states are represented in the DWH – is not made explicit in existing conceptual models. We identify a need to make this relationship more accessible. We introduce the *UML Profile for Representing Business Object States in a DWH*. It makes the relationship between the business objects and the DWH conceptually visible. The UML Profile is applied to an example.

1 Introduction

Data Warehouse (DWH) systems are used by decision makers for performance measurement and decision support. The data offered by a DWH describes business objects relevant to an enterprise organization (e.g., accounts, orders, customers, products, and invoices) and allows to analyze their status, development, and trends. *Business objects*, also known as domain objects, represent entities from the “real world enterprise” and should be recognizable to business people, as opposed to implementation objects (e.g., menu items, database tables).

Business objects can be characterized by the states they have during their lifecycle. For example, a customer can have different states: There are potential customers, first time customers, regular customers, customers who pay their bills on time, fraudulent customers, high volume customers, past customers, etc.

In many organizations today, DWHs have grown over time and become large and complex. Business objects and their states can be found all over the DWH: Data relevant to analyzing e.g. customers may be distributed over many parts of

^{*} This research has been funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

the DWH, and the different states of the customer may appear in many different ways, i.e. as measures, dimensional attributes, levels, etc. A business object can easily have 20 states, which might be hidden in just as many facts or even more: Potential customers in a contacts fact of the marketing data mart, regular customers in a sales fact, customers who pay on time in a payment transaction fact, etc.

Surprisingly, this knowledge - how business objects and their states are represented in the data warehouse - is not made explicit in existing conceptual models. When new analysis requirements appear, it is often difficult, time consuming and costly to find out whether the information about a certain state of a business object is already contained somewhere in the DWH, or whether the data model of the DWH has to be extended or changed.

Our goals therefore are to

- make the relationship between the business objects that the users want to analyze and the DWH visible and accessible
- show where the business objects and their states can be analyzed in the DWH
- offer a new perspective on DWHs, which emphasizes business objects and their states instead of solely fact and dimension tables
- show how the business object states relate to the data model

We achieve these goals by introducing the *UML¹ Profile for Representing Business Object States in a DWH* (Section 4), which makes the connection between data warehouses and business object states visible. State machines (Section 2) describe how objects change states in reaction to events, and are suitable for capturing business logic. We relate state machines of business objects to DWH elements and use them as a new viewpoint on data warehouses. Additionally, using the UML Profile, we define 14 correspondence patterns between state machines and DWH elements (Section 5).

The contributions of the UML Profile for Representing Business Object States in a DWH along with the correspondence patterns are:

- It provides a straightforward way to make visible where a business object such as a customer is available in the DWH and can be analyzed, and how its various states correspond to facts, dimensions and measures.
- Through the business objects, it offers a bigger picture as it links the needs of the business organization to the DWH that stores data on business objects for analytical purposes.
- The Profile for Representing Business Object States in a DWH allows modeling on several levels of detail and thus enables the modeler to choose the right level of detail for different purposes or target audiences. A high level overview model shows only the whole business objects and how they are related to facts and dimensions, where as a more detailed model can show measures, dimension attributes and hierarchy levels for the individual states of the object.

¹ Unified Modeling Language

- By extending standard UML 2.0 state machines, the UML profile offers reuse of a well-known notation as well as tool reuse, avoiding costs of learning a new notation or additional tools.
- The models allow locating business object states in the DWH, and thus recognizing cases where business object states are not available at all, which may indicate business requirements that are not yet addressed. If the model shows that a business object state is available at several locations at the same time, it can be checked whether this was a deliberate design decision or whether this indicates a problematic situation.
- Together with the correspondence patterns the models can support the design phase of a DWH project, as they provide hints on possible facts, measures and dimensions that can be derived from the business object states.

2 UML State Machines

We use UML 2.0 state machines to model business object states. State machines in general describe “the possible life histories of an object” [1]. A state machine comprises a number of states, interconnected by transitions. Events trigger the transitions. There may be several transitions for one event, and they may be guarded by conditions (see Fig. 1).

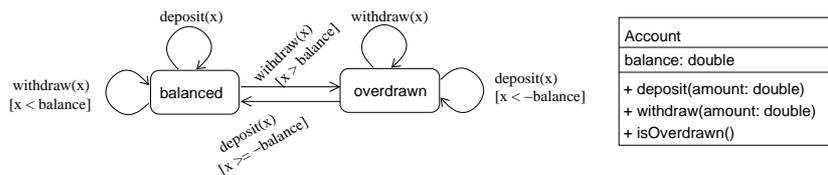


Fig. 1. State machine for Account objects

State machines are used for example in Software Engineering to achieve higher quality software. During the analysis phase of a software development process, state machines for the main business objects (account, order, etc.) are modeled. Such a state machine cannot be transformed directly to code, but is very useful for designers as it allows to recognize “illegal” or conceptually impossible state transitions and thus assess correctness of the final software product. The UML Profile for Representing Business Object States in a DWH presented in this paper aims at bringing the power of state machines to Data Warehousing.

Figure 2 gives an overview of state chart elements and their use in UML 2.0. States may be nested in other states (c), and the so-called sub-machines can be divided into regions (d). This allows access to the model on various levels of detail. Figure 3 shows an example state machine. The object modeled here is the contract between a telecommunications provider and the user of a post-paid mobile phone.

Before it is signed, the contract is in the state “potential”. After the customer has signed it, it is registered and a credit check is performed. During this phase,

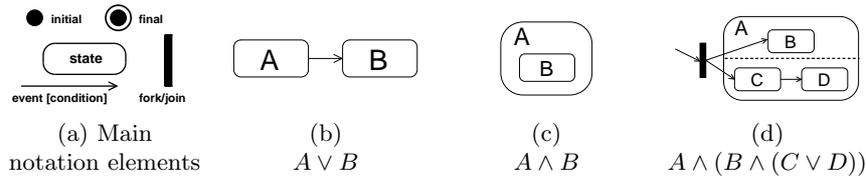


Fig. 2. UML State Machines: Syntax and combinations of states

it may become “cancelled”, if the check fails. Otherwise, if the credit is OK, the overall state is now called “current”, which is a composite state that contains a lot of detail in terms of substates. If the contract is never signed, it becomes “past” after one year.

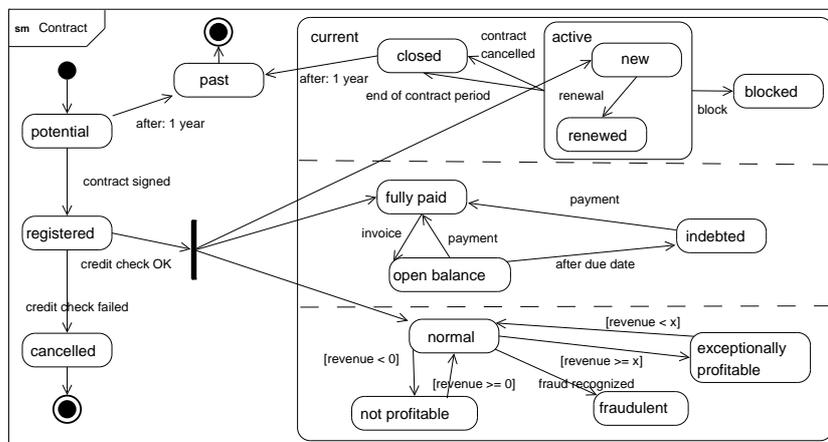


Fig. 3. State Machine diagram for a post-paid mobile phone contract

“Current” contracts are initially “normal”, “fully paid”, “active”, and “new”, as indicated by the transition arcs entering the composite state. The three regions within “current” are concurrent and orthogonal, meaning that at each point in time the object has a valid state in each of the three regions, and that what happens in one region does not influence the other regions.

In the first region, the contract starts as “active” and can become “closed”, either because the contract period ends or because the contract is cancelled beforehand. If a contract is blocked for some reason, it is neither active nor closed. Active contracts start as “new” and can become “renewed”. Closed contracts are moved from “current” to “past” after one year.

The middle region relates to the financial aspects of the contract: Each time an invoice is issued for a contract, it moves from “fully paid” to “open balance”, and when the payment for the invoice arrives, it moves back again. If an open balance is not paid until its due date, the contract is “indebted”. Issues regarding the amounts of the payments are not shown for sake of clarity (cf. Fig. 1).

In the last region, the movements of the contract object between the states depend mostly on certain conditions becoming true: If the revenue gained from this contract supercedes the amount x , it becomes “exceptionally profitable”, meaning that this customer is very valueable to the company. Whereas, if the revenue falls below 0, keeping this contract is actually causing financial damage (i.e. by producing higher costs than monthly payments). The fourth state in this region is “fraudulent”: It indicates that fraud connected to this contract has been discovered.

3 Business Objects in the DWH

The aim of our approach is to create conceptual models for making visible where business objects and their states are available in the DWH. Before we can construct a UML Profile, we have to identify the DWH elements we will use. This section contains a metamodel of these elements, an overview of the correspondences between the DWH elements and business object states, the user groups that the UML Profile is aimed at, and finally a tabular overview of the correspondence patterns.

3.1 The Metamodel

Figure 4 shows a metamodel of the elements that may represent business objects in a DWH. There are different kinds of *data repositories*, one of which may be a *data mart*. Other subtypes of data repositories [2] are not used in this paper for sake of simplicity. Data repositories contain *data objects*, which are *facts* or *entities*, depending on the type of the repository. Entities have *entity attributes*, whereas facts may have *measures*. Each fact consists of at least two *dimensions*, which may have several *levels* connected to each other via *roll-up* relationships. Dimensions are described by *dimension attributes*. Facts come in different types: *Transaction facts* or *snapshot facts*, of which there are two sub-types, *periodic* and *accumulating*.

The elements data repository, data mart, fact and entity of this metamodel (and also the UML Profile) are based on previous work by the authors. See [2] for a more detailed description.

3.2 Representation of Business Objects and their States in a DWH

Not all elements of this metamodel are useful for all conceptual modeling needs. We have identified a number of correspondence patterns along with the main usage scenarios of our approach. They are grouped into three levels and one additional category: the *overview* level, the *fact* level, the *attribute* level, and finally correspondences related to *aggregation and classification*. This subsection gives an overview; details and examples are given in Section 5.

Overview To show at a glance, where a business object can be found in the DWH, it can be linked as a whole to several data marts, fact tables, dimension tables or entities.

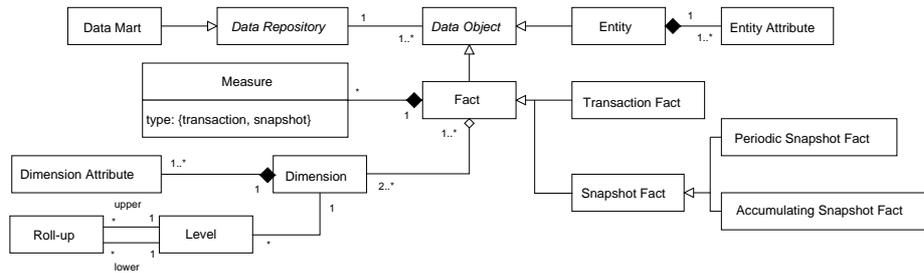


Fig. 4. A metamodel of DWH elements used to represent states of business objects

Fact Level The relationship of transitions and states of business objects to different types of fact tables can be shown on this level. As Kimball described in [3], there are three main types of measurements, which are the fundamental grains of fact tables: *transaction*, *periodic snapshot*, and *accumulating snapshot*. A *transaction fact* corresponds to a *transition* between states, whereas *snapshot facts* correspond to *states*.

Attribute Level Showing even more detail, a state of a business object may be represented by dimensions or measures, whereas the transactions and their guard conditions are found in measures or dimension attributes.

Aggregation and classification Guard conditions on state transitions may also appear in aggregation and classification hierarchies. Nested states correspond directly to roll-up relationships.

3.3 User groups

The conceptual model presented here is designed to answer the needs of two separate types of users:

DWH managers and *business users* are looking for the big picture, an overview of what to find where. They will use models showing business objects with correspondences mainly on the *overview level* (see Section 5.1 below). This allows them to find answers to questions such as “Which business object is in which data mart?”, “Is there a fact that corresponds to this business object?”, “If I have this dimension, does it represent this business object?”

DWH designers and developers need to understand the details of how business objects states and transactions between them can be represented in a DWH. They will use a fine-grained state machine model of a business object mapped to a data model. The *correspondence patterns* identified by us give hints on which elements can be used in the DWH model to represent the characteristics of the business objects, thus improving the creation and evolution of DWH models.

3.4 Correspondence patterns

Based on an analysis of the main requirements of the user groups, Table 1 gives an overview of which elements of the UML Profile may be linked to which in the conceptual model to show how business objects (elements on horizontal lines)

are represented in the DWH, and which elements of the DWH (vertical columns) can be used to represent the states of the business objects (further details of the correspondence patterns in Section 5).

Table 1. Correspondence patterns between elements describing business object states (horizontal) and elements of the DWH (vertical)

	Fact	Transaction F.	Snapshot F.	Measure	Dimension	D. Attribute	Level	Roll-up	Entity	E. Attribute	Data Mart
Object of State Machine	X				X				X		X
State		X	X		X	X					
Transition		X							X		
Guard condition			X		X	X					
Nested States								X			

4 The UML Profile for Representing Business Object States in a Data Warehouse

We introduce the UML Profile for Representing Business Object States in a DWH. It provides an easy to use yet formally founded way to model the correspondences between business object states and DWHs.

The Unified Modeling Language (UML) can be extended and adapted to a specific application area through the creation of profiles [4]. UML profiles are special UML packages with the stereotype `<<profile>>`. A profile adds elements while preserving the syntax and semantic of existing UML elements. It contains stereotypes, constraints and tag definitions.

A *stereotype* is a model element defined by its name and by the base class(es) to which it is assigned. Base classes are usually metaclasses from the UML metamodel, for instance the metaclass `Class`. A stereotype can have its own notation, e.g. a special icon.

Constraints are applied to stereotypes in order to enforce restrictions. They specify pre- or postconditions, invariants, etc., and must comply with the restrictions of the base class [4]. We use the Object Constraint Language (OCL) [5] which is widely used in UML profiles to define constraints in our profile, but any language, such as a programming language or natural language, may be used.

Tag definitions are additional attributes assigned to a stereotype, specified as name-value pairs. They have a type and can be used to attach arbitrary information to model elements.

The UML Profile for Representing Business Object States in a DWH makes it possible to model how the DWH used to analyze business objects is related to the lifecycle and states of these objects. Its stereotypes are described in detail in Table 2. These stereotypes can be used directly in UML State Machine diagrams [4]. In Fig. 5 we show a part of the UML 2 metamodel (light) to illustrate how the stereotypes we designed (dark) fit into to the existing UML

meta-model. The relationships between the stereotypes correspond to the meta-model presented in Section 3.

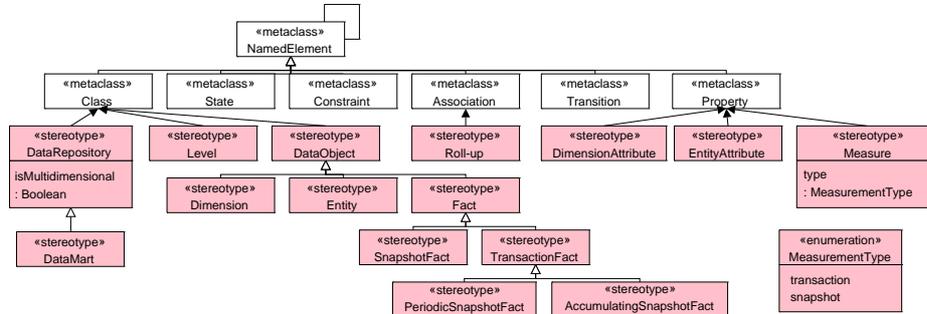
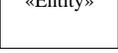


Fig. 5. UML stereotypes for representing business object states in a DWH

We use **Class**, **Association**, and **Property** as base classes for the stereotypes. This allows us to model their relationships with elements used in state machines such as **State**, **Transition** and **Constraint**.

Table 2: Stereotype Definitions of the UML Profile for Representing Business Object States in a DWH (cf. diagrams in Fig. 4 and 5)

Name	DataRepository	
Base class	Class	
Description	A data repository represents a type of database used in data warehouse environments.	
Tag Definition	isMultidimensional Type: AuxiliaryConstructs::PrimitiveTypes::Boolean Multiplicity: 1 Description: Indicates whether the data model of the DataRepository is a multidimensional data model	
Constraints	A DataRepository must be related to at least one DataObject: self.dataObject->size() >= 1	
Name	DataMart	
Generalization	DataRepository	
Description	A data mart is a departmental subset of a DWH focused on a single subject area.	
Name	DataObject	
Base class	Class	
Description	A data object is part of the data model contained in a data repository. The stereotypes Fact and Entity are derived from DataObject.	
Constraints	A DataObject must belong to exactly one DataRepository: self.dataRepository.size() = 1	
Name	Fact	
Generalization	DataObject	
Description	A fact is a data object of a multidimensional data model.	
Constraints	The DataRepository containing a fact must have a multidimensional data model: self.oclassType(DataObject).dataRepository.isMultidimensional = true The Fact must have at least two Dimensions: self.dimension->size() >= 2	
Name	Entity	
Generalization	DataObject	
Description	An entity is a data object of an E/R model.	
Constraints	The DataRepository containing an entity must not have a multidimensional data model: not self.oclassType(DataObject).dataRepository.isMultidimensional The Entity has at least one EntityAttribute: self.entityAttribute->size() >= 1	
Name	TransactionFact	
Generalization	Fact	
Description	A TransactionFact contains measures of transactions.	
Constraints	A TransactionFact may only contain transaction measures. self.allAttributes->forall(a a.oclassTypeOf(Measure) implies a.type=transaction)	

Name Generalization Description Constraints	SnapshotFact Fact A Snapshotfact contains snapshot measures, e.g. inventories. A SnapshotFact may only contain snapshot measures. <code>self.allAttributes->forall(a a.ocIsTypeOf(Measure) implies a.type=snapshot)</code>	
Name Generalization Description	AccumulatingSnapshotFact SnapshotFact The measures of an entry in the AccumulatingSnapshotFact are gathered over time	
Name Generalization Description	PeriodicSnapshotFact SnapshotFact The measures of a PeriodicSnapshotFact are acquired periodically for all instances	
Name Base class Description Tag Definition Constraints	Measure Attribute A numeric Measure is the object of analysis. measurementType Type: MeasurementType (Enumeration) Multiplicity: 1 Description: Indicates the type of measurement: <i>transaction</i> or <i>snapshot</i> A Measure must belong to exactly one Fact: <code>self.fact.size() = 1</code>	«Measure»
Name Base class Description Constraints	EntityAttribute Attribute An attribute to an Entity. An EntityAttribute must belong to exactly one Entity: <code>self.entity.size() = 1</code>	«Entity-Attribute»
Name Base class Description Constraints	Dimension Class Dimensions provide context for the measures and together are assumed to uniquely determine them. A Dimension is used by at least one Fact: <code>self.fact->size() >= 1</code> A Dimension has at least one DimensionAttribute: <code>self.dimensionAttribute->size() >= 1</code>	«Dimension»
Name Base class Description Constraints	DimensionAttribute Attribute DimensionAttributes describe Dimensions. A Dimension Attribute belongs to exactly one Dimension: <code>self.dimension->size() = 1</code>	«Dimension-Attribute»
Name Base class Description Constraints	Level Class Levels of Dimensions are used to aggregate Measures. A Level belongs to exactly one Dimension: <code>self.dimension->size() = 1</code>	«Level»
Name Base class Description Constraints	Roll-up Association A Roll-up-Association between two Levels indicates that Measures can be aggregated from the lower to the upper Level. A Roll-up-relationship has exactly one upper and one lower Level: <code>self.upper->size() = 1 and self.lower->size() = 1</code>	

5 Correspondence Patterns and Examples

To create conceptual models for making visible where business objects and their states are available in the DWH, we link elements from DWH conceptual data models with state machines of business objects. In this section, we apply the correspondence patterns already introduced in Section 3 to examples.

5.1 Overview level

We can link the business object to one or more *data marts* to provide an overview (Figure 6a). Then we can identify where each business object can be analyzed in the DWH model. A business object may correspond to a *fact table* (e.g., order, shipment, account; Figure 6b), a *dimension table* (e.g., product, customer, account), or in the case of a pure E/R-model to an *entity*.

5.2 Fact level

There are three main types of measurements or fundamental grains of fact tables: *transaction*, *periodic snapshot*, and *accumulating snapshot* [3]. Looking at

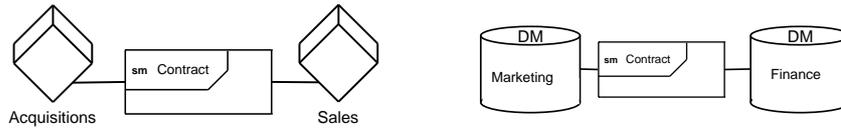


Fig. 6. The business object “Contract” in the fact “Sales” (left) and data marts (right)

state charts, we find that a *transaction fact* corresponds to a *transition* between states (Fig. 7 (a)), whereas *snapshot facts* correspond to *states*. Periodic snapshots record the current state for all instances of a business object at regular intervals (b), whereas accumulating snapshots “follow” each instance as it passes from state to state, adding values to the record over time (c). In the latter case, the order in which the values are added must conform to the state chart.

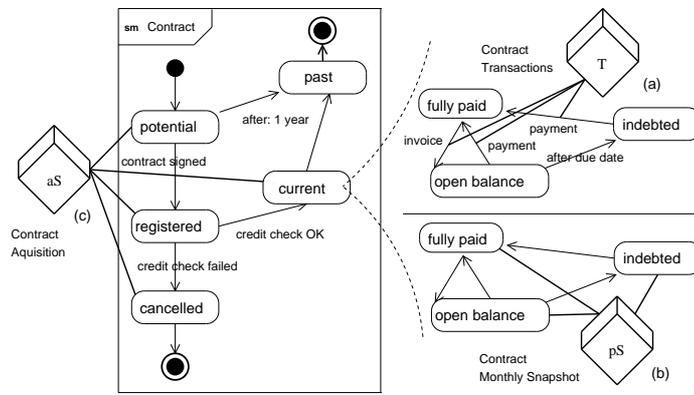


Fig. 7. Transitions in transaction facts (a), states in snapshot facts (b, c)

5.3 Attribute level

To provide a more detailed conceptual model, i.e. for DWH design and development, this level contains correspondence patterns related to all kinds of attributes (measures, dimension attributes, entity, attributes, guard conditions).

States A state in a state chart may play several roles in the DWH model. The most straightforward case is when there is an explicit “status” dimension (e.g., for account facts, insurance policies, etc.). In this case, several states of a business object are modeled by the dimension, i.e. as dimension attributes. Second, a state may be modeled as a measure. For a single state, this measure is of type boolean (either the object is in this state or not), and for several states, the measure would be an enumeration with the values corresponding to the states (which may be seen as a degenerate type of the status dimension).

Transitions Transitions in state charts may be guarded by conditions. These conditions often are found in the DWH as measures or dimension attributes. If

a guard condition contains a recognizable variable (e.g. “revenue” in Fig. 8(a)), this variable can turn into a measure in the DWH model.

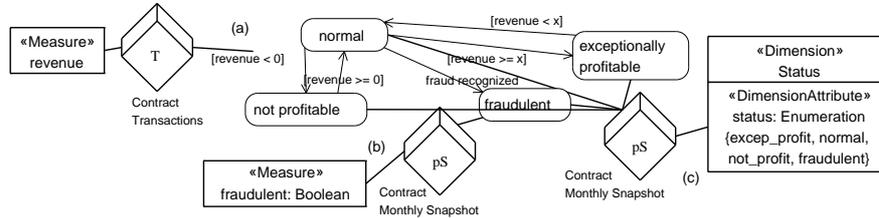


Fig. 8. Guard condition as measure (a), states as measure (b) or dimension (c)

5.4 Aggregation and classification

State charts of business objects may also provide hints regarding aggregation and classification hierarchies. First, nested states correspond to roll-up relationships, the inner state being the special case and the outer state the more general. Also, guard conditions may define levels in the hierarchy, as they separate instances into groups. In the case of a specific “status” dimension mentioned above, the states covered by this dimension then may also correspond to levels.

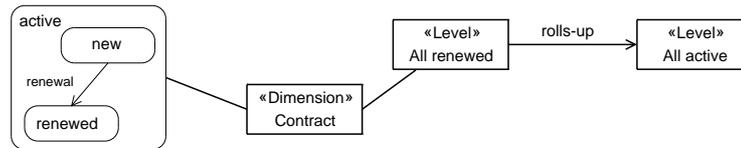


Fig. 9. Nested states of a business object correspond to a roll-up relationship

6 Related Work

Using UML Profiles to model the structure and behavior of Data Warehouses as well as related aspects such as ETL processes has become increasingly popular, also due to the rise of the Model-Driven Architecture (MDA [6]).

Trujillo and Luján-Mora introduced a Data Warehouse design framework in [7], which is supported among others by the UML Profile for Modeling ETL processes [8], a Profile for Physical Modeling of Data Warehouses [9], and a Profile for Multidimensional Modeling [10].

UML has also been applied to aspects such as DWH security. Fernandez-Medina et al. have extended UML for Designing Secure Datawarehouses [11].

In [12], Mazon et al. show how the MDA can be applied to Data Warehousing. Our contribution widens the scope of conceptual modelling in Data Warehousing to include more than structural models.

7 Conclusion

In this work, we have addressed that fact that there are no existing models for describing how business objects and their states are represented in a DWH. Business object states can be distributed over many parts of the DWH, and appear in measures, dimension attributes, levels, etc.

We introduced the *UML Profile for Representing the Lifecycle of Business Objects in a Data Warehouse*. It makes the relationship between the business objects and the DWH visible and accessible, as it allows to model various elements of DWHs and their data models in combination with UML 2.0 state machines. Using the UML Profile, we identified 14 correspondence patterns between state machines and DWH. The UML Profile and the correspondences are applied to an example.

References

1. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. 2 edn. Addison-Wesley (2004)
2. Stefanov, V., List, B., Korherr, B.: Extending UML 2 Activity Diagrams with Business Intelligence Objects. In: Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery, DaWaK 2005. Volume 3589 of Lecture Notes in Computer Science., Springer (2005) 53–63
3. Kimball, R.: Fundamental Grains. *Intelligent Enterprise* **2**(5) (1999)
4. Object Management Group, Inc.: UML 2.0 Superstructure. <http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf> (2005)
5. Object Management Group, Inc.: UML 2.0 Object Constraint Language (OCL) Specification. <http://www.omg.org/cgi-bin/apps/doc?ptc/05-06-06.pdf> (2005)
6. Object Management Group, Inc.: Model Driven Architecture (MDA). <http://www.omg.org/cgi-bin/doc?formal/03-06-01> (2004)
7. Luján-Mora, S., Trujillo, J.: A Data Warehouse Engineering Process. In: Advances in Information Systems, ADVIS 2004, LNCS 3261, Springer (2004) 14–23
8. Trujillo, J., Luján-Mora, S.: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In: Conceptual Modeling - ER 2003. Volume 2813 of LNCS., Springer (2003) 307–320
9. Luján-Mora, S., Trujillo, J.: Physical modeling of data warehouses using UML. In: DOLAP 2004, Proceedings, ACM (2004) 48–57
10. Luján-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modeling in data warehouses. *Data Knowl. Eng.* **59**(3) (2006) 725–769
11. Fernández-Medina, E., Trujillo, J., Villarroel, R., Piattini, M.: Extending UML for Designing Secure Data Warehouses. In: Conceptual Modeling - ER 2004. Volume 3288 of LNCS., Springer (2004) 217–230
12. Mazon, J.N., Trujillo, J., Serrano, M., Piattini, M.: Applying MDA to the development of data warehouses. In: Proceedings DOLAP '05, New York, NY, USA, ACM Press (2005) 57–66