



# From conceptual models to schemata: An object-process-based data warehouse construction method

Dov Dori<sup>a</sup>, Roman Feldman<sup>a</sup>, Arnon Sturm<sup>b,\*</sup>

<sup>a</sup>*Technion, Israel Institute of Technology, Haifa, Israel*

<sup>b</sup>*Ben Gurion University of the Negev, Beer Sheva, Israel*

Received 10 June 2007; received in revised form 19 January 2008; accepted 6 February 2008

Recommended by: F. Carino Jr

## Abstract

Data warehouse modeling is a complex task, which involves knowledge of business processes of the domain of discourse, understanding the structural and behavioral system's conceptual model, and familiarity with data warehouse technologies. The suitability of current data warehouse modeling methods for large-scale systems is questionable, as they require multiple manual actions to discover measures and relevant dimensional entities and they tend to disregard the system's dynamic aspects. We present an Object-process-based Data Warehouse Construction (ODWC) method that overcomes these limitations of existing methods by utilizing the operational system conceptual model to construct a corresponding data warehouse schema. We specify the ODWC method, apply it on a case study, evaluate it, and compare it to existing methods.

© 2008 Elsevier B.V. All rights reserved.

**Keywords:** Data warehousing; Data models; Database design; Conceptual modeling; Object-Process Methodology

## 1. Introduction

Data warehousing is a rapidly growing area within the field of information systems that focuses on the enablement of business intelligence [31,2]. Many companies, which invested resources in a new generation of operational information systems during the 1990s, are trying to gain added value from the vast amounts of information stored in their systems by applying different tools and techniques,

jointly dubbed “Business Intelligence” [31,3], for information analysis, decision support, and strategic planning. These techniques and tools analyze massive amounts of data, which are continuously gathered during the operation of large, complex information systems, in order to produce the information needed for decision-making processes while providing the end user with autonomy and flexibility in browsing, summarizing, cross-cutting, and analyzing the operational data. The requirements from these analytical tools have led to data warehousing—an approach to data analysis built on top of multidimensional data models that go beyond normalized relational databases [2,4]. Off-the-shelf data warehousing products supply solid

\*Corresponding author. Tel.: +972 8 6479337.

E-mail addresses: [dori@ie.technion.ac.il](mailto:dori@ie.technion.ac.il) (D. Dori), [feldmanrom@gmail.com](mailto:feldmanrom@gmail.com) (R. Feldman), [sturm@bgu.ac.il](mailto:sturm@bgu.ac.il) (A. Sturm).

physical data models and on-line analytical processing (OLAP) foundation. Yet, conceptual and logical design of the data warehouse remains a complicated human-intensive task, which is assigned to systems engineers and analysts of the organization that own and run these data warehouses.

Data warehouse (DW) design usually follows the Star schema diagram [2]. A Star schema consists of a fact table and a single de-normalized dimension table for each dimension of the corresponding data model. The dimension tables can be normalized to create a Snowflake schema, which can support attribute hierarchies. Both the Star and the Snowflake schemata present data as a single cube, which is the basic data warehouse structure that allows users to perform multidimensional data analyses.

Several conceptual data warehouse models have been developed to support the design of data warehouses [5–7]. While these approaches assist in data warehouse modeling, they address implementation issues only partially by creating or manipulating the Star or Snowflake architectures. For example, the Object Relational View (ORV) approach [8], which is an object-oriented data warehouse design framework, is obtained by a transformation of the Snowflake schema. Arguably, ORV can lead to better system performance through object links instead of the more expensive joins. After the transformation, the object-oriented schema has to be partitioned in order to reduce the number of joins and disk accesses. UML [9] extensions for data warehouse conceptual modeling have also been proposed [10,11]. OMG suggested the Common Warehouse Metamodel (CWM) [12,13], an approach to metadata interchange, which is capable of expressing major multidimensional properties [10].

Most data warehouse modeling techniques refer mainly to the user requirements [2] and advocate building a new multidimensional model regardless of existing models of the underlying operational systems. This approach raises several problems, as stated by [14]:

1. User requirements are unpredictable, often not validated, and subject to change over time, so they produce an unstable basis for design. One may have better chances of foreseeing future analysis requirements, such as the need for additional dimensions, in case the operational system schema, rather than user requirements,

serves as the basis for the data warehouse design.

2. Incorrect data warehouse designs might be generated if the designer does not understand the underlying relationships among the various data items.
3. Premature aggregation often causes loss of information that limits the options to analyze the data in future desired fine-grained resolutions.

These drawbacks have motivated studies that focus on using operational system conceptual models as a basis for data warehouse modeling, design, and construction. There are two major approaches for utilizing an operational system model for the construction of data warehouse models: the structure-based approach and the process-based approach.

The structure-based approach is motivated by the understanding that since data warehouses depend on the respective underlying operational systems as their data supply sources, the latter become important for the conceptual and logical design of a data warehouse. The techniques that follow this approach (for example, [1,5,14–19]) use mainly the entity-relationship concept [20] as the basis for the proposed solution. These techniques suffer from the following limitations: (1) the data warehouse designer is assumed to be familiar with the organization's business processes, (2) the behavioral aspects of the system are ignored, and (3) multiple manual transformations are needed to obtain a data warehouse model.

The process-based approach stems from the understanding that the fundamental role of data warehousing is to provide business performance measurement and redesign support [21,22]. With this in mind, several techniques were offered to create data warehouse models from business process models. Only one model, discussed in [23], creates a data warehouse from relatively generic business process models [24], and it can be applied to various content areas. However, since this technique is manual and does not relate to the operational system structure, the Extract, Transform, and Load (ETL) process is complicated. Another transformation technique [25], which is content-specific, is applied to rule-based Customer Relationship Management (CRM) systems. List et al. [22] offered a specific data warehouse model aimed at analyzing business workflow performance that uses a workflow management system as the data source. This approach might yield infeasible data warehouse

models that cannot be attained by loading data from operational sources, because it does not account for the data structure in the operational systems.

A survey of the state-of-the-art methods for transforming operational system conceptual models to data warehouse models along with their evaluation is presented in [26]. Based on this survey, a comparison between the various operational system-based data warehouse generation techniques is presented in Table 1 along the various criteria and their importance. Each technique is rated as good (G), average (A), low (L), very low (VL), or irrelevant (I). While the method ratings are subjective, we believe they represent the state of affairs with respect to techniques of data warehouse generation from operational system.

The *process automation* criterion was divided into five sub-criteria, and with respect to this criterion, most of the examined methods provide some means for semi-automatic transformation from an operational conceptual model to a data warehouse specification, but they comprise several manual activities that the data warehouse designer needs to perform. Most methods' *applicability to large-scale systems* is low. The criterion *required business process knowledge* is rated low for almost all the surveyed methods, since they utilize mostly structural models of the operational systems. As noted,

one needs to be acquainted with the business processes in order to select relevant transactional entities, make decisions in dimensional design, and define summarizability constraints. Nonetheless, the methods do not utilize information about the functionality of the underlying operational system. With respects to *handling business process analysis*, most methods do not adequately analyze the process perspective. They use the data of the system, but not its operational semantics or the system's dynamic aspect, which are expressed through processes. Disregarding the process perspective diminishes the usefulness of the model during the conceptual data warehouse design, as the modeling technique can express the ultimately obtained actual structure of the metadata in the data warehouse, but not the information needs that might have been discovered along the process path. The methods leave *mapping relevant dimensional attributes* to the designer. As for *support for data transformation from the operational system to the data warehouse*, none of the methods considers Extract, Transform, and Load (ETL) issues directly. However, most methods derive the data warehouse entities from the operational sources ER diagrams, implying that the resulting structures are reachable by transformations of the operational sources. While SOM utilizes business process models, it is not based on actual information offered by the operational sources.

Table 1  
Comparison between data warehouse generation techniques

Criterion	Importance	Technique						
		[16]	[1]	[15]	[18]	[14]	[19]	[23]
<b>Original Operational Model</b>		E/R	XML DTD	SERM ([32])	E/R	E/R	E/R	SOM ([24])
<b>Derived DW Model</b>		Dimensional Fact Model	Dimensional Fact Model	Star Schema	Fact Schema	Various	MER ([6])	Star Schema
<b>Process automation</b>	High							
a. Facts definition		L	L	L	L	L	H	L
b. Initial model creation		H	H	L	A	I	H	L
c. Dimensions (Hierarchies) definition		L	L	L	A	A	H	L
d. Refinement		L	L	L	L	L	VL	L
e. Summarizability constraints		L	L	L	A	L	L	L
<b>Applicability to large-scale systems</b>	High	L	L	A	L	L	VL	A
<b>Required business process knowledge</b>	Medium-High	L	L	L	L	L	L	A
<b>Handling business process analysis</b>	High	L	L	L	L	L	L	H
<b>Mapping relevant dimensional attributes</b>	Low-Medium	A	A	A	A	A	A	A
<b>Support for data transformation from an operational system to the data warehouse</b>	High	H	A	H	H	H	H	VL

To address the problems of existing methods for data warehouse generation from operational system specification, we suggest considering both the structural and behavioral aspects of the system. We have chosen Object-Process Methodology (OPM) as the system specification language, primarily because it unifies the major system aspects—function, structure, and behavior—within a single view. Consequently, we propose a new DW construction method, called ODWC—Object-process-based Data Warehouse Construction, which creates a multidimensional conceptual data warehouse model. We automate and simplify the data warehouse design process by utilizing knowledge about the business processes within the OPM model in order to analyze their performance.

The rest of the paper is organized as follows. In Section 2 we present the Object-Process Methodology, discuss its suitability for data warehouse construction, and introduce a case study of a trading catalog system. In Section 3 we present the Object-process-based Data Warehouse Construction (ODWC) method, formalize its rules, and exemplify its use. We evaluate the method in Section 4 and conclude in Section 5.

## 2. Object-Process Methodology

Object-Process Methodology (OPM) is an integrated approach to the study, development, and lifecycle support of systems in general and information systems in particular [27]. The basic premise of the holistic OPM paradigm is that objects and processes are two types of equally important classes of things, which together describe the function, structure, and behavior of complex, multidisciplinary systems in a single, domain-independent model, expressed in both formal yet intuitive graphics and an equivalent subset of natural language. OPM unifies the major system lifecycle stages—initiation, development, and deployment—within one frame of reference, using a single diagramming tool, a set of Object-Process Diagrams (OPDs), and a corresponding subset of English, called Object-Process Language (OPL). The OPM elements are presented in Appendix A. In OPM, existing things, their attributes, physical devices, human users, and environmental interfaces, are modeled as object classes. An object can be stateful, i.e., it can be at one of a certain number of possible states. Processes are things that transform objects by generating or consuming them, or by changing their state. At any

point in time, each object is at some state, and objects are transformed (generated, consumed, or affected, i.e., their state is changed) through the occurrence of a process. Things—objects or processes—can be either systemic (internal to the system) or environmental (external to the system and interacting with it). In an orthogonal manner, a thing can be either physical or informational (logical).

Unlike the object-oriented approach, behavior in OPM is not necessarily encapsulated as an operation or method within a particular object class construct. A process class can involve any number of object classes rather than be an operation of exactly one object class, as imposed by the object-oriented encapsulation principle. By allowing for the existence of stand-alone processes, one can model behavior that involves several object classes intuitively and in a straightforward manner rather than having to twist reality to conform to the encapsulation principle, as is often the case with OO-based modeling. Moreover, OPM enables the modeler to specify that some of the involved objects are transformed, while others enable the process without being transformed. The modeled behavior is integrated into the system's structure in a single bimodal (graphical and textual) model, so system dynamics is represented in a balanced way beside its static composition. Processes are connected to the involved object classes through procedural links, which are classified into enabling, transformation, and event links.

OPM's built-in scaling (refinement/abstraction) mechanisms, which are unfolding/folding, in-zooming/out-zooming, and state expression/suppression, help manage the system's complexity by controlling the visibility of details in any OPD and providing for creating new, interconnected OPDs at higher or lower levels of details. Unfolding/folding is applied by default to objects for exposing/hiding their structural components (parts, specializations, features, or instances) in a hierarchical manner. In-zooming/out-zooming is applied by default to processes for exposing/hiding their sub-process components and details of the process execution. Time within an in-zoomed process flows downwards, providing for the specification of the order in which subprocesses are executed, so processes depicted at the same height are concurrent. The third scaling mechanism, state expression/suppression, enables showing or hiding all or some of the states of an object class.

An OPM specification of a system includes a hierarchical set of Object-Process Diagrams (the OPD set). The root of the hierarchy, the System Diagram (SD), is the top-level OPD, which models the function—the highest level process of the system—along with the interacting systemic and environmental objects. Lower level diagrams recursively apply the scaling mechanisms, mainly in-zooming of high-level processes or unfolding of high-level objects.

### 2.1. Benefits of OPM for data warehouse construction

We found OPM adequate for the task of transforming an operational system specification into a data warehouse schema specification for the following reasons:

OPM's scaling (abstraction/refinement) mechanisms and its hierarchical system model structure lend themselves conveniently to presenting the operational system and the supported business processes at different levels of abstraction or granularity in a unified manner. This feature enables the designer to easily navigate and select the business process to be analyzed at the right level of detail. This is in contrast with the need to seek and discover measures throughout the (often very large and complicated) entity-relationship diagram, as is the case with most methods that utilize the operational system structure to construct a data warehouse. The scaling mechanisms of OPM may also be used to create cubes at different summation levels. Using processes that are located at lower levels in the hierarchy leads to the generation of more detailed cubes with less summation of facts and consequently higher data resolution, and vice versa.

Existing methods that create multidimensional structures from operational systems construct dimensions and dimensional hierarchies by progressing from the measures (facts) along all the possible many-to-one relationship chains. While this approach may create correct data warehouse structures, it lacks the ability to collect all the needed data to investigate and analyze the business process measures. This procedure may cause insertion of irrelevant entities in dimensions and/or miss other relevant entities, since they were not traced along the many-to-one relationship chains. The OPM representation of business processes as complementary things alongside objects at any level of

abstraction provides for pinpointing dimensions that are relevant to the business processes of interest and for situating them precisely within the dimension hierarchy of the data warehouse schema.

OPM enables clear identification of the outcomes of a business process. These outcomes may include one or more newly created objects, identified by the result link, and one or more affected objects (objects whose state changed), identified via the effect link or via a pair of an input link (from the input state to the process) and an output link (from the process to the output state). To assess the performance of business processes of interest, data warehouse based analysis should inspect the objects resulting from, or affected by, the process under examination, along with their attributes. Hence, the measures in the Star or Snowflake structure can be these objects' attributes, and in most cases this decision can be automated based on the system's OPM model.

### 2.2. A purchasing system case study

In this section we present an OPM-based case study on which we demonstrate the proposed Object-process-based Data Warehouse Construction (ODWC) method. The case study focuses on purchasing processes of a make-to-order organization. The organization maintains separate planning and purchasing functions. The purchasing activities begin as soon as the planning terminates with approved purchasing decisions. The purchase system supports the following functions: For regular (non-urgent) purchase decisions, vendor selection includes issuing a Request for Proposals (RFP) to vendors, receiving their proposals, and negotiating further to improve the proposals. Eventually, the best proposal is selected. For an urgent purchase, the default vendor is assigned automatically and the order is issued. After the goods have been received and treated by the inventory worker, the order is paid for and closed.

Fig. 1 is the System Diagram (top-level Object-Process Diagram, OPD) of **Goods Purchasing**, which represents the function of the system. The planning process, which is out of the scope of the system, results in an approved **Purchasing Decision**, which is then used as the main input for the **Goods Purchasing** process. Following is a description of this OPD in Object-Process Language (OPL), a subset of English, which was created automatically from the Object-Process Diagram in Fig. 1 by OPCAT-Object Process

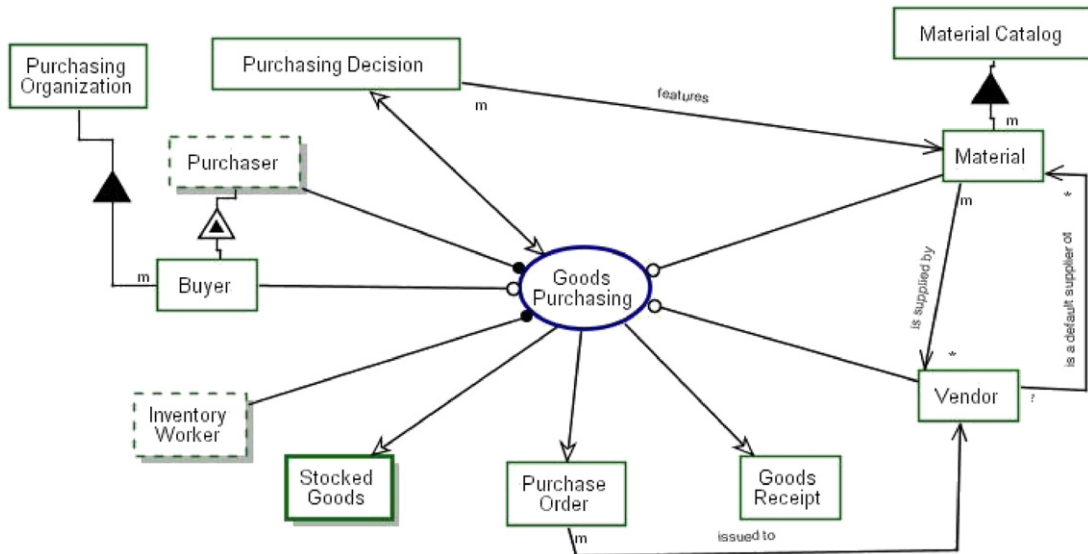


Fig. 1. The System diagram (SD) – the top-level OPD describing a **Goods Purchasing** process.

CASE Tool [28], the software environment that supports OPM modeling. Regular Arial font (as in “consists of”) denotes reserved OPL phrases, while bold font (as in “**Purchasing Organization**”) is used for non-reserved phrases. Text in brackets explains the OPL terms and is not part of the automatically generated text. Note the slight grammatical mistakes resulting from the automatic OPL sentence generation.

**Material Catalog** consists of many **Materials**. **Purchaser**, which is environmental and physical, exhibits [has the attribute of] **Buyer** [**Buyer** is the system representation of the physical and environmental **Purchaser**]. **Purchasing Organization** consists of many **Buyers**. **Purchasing Decision** exhibits **Material**. Many **Purchase Orders** issued to **Vendor**. Many **Materials** are supplied by optionally many [zero or more] **Vendors**. **Vendor is a default supplier of** optionally many **Materials**. **Purchaser** handles [is agent of] **Goods Purchasing**. **Goods Purchasing** requires **Material**, **Buyer**, and **Vendor**. **Goods Purchasing** affects **Purchasing Decision**. **Goods Purchasing** yields **Stocked Goods**, which is physical, **Purchase Order**, and **Goods Receipt**.

The other parts of the system specification appear in Appendix B as OPDs along with their correspondence OPL paragraphs.

### 3. The object-process-based data warehouse construction method

Having introduced object-process, in this section we introduce the OPM-based Data Warehouse Construction (ODWC) method, which transforms the OPM operational system conceptual model into a data warehouse schema. We present the basic ODWC steps, the ODWC transformation rules and their rationale, the formalization of these rules, and a demonstration of their application on the case study.

#### 3.1. The basic ODWC steps

The ODWC approach follows an iterative user-guided search process. Each iteration consists of the following steps:

1. The data modeler selects a business process.
2. The ODWC system proposes possible Snowflake schemata that might describe the selected business process. This is done in two stages: creating a basic Snowflake schema and populating facts and dimensions. Each stage is carried out subject to several action rules. The rules for each stage are specified in the next subsection, along with the motivation behind each one of them.
3. The data modeler selects the Snowflake schema that is most appropriate for the organization's data mining needs.

4. Optionally, the data modeler can hierarchically navigate to more detailed OPDs that zoom into lower level processes and repeat the procedure from step 1 as many times as needed.

The transformation executed by the ODWC method constructs Snowflake schemata. A Snowflake schema is more informative than a Star schema, as it includes information also about possible navigation through the cube for obtaining information in response to specific queries. A corresponding Star schema can easily be obtained from the Snowflake schema by collapsing the dimensional hierarchies [14].

### 3.2. ODWC transformation rules and their rationale

#### Stage 1: Creating a basic Snowflake schema

*Action Rule 1.1: Select a process and create an empty Snowflake schema for each informatival (non-physical) object which is transformed (i.e., created, consumed, or affected) by the selected process.*

**Motivation:** One of the goals of the method is to enable business process measurement. The center of any multidimensional schema, its fact table, includes measurements (facts). A dimensional modeler is supposed to “strive to store the measurement data resulting from a business process in a single data mart” [2]. In the ODWC method, a business process is measured by its influence on the enterprise, which is identified in OPM by transformations that these objects undergo as a result of the process execution.

*Action Rule 1.2: Create a dimension for each object that the process transforms and for each object that enables the process, either as an instrument (a non-human enabler) or an agent (a human enabler).*

**Motivation:** Dimensions are supposed to include “virtually all interesting constraints and report labels” [20]. Methods that construct data warehouses from data models of operational systems use the structural relations between the fact entity and its neighboring entities to discover dimensions [1,14–16,18,19]. In OPM, this detection is refined by identifying objects that contribute to the business process and introducing procedural links between the dimension entities and the chosen business processes. This helps bring relevant dimensions into the constructed multidimensional scheme and

enhances its ability to measure the business process in terms of its inputs and outputs.

#### Stage 2: Populating facts and dimensions

*Action Rule 2.1: Create a fact for each quantitative attribute of a transformed object.*

**Motivation:** Facts in the fact table have to be quantitative to enable performing statistical operations such as summation or means analysis on them. We use attributes of the fact entity to ensure that the granularity of all the facts in the fact table is the same, as required by Kimball [22]. In the framework of an automatic schema creation, it is impossible to estimate the relevance of a given quantitative attribute as a fact. Therefore, we define all the quantitative attributes in the fact entity as candidate facts. Several other techniques [14,19] use a similar approach, while further manual assistance is required in others [15,16,18]. However, since all these techniques require at least one fact in order to create a multidimensional schema, they cannot be used to create a factless fact table [22]. In the proposed ODWC method, creating a factless fact table is possible, since a multidimensional schema creation is initiated (in Action Rules 1.1 and 1.2) regardless of the existence of quantitative attributes in the fact entity.

*Action Rule 2.2: Add all attributes of dimensional objects into the Snowflake schema.*

**Motivation:** We strive to create as wide dimension tables as possible, since dimension attributes “serve as the primary source of query constraints, groupings and report labels” [22]. Once a dimensional entity has been chosen (in Action Rule 1.2), each of its attributes may serve as a meaningful query constraint. Therefore, we include all the attributes of the dimensional entities as dimensional attributes.

*Action Rule 2.3: Define foreign keys in dimensional objects as navigational attributes.*

**Motivation:** To create dimensional hierarchies that expand our schema from Star model to Snowflake model, we make a single pass through the structural relations of the dimensional entity. We balance between two objectives of dimensional modeling: creating deep and eminent dimensions, which are vital if we are to make the data warehouse usable

and understandable on one hand, and exercising as little snowflaking as possible on the other hand. The reason for minimizing snowflaking is that it is believed to create slower applications that rely on more join operations, thereby hampering users' ability to browse within dimensions [2]. Our compromise allows the generation of one snowflaking level, as implemented by some commercial products [29]. The user can still perform deeper search for dimensional attributes and create more snowflaked schemata by repeating this action rule. This is possible since during the physical database design, Snowflake schemata can be reduced, or even transformed into de-normalized Star schemata, by the application of the "collapse" operator, as shown by Moody and Kortink [14].

*Action Rule 2.4: Add basic dimensions.*

**Motivation:** Basic dimensions include Time, Currency, and Measurement Unit. The Currency and Measurement Unit dimensions are needed to describe currency-dependent and unit-dependent facts [29], and must therefore be added to the schema if such facts that use them are present. The Time dimension, on the other hand, is needed in virtually every multidimensional schema [2] and is considered a key dimension in a data warehouse [17]. One of the Time dimension roles is to add the historical perspective to the analyzed data, which is vital to decision-making processes. Indeed, several data warehouse modeling techniques advocate addition of a time dimension regardless of the presence of time attributes in the chosen operational entities [15,16,19,23].

### 3.3. Formalizing the ODWC rules

Having described the ODWC rules and their rationale, in this section we present them formally. We start with the partial definition of OPM specification, followed by the definition of a Snowflake scheme, and finally we present the formalized transformation rules.

#### 3.3.1. A formal opm system model specification

**Definition 1:.** An OPM system model (*System*) is a set of OPDs. An *OPD* is specified by a six-tuple (*Obj*, *Proc*, *Li*, *Rel*, *fli*, *fre*), where:

- *Obj* is a finite set of (class) objects. Each object is defined by the tuple (name, type, essence, affiliation, key).

- *Proc* is a finite set of (class) processes. Each process is defined by the tuple (name, essence, affiliation).
- $fli : Li \rightarrow \{\text{effect, result, consumption, instrument, agent, invocation}\}$  is a function which maps each link from *Li* to its type.
- $fre : Rel \rightarrow \{\text{exhibition, generalization, aggregation, general}\}$  is a function which maps each relationship from *Rel* to its type
- $Li \subseteq (Obj \times Proc) \cup (Proc \times Obj) \cup (Proc \times Proc)$  is a set of procedural links.
- $Rel \subseteq (Proc \times Obj) \cup (Obj \times Proc) \cup (Obj \times Obj) \cup (Proc \times Proc)$  is a set of structural relationships.
- In case of exhibition relationship  $Rel \subseteq (Proc \times Obj) \cup (Obj \times Proc) \cup (Obj \times Obj)$
- In case of generalization, aggregation, and general relationship  $Rel \subseteq (Obj \times Obj) \cup (Proc \times Proc)$ .
- *Li* is a finite set of procedural links.
- In case of effect link  $el \in Li \subseteq (Obj \times Proc) \cup (Proc \times Obj)$
- In case of result link  $rl \in Li \subseteq (Proc \times Obj)$
- In case of consumption link  $cl \in Li \subseteq (Obj \times Proc)$
- In case of instrument link  $il \in Li \subseteq (Obj \times Proc)$
- In case of agent link  $al \in Li \subseteq (Obj \times Proc)$
- In case of invocation link  $inl \in Li \subseteq (Proc \times Proc)$

#### 3.3.2. A formal representation of a snowflake scheme

**Definition 2:.** A multidimensional data warehouse model (*DW*) may consist of one or more Snowflake schemata. A Snowflake schema *Snowflake* is specified by a nine-tuple (*Fact*, *Dim*, *Nav*, *Ass*, *Att*, *Key*, *getKey*, *getQAtt*, *getAtt*), where:

- *Fact*, *Dim*, and *Nav* are entities.
- $key \in Key$  is part of an entity identifier, defined by the tuple (name, type).
- $att \in Att$  is a property of an entity defined by the tuple (name, type).
- $Entity \subseteq (Key \cup Att)^*$  is a collection of keys and attributes.
- *Fact* (*Entity*) is a single entity within a Snowflake schema, defined by a key and a set of attributes.
- $Dim \subseteq Entity$  is a finite set of dimension entities, defined by a key and a set of attributes.
- $Nav \subseteq Entity$  is a finite set of navigational entities, defined by a key and a set of attributes.
- $Ass \subseteq (Dim \times Fact) \cup (Nav \times Dim)$  is a finite set of relationships.

### Function Definitions

- *getKey* is a function that return the keys of an OPM object or a Snowflake entity.
- *getQAtt* is a function that return the quantitative attributes of an OPM object or a Snowflake entity.
- *getAtt* is a function that return the attributes of an OPM object or a Snowflake entity.

#### 3.3.3. Transformation rules

*Stage 1: Creating a basic Snowflake schema*

*Action Rule 1.1: Create an empty Snowflake schema for each informatical object transformed (created, consumed or affected) by the selected process.*

$$\begin{aligned} &\forall O_i \in OPD_{current}, ((\exists(O_i, P_{selected}) \in Li \wedge (O_i, P_{selected}) \\ &\quad \rightarrow consumption) \vee \\ &(\exists(O_i, P_{selected}) \in Li \wedge (O_i, P_{selected}) \\ &\quad \rightarrow effect) \vee (\exists(P_{selected}, O_i) \in Li \wedge (P_{selected}, O_i) \\ &\quad \rightarrow result)) \rightarrow O_{ik} \equiv O_i \wedge O_{ik} \in Fact_k \wedge Fact_k \in Snowflake_k \end{aligned}$$

$$\begin{aligned} &\forall O_i \in OPD_{current}, ((\exists(O_i, P_{selected}) \in Li \wedge (O_i, P_{selected}) \\ &\quad \rightarrow consumption) \vee (\exists(O_i, P_{selected}) \in Li \wedge (O_i, P_{selected}) \\ &\quad \rightarrow effect) \vee (\exists(P_{selected}, O_i) \in Li \wedge (P_{selected}, O_i) \\ &\quad \rightarrow instrument)) \rightarrow O_{ik} \equiv O_i \wedge O_{ik} \in Dim_k \\ &\quad \wedge Dim_k \in Snowflake_k \end{aligned}$$

$$\begin{aligned} &\forall O_i \in OPD_{current}, (\exists(O_i, P_{selected}) \in Li \wedge (O_i, P_{selected}) \\ &\quad \rightarrow agent) \wedge (O_i, O_j) \in Rel \wedge (O_i, O_j) \rightarrow exhibition \\ &\quad \rightarrow O_{jk} \equiv O_i \wedge O_{jk} \in Dim_k \wedge Dim_k \in Snowflake_k \end{aligned}$$

$$\begin{aligned} &\forall O_i \in OPD_{current}, \forall K \in getKey(O_i), \forall O_{ik} \in Entity_k, \\ &\exists O_i \equiv O_{ik} \rightarrow K \in getKey(O_{ik}) \wedge \forall O_{ik} \in Dim_k \\ &\quad \rightarrow (O_{ik}, Fact_k) \in Ass_k \end{aligned}$$

*Action Rule 1.2: Create a dimension for each object that the process transforms and for each object that <Object Deletion Spot> enables the process (as an instrument or an agent).*

*Stage 2: Populating facts and dimensions*

*Action Rule 2.1: Create a fact for each quantitative attribute of a transformed object.*

$$\begin{aligned} &\forall O_i \in OPD_{current}, \forall A \in getAtt(O_i), \forall O_{ik} \in Dim_k, \exists O_i \\ &\quad \equiv O_{ik} \rightarrow A \in getAtt(O_{ik}) \end{aligned}$$

*Action Rule 2.2: Add all the attributes of dimensional objects into the Snowflake schema.*

$$\begin{aligned} &\forall O_{ik} \in Dim_k, \forall O_j \in System, \exists(O_j, O_{ik}) \\ &\quad \in Rel \wedge (O_j, O_i) \rightarrow general \rightarrow O_{jk} \\ &\quad \in Nav_k \wedge O_{jk} \equiv O_j \end{aligned}$$

$$\begin{aligned} &\forall O_{ik} \in Nav_k \rightarrow (O_{ik}, Nav_k) \in Ass_k \\ &\forall O_i \in System, \forall A \in getAtt(O_i), \forall O_{ik} \\ &\quad \in Nav_k, \exists O_i \equiv O_{ik} \rightarrow A \in getAtt(O_{ik}) \end{aligned}$$

*Action Rule 2.3: Define foreign keys in dimensional objects as navigational attributes.*

$$\begin{aligned} &\forall O_{ik} \in Dim_k, \forall O_j \in System, \exists(O_j, O_{ik}) \\ &\quad \in Rel \wedge (O_j, O_i) \rightarrow general \rightarrow \\ &\quad O_{jk} \in Nav_k \wedge O_{jk} \equiv O_j \\ &\forall O_{ik} \in Nav_k \rightarrow (O_{ik}, Nav_k) \in Ass_k \\ &\forall O_i \in System, \forall A \in getAtt(O_i), \forall O_{ik} \\ &\quad \in Nav_k, \exists O_i \equiv O_{ik} \rightarrow A \in getAtt(O_{ik}) \end{aligned}$$

*Action Rule 2.4: Add basic dimensions.*

$$\begin{aligned} &\forall A \in getQAttr(Fact_k), \forall O_j \in System, \\ &\quad A \equiv O_j, \exists(O_j, O_i) \in Rel \wedge (O_j, O_i) \rightarrow characterization, \\ &\quad O_i \in \{MeasurementUnit, Date, Currency\} \\ &\quad \rightarrow O_i \equiv O_{ik} \wedge O_{ik} \in Dim_k \wedge (O_{ik}, Fact_k) \in Ass_k \wedge \\ &\forall O_i \in System, \forall A \in getAtt(O_i), \exists O_i \equiv O_{ik} \\ &\quad \rightarrow A \in getAtt(O_{ik}) \end{aligned}$$

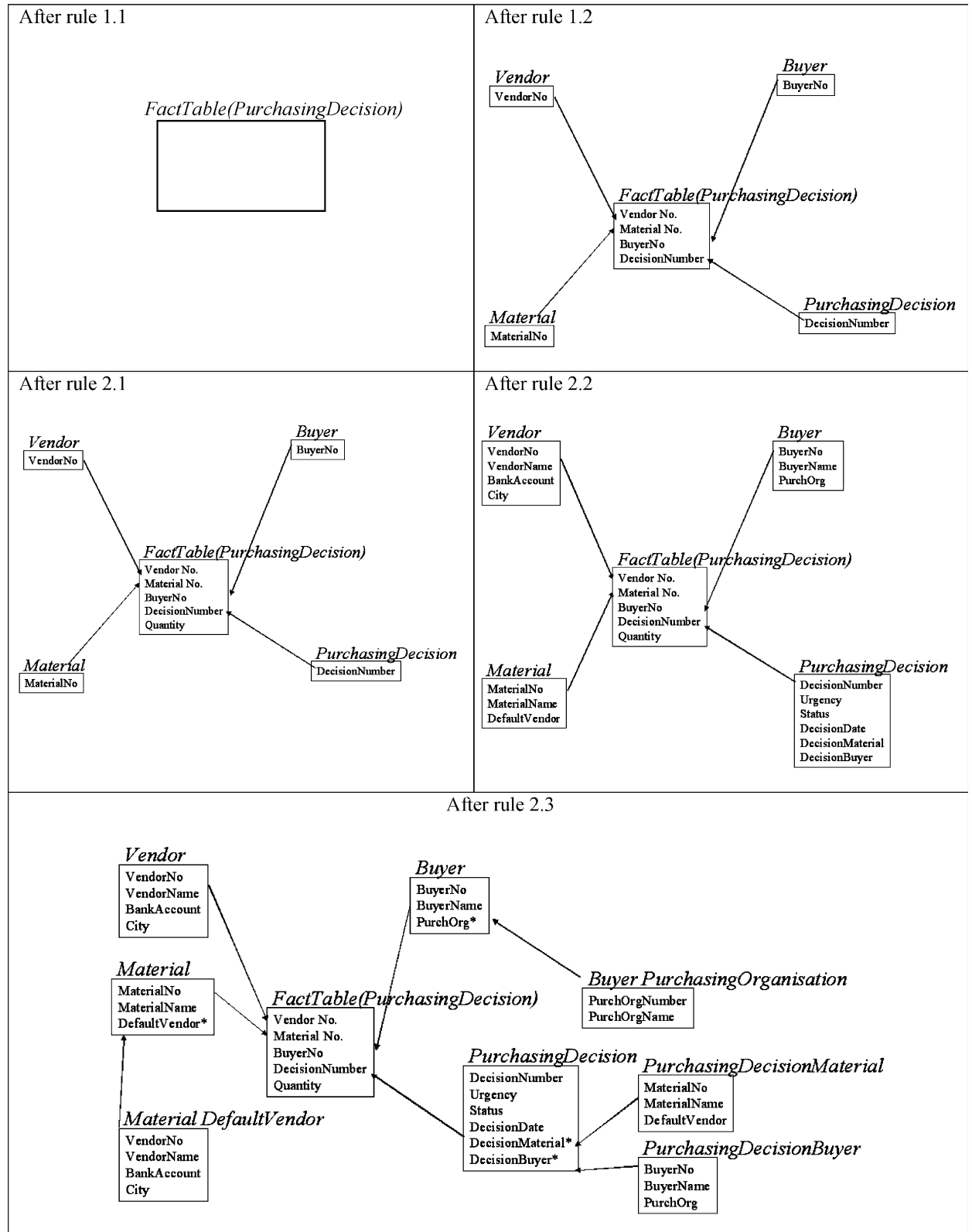
#### 3.4. ODWC in action

To demonstrate the work of the ODWC method, we start with the **Goods Purchasing** process, as presented by the System Diagram in Fig. 1. The construction of possible Snowflakes from an OPM process is performed in two stages, each consisting of several action rules, followed by an example from the **Goods Purchasing** process. Table 2 shows the intermediate results that the ODWC method produces related to **Purchasing Decision** DW schema.

*Stage 1: Creating a basic Snowflake schema*

*Action Rule 1.1: Create an empty Snowflake schema for each informatical object transformed (created, consumed or affected) by the selected*

Table 2

Intermediate conversion stages for **Purchasing Decision**

process. A transformed object is identified via the OPM link semantics specified in Appendix A and in [27].

Since a data warehouse is designed to analyze the performance of a business process, and since processes in OPM transform objects, we place the transformed objects in the middle of our cube as holders of the facts.

In our case study, empty Snowflakes are created for the object **Goods Receipt** and for the object **Purchase Order**, both created by the **Goods Purchasing** process, and for the object **Purchasing Decision**, which is affected by that process. Since the object **Stocked Goods** is physical, it does not trigger the creation of a corresponding Snowflake. In the sequel, we refer to the **Purchase Order** and the related **Purchasing Decision** schemata.

*Action Rule 1.2: Create a dimension for each object that the process transforms and for each object that enables the process (as an instrument or an agent). Only objects linked to the process at the current level of detail are to be taken in account.*

In our case study, referring to the **Purchase Order** and **Purchasing Decision** schemata, **Vendor**, **Material**, **Purchasing Decision** and **Buyer** are added as dimensions. Only informational objects should be proposed as attributes (dimensions) of the cubes. **Inventory Worker** also participates in the **Goods Purchasing** process, but being a physical object, it is not included in the schema.

If a physical object is an agent (a human) for the selected process, a corresponding informational object exhibited by the physical agent (and which represents it) is proposed as a dimension. In our case study, **Purchaser** is a physical agent represented by the informational object **Buyer**, which is an attribute of **Purchaser**, as expressed in the OPD in Fig. 1 and by the corresponding OPL sentence “**Purchaser** exhibits **Buyer**.”

At this stage of the Snowflake schema construction, the schema of **Purchasing Decision** deserves special attention. Since **Purchasing Decision** is affected by **Goods Purchasing**, it was recognized both as a fact entity by Rule 1.1 and as a dimension entity by Rule 1.2. This reflects the fact that the **Goods Purchasing** process uses this object as a source of information and

at the same time also contributes to this object. This exemplifies how ODWC supports the *line item* dimension [2] of a data warehouse, which is a Snowflake schema where a dimension table is like the fact table.

## Stage 2: Populating facts and dimensions

*Action Rule 2.1: Create a fact for each quantitative attribute of a transformed object.*

Quantitative (numerical or Boolean) attributes of the objects created or modified by the analyzed process (or objects characterizing it), which were placed in the middle of the Snowflake schema, are proposed automatically as facts. The quantitative attribute for the **Purchasing Decision** schemata is **Quantity** (see Table 2). Quantitative attributes for the **Purchase Order** schemata are **DeliveryInterval**, **Quantity**, **PayToVendor**, **DeliveryTermsCode**, and **ActuallyPaid**.

*Action Rule 2.2: Add all attributes of dimensional objects into the Snowflake schema.*

Objects that were chosen to be dimensions in the previous stage inherit their entire attribute set as dimensional attributes. In both schemata, the dimensional objects **Vendor**, **Material**, **Purchasing Decision**, and **Buyer** transfer their attribute sets into the created scheme.

*Action Rule 2.3: Define foreign keys in dimensional objects as navigational attributes.*

Attributes of dimensional objects, which participate in a structural link with other objects in the operational system conceptual model, are proposed as navigational attributes of those objects in the cube. This way, a dimensional hierarchy is created. The dimensional object **Vendor** has no foreign keys among its attributes. **Purchasing Decision**, on the other hand, has **Purchasing Decision Buyer** and **Purchasing Decision Material** as foreign keys, since many-to-one relationships are present between **Decision** and **Buyer** and between **Decision** and **Material**. Therefore, we define **Material** and **Buyer** as navigational attributes of the **Purchasing Decision** dimension and snowflake it into the **Purchasing Decision Material** and **Buyer** objects. To facilitate the identification of navigational objects, we use the concatenation of the dimension name and the attribute name. Similarly, we snowflake it

into **Default Vendor** in the **Purchasing Decision Material** object. Note that we do not continue this process indefinitely. For example, we do not search for foreign keys in the **Purchasing Decision Buyer** object. Infinite snowflaking is not recommended due to performance reasons [2] and may not be supported by industrial OLAP products.

*Action Rule 2.4: Add basic dimensions.*

The following dimensions are added automatically, regardless of the operational system conceptual model: time dimension (always), unit dimension (if a fact is characterized by the **Measurement Unit** reserved object [27] containing a unit, like a gallon or a kg.) currency dimension (if a fact is characterized by a currency object), and data packet (at the designer's discretion). Automatically added dimensions need to be populated with the relevant data (transaction time, unit of measurement, currency, or data upload packet) from the selected transactional data object. Filling the time dimension is done using the attributes of the transformed object that were modeled as date or date-time. If this data does not exist, the time dimension can be populated with the data loading date. Similarly, unit and currency dimensions should be established whenever the chosen facts are of quantitative type and are characterized by a **Measurement Unit** object or a **Currency** object. In these cases, a key attribute like **TimeKey**, **CurrencyKey**, or **UnitKey**, pointing to the relevant entry in the time/unit dimension table is created in the fact table.

We add time dimensions to our Snowflake schemata based on the **Decision Date** attribute of **Purchasing Decision** and **Order Date** of **Purchase Order**. In both Snowflake schemata, a **Unit of Measurement** dimension is added since the **Quantity** attributes of the **Purchasing Decision** and **Purchase Order** objects are characterized by **Measurement Unit** objects.

Applying the proposed method to the **Goods Purchasing** process, we obtain three similar Snowflake schemata, in which the dimensions are **Vendor**, **Material**, **Purchasing Decision** and **Buyer**, as well as **Unit of Measurement** and **Date**. The facts come from **Purchase Order**, **Purchasing Decision** or **Goods Receipt**. Fig. 2 presents the results of the transformation process for **Purchasing Decision** snowflake while Fig. C1 in Appendix C presents the resultant **Purchase Order** snowflake.

As can be seen without having to analyze the requirements for the DW, the cubes obtained by the ODWC method can be used for the following needs:

1. Analysis of workload over time in terms of the amount of **Purchasing Decisions** assigned to a **Buyer (Decision Dates)** can be performed using the **Purchasing Decision** cube. Further development will allow what-if analysis that simulates reassignment of **Materials** to **Buyers** or **Purchasing Organizations** in the enterprise.
2. Relevance of **Default Vendors** can be determined by comparing selected vendors to **Material Default Vendors**, either in the **Purchase Order** cube or the **Purchasing Decision** cube.
3. Updating **Default Vendors** can be done for certain **Materials** based on the actual **Orders** issued.
4. The influence of the urgency of purchasing decisions on the **Delivery Interval** and prices **Paid to Vendor** in the **Purchase Order** can be evaluated using the **Purchase Order** cube. The expectation is to get shorter **Delivery Intervals** while paying slightly higher prices.
5. Identification of preferred business partners (**Vendors**) in terms of overall purchasing volume, or purchasing volume for certain **Materials**, in order to introduce and negotiate scheduling agreements with them, allowing lower cost and more time-efficient purchasing.
6. Pareto analysis of quantity vs. volume of **Purchasing Decisions**, in order to identify low-cost commonly used materials and change their purchasing policies to purchasing to stock instead of make-to-order, thereby allowing buyers to devote more time to search alternative suppliers for the more expensive **Materials**.

Moving to the next iteration, we perform the transformation operations on the **Order Issuing** process, shown in Fig. 3, in which the **Buyer** issues an **Order** based on the **Selected Proposal** and **Purchasing Decision**. Fig. C2 in Appendix C shows the Snowflake scheme that was generated this time, where **Purchase Order** is the fact entity, **Selected Proposal** is transferred into a dimension, and **Decision**, **Buyer** and **Material** serve as additional dimensions. The time and unit dimensions were added, while the **Purchasing**

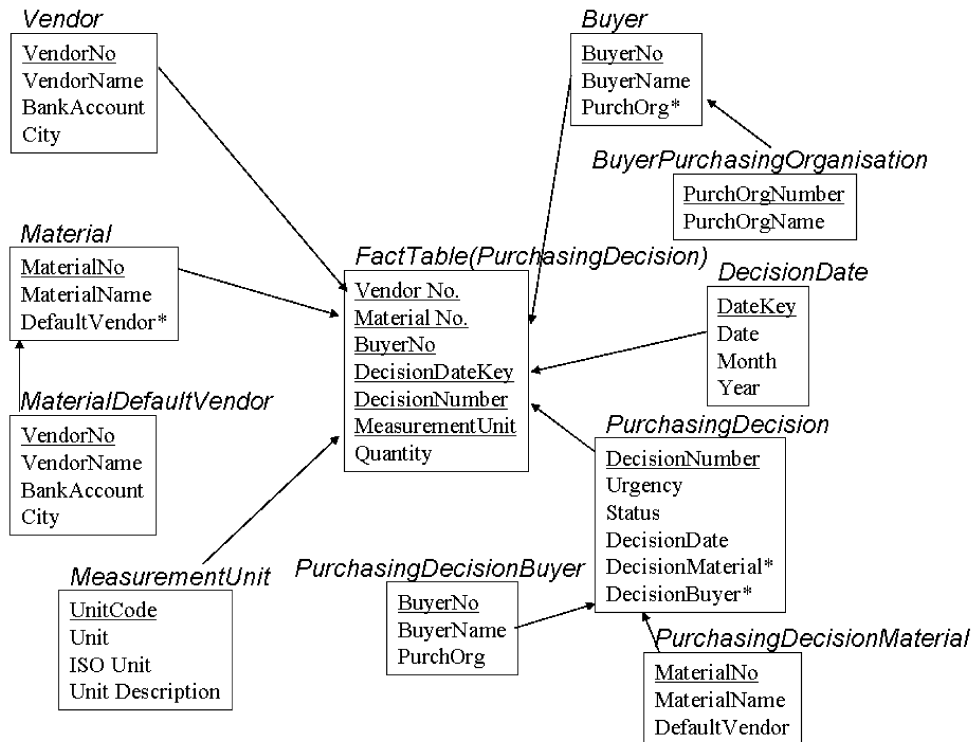


Fig. 2. A Snowflake schema of the **Purchasing Decision**, drawn from the **Goods Purchasing** process.

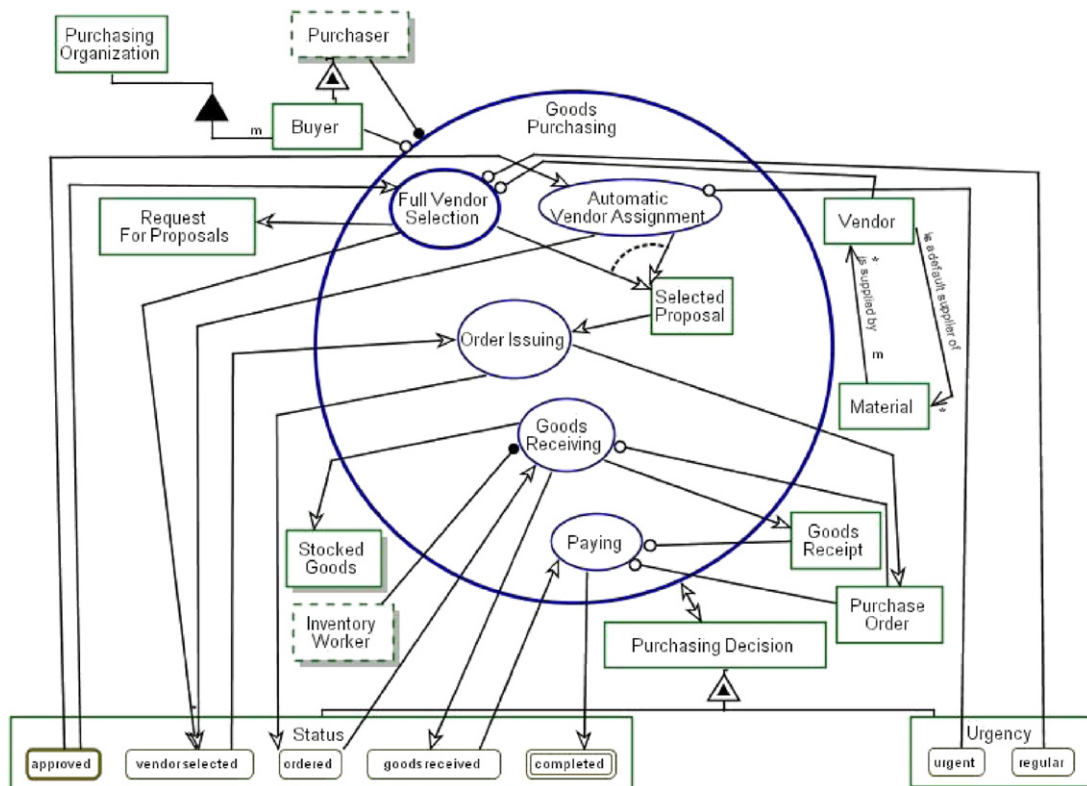


Fig. 3. Zooming into the **Goods Purchasing** process.

**Decision, Buyer, and Selected Proposal** dimensions snowflaked into their classifying entities.

Using this scheme, additional analysis can be performed, including the following.

1. Comparing **Quantity, Price, and Delivery** in the **Selected Proposal** and the actual **Order**: This is needed to ensure that while allowing flexibility to negotiate a quantity raise and a commensurable price adjustment, the **Buyer** indeed represents the enterprise's goal of increasing its profits. The cube helps discover unusual differences in quantities or prices between **Orders** and **Proposals** and/or **Decisions**.
2. Comparing between **Buyers, Vendors, and Materials**: The comparison can be done in terms of the result of negotiation after a proposal has been selected. It can help locating **Buyers** and **Vendors** with whom negotiation has achieved better results. It can also help pointing out adequate **Materials**, whose price was successfully negotiated. This indicates whether more time should be spent in order to find other new vendors that would offer an even lower price for these **Materials**.

Other interesting analyses include the match between **Request for Proposals** created by the **RFP Issuing** process and the resulting **Selected Proposal** and the relation to the **Proposal Selecting** process (Fig. 4).

#### 4. Method evaluation

Having presented and demonstrated the ODWC method, in this section we evaluate it according to the criteria discussed in Section 1, compare it to a representative method of data warehouse generation, and discuss its limitations.

##### 4.1. Criteria-based evaluation

###### 1. Process automation:

- *Facts definition*: Selecting a process automatically leads to definition of facts, so the ODWC method provides great assistance in carrying out this task.
- *Initial model creation*: Action rule 1.2 of the ODWC method creates dimensions automatically from the objects used by the selected process.

- *Dimensions definition*: Dimensions are defined automatically by searching the attribute sets of the dimensional objects.
  - *Schema refinement*: No automatic assistance is provided for schema refinement (as is the case with the other DW construction methods).
  - *Summarizability constraints*: Summarizability constraints need to be added manually (as is the case with the other DW construction methods).
2. *Applicability to large-scale systems*: The ODWC method inherits the complexity management strategy of OPM, enabling the architect to control the detail level of the things described in the OPM model. The transformation is performed on a selected process, which can be at any level of abstraction. While browsing through a hierarchical set of Object-Process Diagrams, the designer can perform top-down, bottom-up, or middle-out search to find processes of interest at their appropriate abstraction level, and then translate them into the corresponding data warehouse structures. The search can be performed using OPCAT, which provides such navigational features. Therefore, the size and complexity of the operational system are much less of a challenge to the DW designer compared with non-hierarchical representations of the operational system proposed by other data warehouse construction methods using operational system conceptual models.
  3. *Required business processes knowledge*: Following our approach, the data warehouse modeling is a natural extension of the business and operational system modeling. As such, the data warehouse modeler is assisted by the model itself in understanding the business processes.
  4. *Analyzing business processes*: The OPM-based operational conceptual system model is a straightforward representation of the enterprise's business processes.
  5. *Mapping relevant dimensional attributes*: Dimensional attributes are mapped by their relation to the process being analyzed.
  6. *Support for data transformation from the operational system to the data warehouse*: We derive the data warehouse entities from the operational source entities, thereby greatly simplifying the Extract, Transform, and Load (ETL) processes.

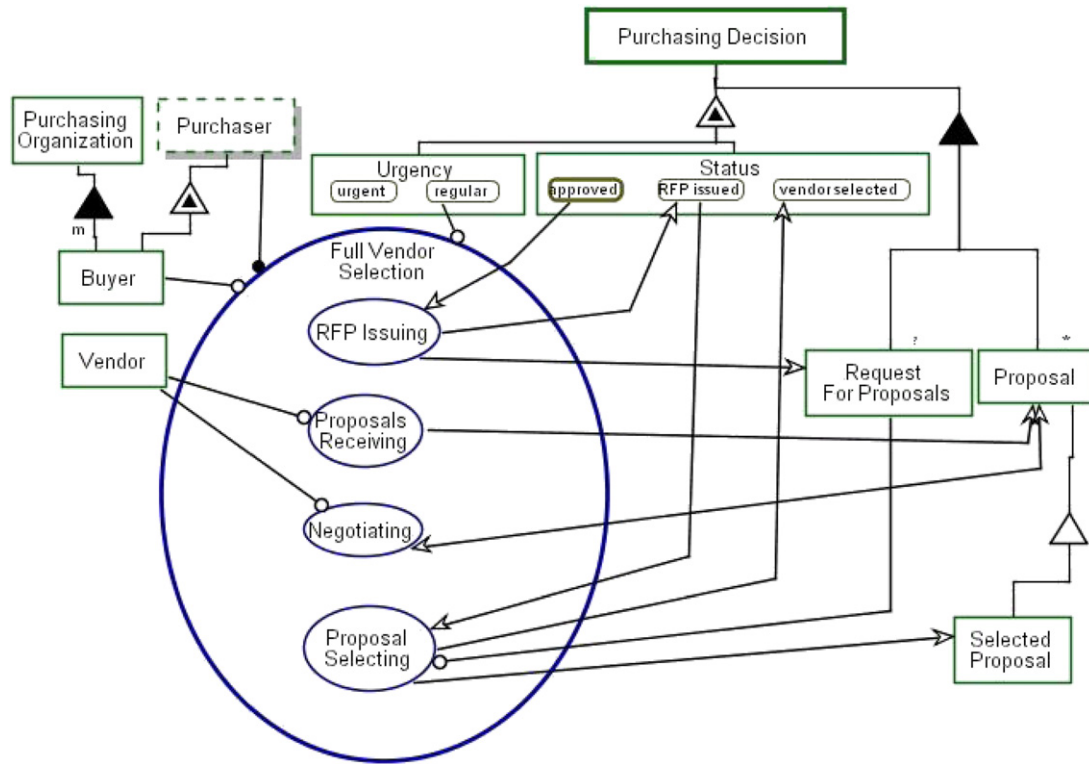


Fig. 4. Zooming into the **Full Vendor Selection** process.

#### 4.2. Comparison to a structure-based technique

To gain better understanding of the features, strengths and weaknesses of the ODWC method, we apply a structure-based method to our case study for the sake of comparison. We then analyze the differences between the structure method and ODWC method. We choose the Golfarelli-Maio-Rizzi Dimensional Fact technique suggested by Golfarelli et al. [16], since it is a well known method, which is also used as the framework for other methods for generating DW schemata. Following the principles of the chosen method, we construct an alternative multidimensional representation of the system shown in the Purchasing System case study.

Using the structural relationships between objects in our Object Process Diagram set, we developed the Entity-Relationship diagram (ERD) of our Purchasing Process case study. The resulting diagram is presented in Fig. D1 in Appendix D. Informational objects are transferred into entities along with their attributes, while structural links are translated to relationships. According to the ER principles, the diagram is a single-level visualization, neglecting the OPM hierarchical diagram structure.

To overcome this problem, Golfarelli et al. advise that “a portion of interest of an ER scheme”, rather than the whole diagram, should be used for the task [16]. Following the stages they suggested, we select **Purchasing Decision** as our fact entity. The attribute tree created appears in Appendix D. We then manually define the dimension and the facts. The resulting Snowflake scheme appears in Fig. D3 in Appendix D. Examination of the construction process performed by the Golfarelli-Maio-Rizzi method and its outcomes, compared to the outcomes achieved by ODWC method leads to the following insights:

1. The Golfarelli-Maio-Rizzi method does not aim at automating the initial multidimensional structures design, nor is it capable of achieving this task. Numerous intelligent decisions must be made by the designer during the initial model creation, the most crucial of which is the identification of the fact entity. Other decisions relate to pruning and grafting the attribute tree, where the designer should decide which tree vertices are “interesting” and omit definitions of some dimension, hierarchies, and fact attributes.

2. While the ODWC method allows for creating cubes with similar facts at different levels of summation, depending on the context of the selected business process, the Golfarelli-Maio-Rizzi method can reach variability of multi-dimensional representations only by repeating the design process while manually making various design decisions. The basic structure of cubes based on the same fact entity will remain unchanged, except for “pruning” and “grafting” different dimensional entities.
3. Fig. D2 presents **urgency** and **status** as direct descendants of the root vertex (decision number). Following the manual modeling stages, the child vertices of the root entity may either become dimensions or fact attributes. In our case, we chose neither to mark **urgency** and **status** as dimensions nor to use them in fact calculations. By doing so, we compromised our cube’s ability to perform data analysis, since we cannot use our cube to discover materials for which purchasing is badly planned (and therefore are regularly purchased in an urgent fashion), nor can we answer simple operational control queries, such as regarding the volume of decisions that are yet to be ordered. In the general decisions cube proposed by our method (see Fig. 2), both these scenarios are met by the presence of a line item dimension, a multidimensional design alternative that cannot be exploited using the Golfarelli-Maio-Rizzi method.
4. Structure-based methods use only structural relations between entities to construct a multi-dimensional structure. This approach prevents the incorporation of context-relevant entities. In our case, the chosen vendor has no direct structural relation with the decision, and therefore, the Golfarelli-Maio-Rizzi method does not include it as a dimension. **Vendor** is only included as the default vendor of a material. Hence, if one or more other vendors are involved in a decision-making process, the multidimensional representation does not reflect it.
5. Both cubes include **Purchasing Decision Quantity** as a fact. Since a quantity is categorized by a **Measurement Unit**, our method adds it as another dimension. This solution complies with commercial products standards on this issue [29]. The Golfarelli-Maio-Rizzi method disregards this issue, and in fact creates a scheme in which a single fact is non-additive on any of its dimensions.
6. The diagram achieved by the Golfarelli-Maio-Rizzi method includes the **Request for Proposals** dimension, which is not included in the cube proposed by our method. Since **Request for Proposals** has a 1-1 relationship with **Purchasing Decisions**, this difference does not alter the cube’s granularity (data resolution) while adding less analytic capabilities than what the ODWC based **Purchasing Decision** snowflake offers. Moreover, this “disadvantage” could easily be reverted by either (1) following the Golfarelli-Maio-Rizzi guideline to graft a 1-1 related vertex in an attribute tree or (2) using our method’s scalability in order to produce an additional cube, such as the **Purchasing Decision** cube from the **Proposals Selecting** process, where **Request for Proposals** will automatically be included as a dimension.

The most eminent disadvantage of the structure-based technique is its lack of ability to utilize existent many-to-one structural relations for tasks other than expanding the multidimensional structure. Since a valid Entity-Relationship Diagram includes a sufficient set of relations (as opposed to every imaginable relation), certain paths may become too short to construct a beneficial multi-dimensional structure. For example, considering **Purchase Order** as the fact entity, the only entity in the ERD of our case study related to **Order** by a many-to-one relationship is **Vendor**. As a result, **Vendor** is the only possible dimension in the diagram apart from attributes of **Order**, making the resulting cube almost useless for analysis. On the other hand, our method, which uses the procedural rather than structural links to identify possible dimensions, can achieve several multidimensional structures based on the **Purchase Order** as a fact entity, including those presented in Fig. C1 and Fig. C2.

#### 4.3. Limitations of the ODWC method

The limitations of the ODWC method relate to features which cannot be automatically specified. These include:

1. Addition of facts which do not exist in the operational system schema or need to be calculated during data load. Our example demonstrated the use of such calculated facts, which are defined manually in some of the

existing techniques [15,23]. Other techniques select (or allow manual selection of) numeric fields in operational entities as facts. Since our method locates facts and constructs initial multidimensional models automatically, such calculated facts can be added only after the Snowflake schema creation.

2. Introduction of summarizability constraints: As shown in [17], not all the possible aggregations of facts make sense. In Section 1 we classified existing techniques into those that allow manual addition of summarizability constraints and those that do not address this issue. In our technique, following the automatic multidimensional model creation, the modeler can add these multidimensional constraints manually.

## 5. Summary and future work

The newly proposed Object-process-based Data Warehouse Construction (ODWC) method transforms an operational system conceptual model into a data warehouse schema by following an ordered stepwise rule-based algorithm for the derivation of the DW schema from the source operational model. Using the ODWC method, a data warehouse architect or designer first selects processes to be represented in data warehouse structures. This can be conveniently accomplished by browsing the set of hierarchically organized Object-Process Diagrams of the OPM-based operational system model. The method then proposes adequate data warehouse structures based on the abstraction level of each selected process. We demonstrated the proposed approach via a case study, evaluated it by a set of predefined criteria, compared it to a mainstream method, and discussed its limitations. The evaluation has shown that the ODWC method is highly appropriate for the task of constructing data warehouse schemata, which can be effectively utilized for analyzing a plethora of business process performance tasks.

Existing methods appear to provide primarily a toolbox rather than an algorithm or a heuristic for the DW schema construction. This is one of the reasons for which the ODWC method overcomes a number of significant limitations of other methods. These limitations include the large amount of manual work needed to construct the DW schema from the operational system conceptual models, lack of assistance in discovering facts and in

identification of relevant dimensional attributes, and lack of appropriate reference to the process perspective.






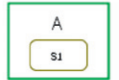
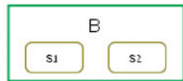
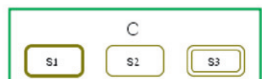
The adoption of Object-Process Methodology as both the conceptual modeling framework of the operational system and as the basis for the DW schema construction method is the source of the advantages of this combination. The guidelines for transforming the OPM system model into a data warehouse schema utilize the synergy of having the structural and procedural aspects of the underlying system in the same OPM model, enabling direct identification of objects representing both relevant input data entities and outcomes of business processes. This advantage, which is due to the unified single structure-behavior OPM view, is boosted by OPM's built-in scaling (abstraction-refinement) mechanisms that provide for unlimited scalability. In addition, the OPM scaling mechanisms allow for the creation of cubes at different summation levels. This unique feature of the method, demonstrated in the case study in this paper as well as in several other case studies, capitalizes on OPM's system modeling guidelines, which include detailed diagram creating by means of in-zooming the processes while adequately unfolding the objects.


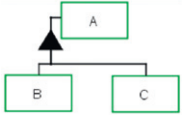
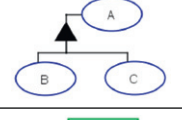
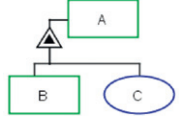
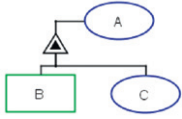
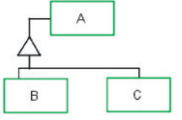
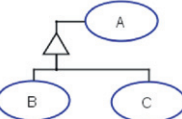
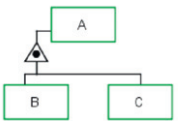
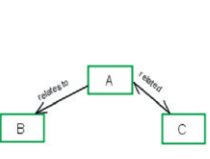

The ODWC method has been implemented and integrated into OPCAT [28] for assisting in semi-automatic creation of data warehouse schemata from an OPM model of the underlying operational system. The DW schema construction, which starts with a selection of a process by the user, allows the creation of visual models of the schema and exporting it as a UML class diagram in the XMI format [30] or as a relational diagram in the CWM relational metadata interchange format [13]. The ODWC module enables quick interactive construction of a DW schema by choosing the most appropriate one from amongst several automatically-created diagrams. Using this module we plan to continue the evaluation of the ODWC method on larger real life case studies and its subsequent improvement.


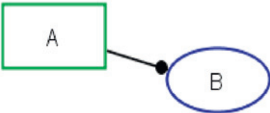
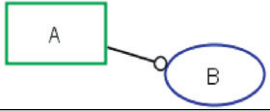
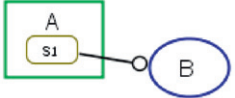
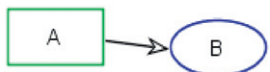
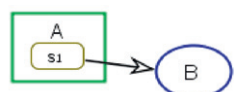
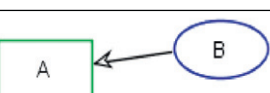
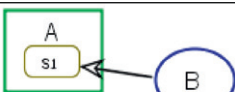
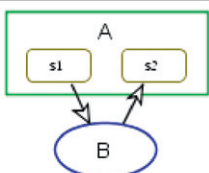

While in this paper we applied the ODWC method to OPM-based conceptual model of an enterprise operational system, we believe that the method can be applied to any modeling language that keeps the system structure and behavior closely related. This can be done for example in UML by synchronizing class and sequence diagrams or class and activity diagrams.

## Appendix A. A Quick Guide to the Syntax and Semantics of Object-Process Methodology (OPM)

A synopsis of the Syntax and Semantics of Object-Process Methodology is presented. More information is found in [27].

ENTITIES				
Name	Symbol	OPL	Definition	
Things	Object	 	<b>B</b> is physical. (shaded rectangle) <b>C</b> is physical and environmental. (shaded dashed rectangle)	An <b>object</b> is a thing that exists. A <b>process</b> is a thing that transforms at least one object.
	Process	 	<b>E</b> is physical. (shaded ellipse) <b>F</b> is physical and environmental. (shaded dashed ellipse)	Transformation is object generation or consumption, or effect—a change in the state of an object.
State		  	<b>A</b> is <b>s1</b> . <b>B</b> can be <b>s1</b> or <b>s2</b> . <b>C</b> can be <b>s1</b> , <b>s2</b> , or <b>s3</b> . <b>s1</b> is initial. <b>s3</b> is final.	A <b>state</b> is situation an object can be at or a value it can assume. States are always within an object. States can be initial or final.

STRUCTURAL LINKS & COMPLEXITY MANAGEMENT				
Name	Symbol	OPL	Semantics	
Fundamental Structural Relations	Aggregation-Participation		A consists of B and C.	A is the whole, B and C are parts.
			A consists of B and C.	
	Exhibition-Characterization		A exhibits B, as well as C.	Object B is an attribute of A and process C is its operation (method).  A can be an object or a process.
			A exhibits B, as well as C.	
	Generalization-Specialization		B is an A. C is an A.	A specializes into B and C.  A, B, and C can be either all objects or all processes.
			B is A. C is A.	
	Classification-Instantiation		B is an instance of A. C is an instance of A.	Object A is the class, for which B and C are instances. Applicable to processes too.
	Unidirectional & bidirectional tagged structural links		A relates to B. (for unidirectional)  A and C are related. (for bidirectional)	A user-defined textual tag describes any structural relation between two objects or between two processes.
	In-zooming		A exhibits C. A consists of B. A zooms into B, as well as C.	Zooming into process A, B is its part and C is its attribute.
			A exhibits C. A consists of B. A zooms into B, as well as C.	Zooming into object A, B is its part and C is its operation.

ENABLING AND TRANSFORMING PROCEDURAL LINKS				
Name	Symbol	OPL	Semantics	
Enabling links	Agent Link		<b>A</b> handles <b>B</b> .	Denotes that the object is a human operator.
	Instrument Link		<b>B</b> requires <b>A</b> .	"Wait until" semantics: Process B cannot happen if object A does not exist.
	State-Specified Instrument Link		<b>B</b> requires <b>s1</b> <b>A</b> .	"Wait until" semantics: Process B cannot happen if object A is not at state s1.
Transforming links	Consumption Link		<b>B</b> consumes <b>A</b> .	Process B consumes Object A.
	State-Specified Consumption Link		<b>B</b> consumes <b>s1</b> <b>A</b> .	Process B consumes Object A when it is at State s1.
	Result Link		<b>B</b> yields <b>A</b> .	Process B creates Object A.
	State-Specified Result Link		<b>B</b> yields <b>s1</b> <b>A</b> .	Process B creates Object A at State s1.
	Input-Output Link Pair		<b>B</b> changes <b>A</b> from <b>s1</b> to <b>s2</b> .	Process B changes the state of Object A from State s1 to State s2.
	Effect Link		<b>B</b> affects <b>A</b> .	Process B changes the state of Object A; the details of the effect may be added at a lower level.

EVENT, CONDITION, AND INVOCATION			
PROCEDURAL LINKS			
Name	Symbol	OPL	Semantics
Instrument Event Link		A triggers B. B triggers A.	Existence or generation of object A will attempt to trigger process B once. Execution will proceed if the triggering failed.
State-Specified Instrument Event Link		A triggers B. when it enters s1. B requires s1 A.	Entering state s1 will attempt to trigger the process once. Execution will proceed if the triggering failed.
Consumption Event Link		A triggers B. B consumes A.	Existence or generation of object A will attempt to trigger process B once. If B is triggered, it will consume A. Execution will proceed if the triggering failed.
State-Specified Consumption Event Link		A triggers B when it enters s2. B consumes s2 A.	Entering state s2 will attempt to trigger the process once. If B is triggered, it will consume A. Execution will proceed if the triggering failed.
Condition Link		B occurs if A exists.	Existence of object A is a condition to the execution of B. If object A does not exist, then process B is skipped and regular system flow continues.
State-Specified Condition Link		B occurs if A is s1.	Existence of object A at state s2 is a condition to the execution of B. If object A does not exist, then process B is skipped and regular system flow continues.
Invocation Link		B invokes C.	Execution will proceed if the triggering failed (due to failure to fulfill one or more of the conditions in the precondition set).

## Appendix B. The OPM specification of the purchasing process

The **Goods Purchasing** process is the main function of the system. In the OPD of Fig. 3, the **Goods Purchasing** process is in-zoomed. This OPD, like its ancestor, is next explained via its corresponding OPL paragraph below.

**Purchaser**, which is physical and environmental, exhibits **Buyer**. **Purchasing Organization** consists of many **Buyers**. **Purchasing Decision** exhibits **Status** and **Urgency**. **Status** can be **approved**, **vendor selected**, **ordered**, **goods received**, or **completed**. **Urgency** can be **urgent** or **regular**. **Purchaser** handles **Goods Purchasing**.

**Goods Purchasing** requires **Buyer**. **Goods Purchasing** affects **Purchasing Decision**. **Goods Purchasing** zooms into **Full Vendor Selection**, **Automatic Vendor Assignment**, **Order Issuing**, **Goods Receiving**, and **Paying**, as well as **Selected Proposal**. **Full Vendor Selection** requires **regular Urgency** and **Vendor**. **Full Vendor Selection** yields **Selected Proposal** and **Request For Proposals**. **Full Vendor Selection** changes **Status** from **approved** to **vendor selected**. **Automatic Vendor Assignment** requires **urgent Urgency**. **Automatic Vendor Assignment** yields **Selected Proposal**. **Automatic Vendor Assignment** changes **Status** from **approved** to **vendor selected**. **Order Issuing** consumes **Selected Proposal**. **Order Issuing** yields **Purchase Order**. **Order Issuing** changes **Status** from **vendor selected** to **ordered**. **Inventory Worker**, which is environmental and physical, handles **Goods Receiving**. **Goods Receiving** requires **Purchase Order**. **Goods Receiving** yields **Goods Receipt** and **Stocked Goods**. **Goods Receiving** changes **Status** from **ordered** to **goods received**. **Paying** requires **Purchase Order** and **Goods Receipt**. **Paying** changes **Status** from **goods received** to **completed**.

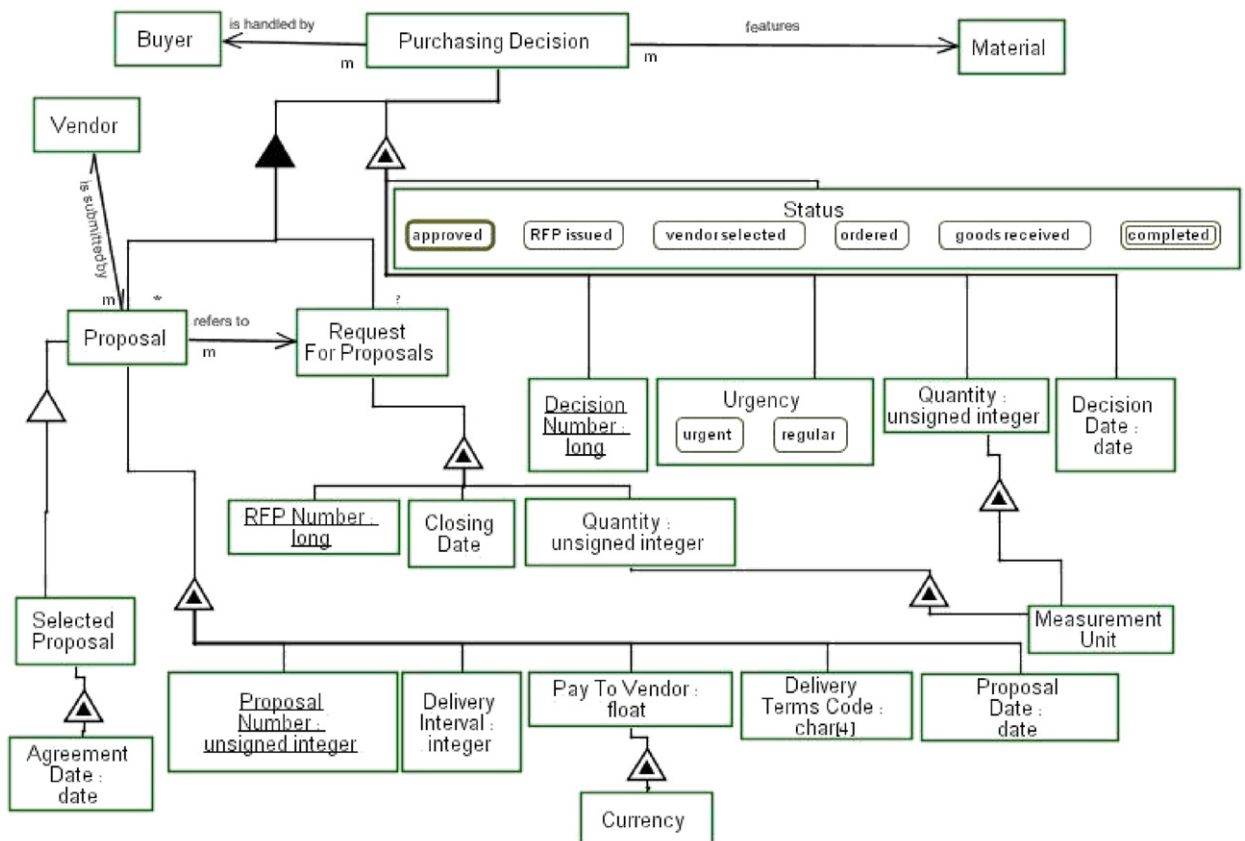


Fig. B1. Unfolding of the **Purchasing Decision** object.

The Full Vendor Selection process is in-zoomed in the OPD of Fig. 4. **Purchaser**, which is physical, exhibits **Buyer**. Purchasing Organization consists of many **Buyers**. **Purchasing Decision** exhibits **Status** and **Urgency**. **Status** can be **approved**, **RFP issued**, **ordered**, and **vendor selected**. **Urgency** can be **urgent** or **regular**. **Purchasing Decision** consists of an optional **Request For Proposals** and optionally many **Proposals**. **Purchaser** handles **Full Vendor Selection**. **Selected Proposal** is a **Proposal**. **Full Vendor Selection** requires **Buyer** and **regular Urgency**. **Full Vendor Selection** zooms into **RFP Issuing**, **Proposals Receiving**, **Negotiating**, and **Proposal Selecting**. **RFP Issuing** changes **Status** from **approved** to **RFP issued**. **RFP Issuing** yields **Request For Proposals**. **Proposals Receiving** requires **Vendor**. **Proposals Receiving** yields **Proposal**. **Negotiating** requires **Vendor**. **Negotiating** affects **Proposal**. **Proposal Selecting** requires **Request For Proposals**. **Proposal Selecting** changes **Status** from **RFP issued** to **vendor selected**. **Proposal Selecting** yields **Selected Proposal**.

Fig. B1 is an OPD in which the **Purchasing Decision** object is unfolded. The OPL paragraph of this OPD follows. **Purchasing Decision** exhibits **Status**, **Urgency**, **Decision Date**, **Quantity**, and **Decision Number**. **Status** can be **approved**, **RFP issued**, **vendor Selected**, **ordered**, **goods received**, or **completed**. **Urgency** can be **urgent** or **regular**. **Quantity** exhibits **Measurement Unit**. **Purchasing Decision** consists of an optional **Request For Proposals** and optionally many **Proposals**. **Request For Proposals** exhibits **RFP Number**, **Closing Date**, and **Quantity**. **Proposal** exhibits **Proposal Number**, **Delivery Interval**, **Pay To Vendor**, **Delivery Terms Code**, and **Proposal Date**. **Pay To Vendor** exhibits **Currency**. **Proposal** refers to **Request For Proposals**. **Proposal** is submitted by **Vendor**. **Buyer** handles **Purchasing Decision**. **Purchasing Decision** features **Material**. **Selected Proposal** is a **Proposal**. **Selected Proposal** exhibits **Agreement Date**.

Figs. B2 and B3 unfold other important objects of our case study.

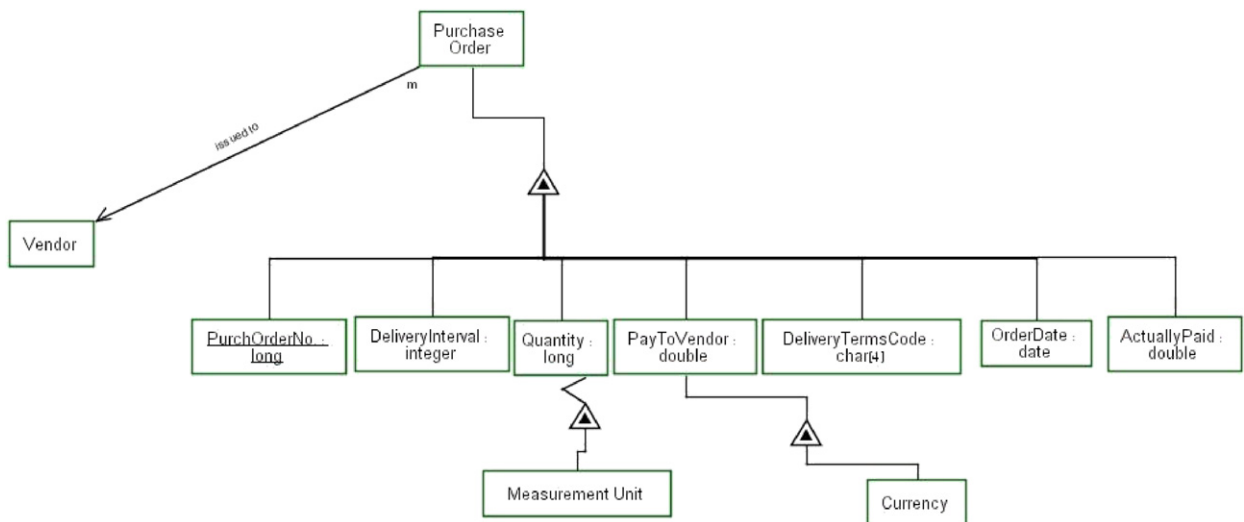


Fig. B2. Unfolding of the **Purchase Order** object.

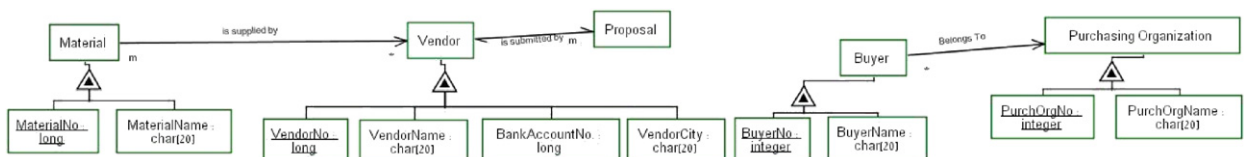


Fig. B3. Unfolding of the **Material**, **Vendor**, and **Buyer** objects.

## Appendix C. ODWC-Constructed Purchase Orders Cubes

Figs. C1 and C2

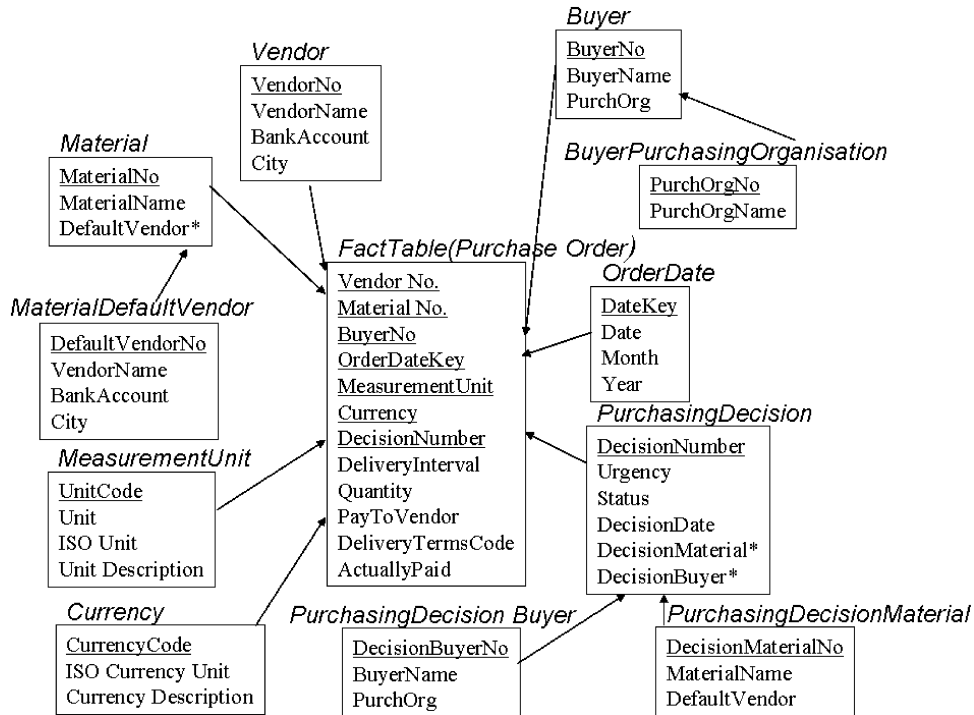
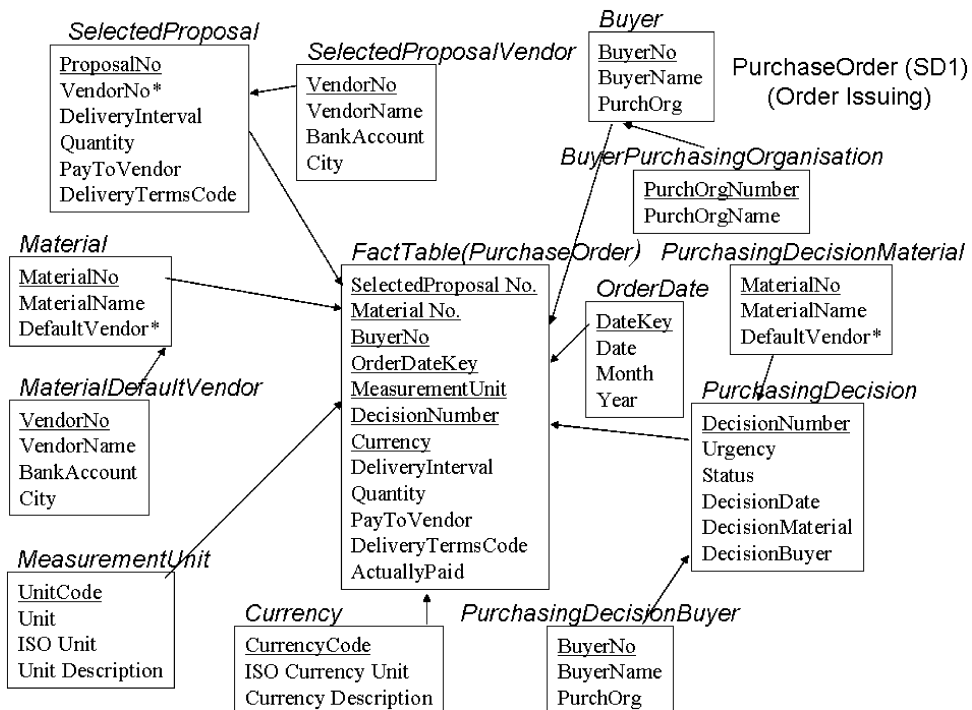
Fig. C1. A Snowflake schema of the **Purchase Orders**, drawn from the **Goods Purchasing** process.

Fig. C2. A Snowflake schema of the Purchase Orders, drawn from the Order Issuing sub process.

## Appendix D. Purchasing cubes construction with the use of a structure-based technique

Figs. D1 and D2 and D3

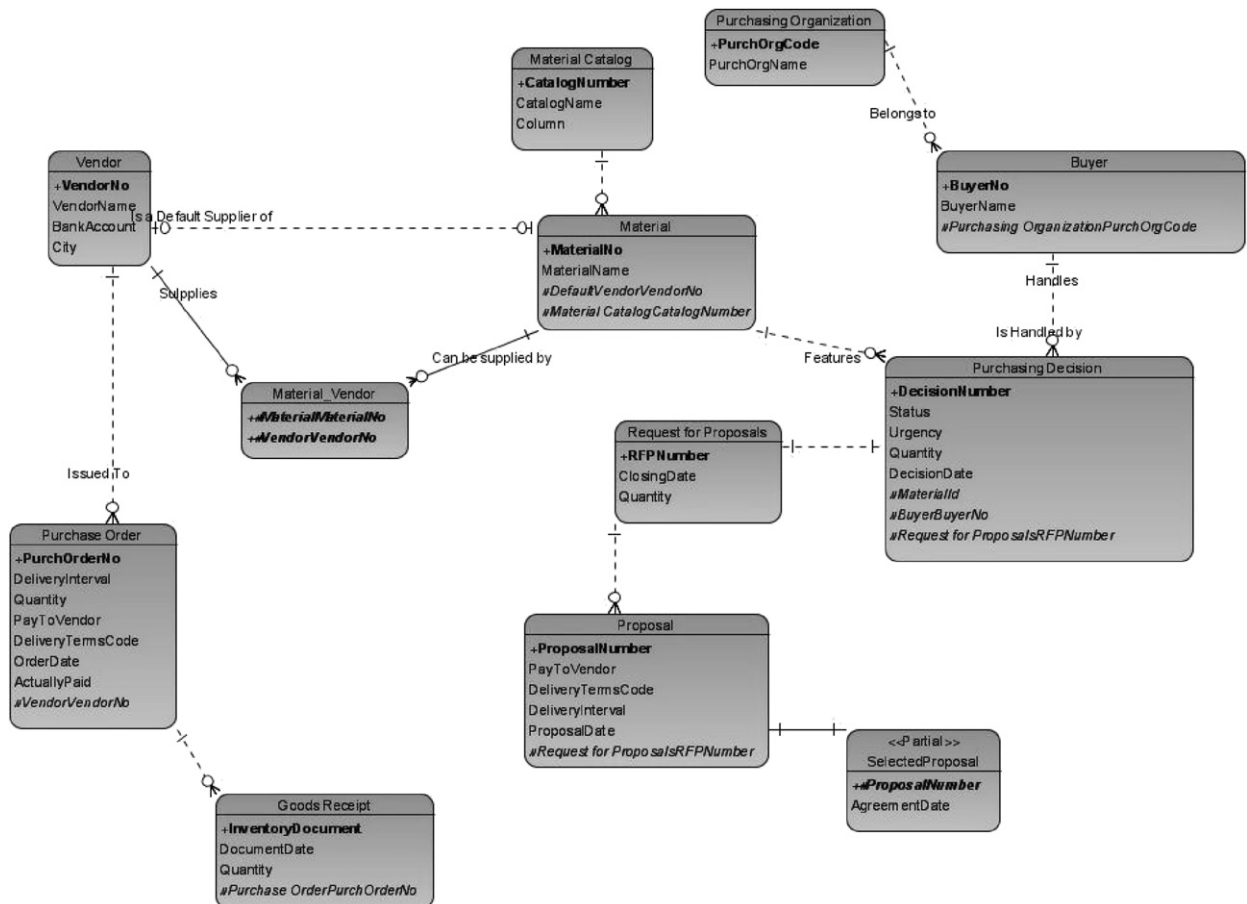


Fig. D1. An Entity-Relationship Diagram featuring the structural relationships of the Purchasing case study.

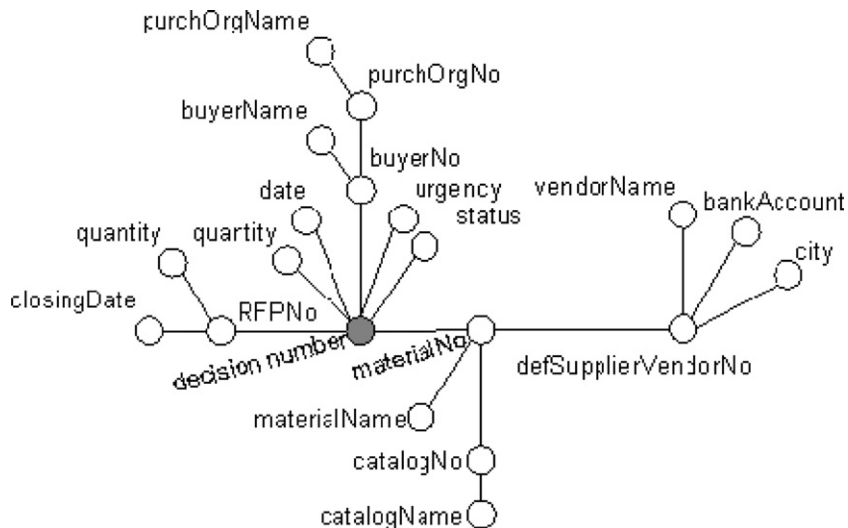


Fig. D2. An Attribute tree corresponding for the Purchasing case study.

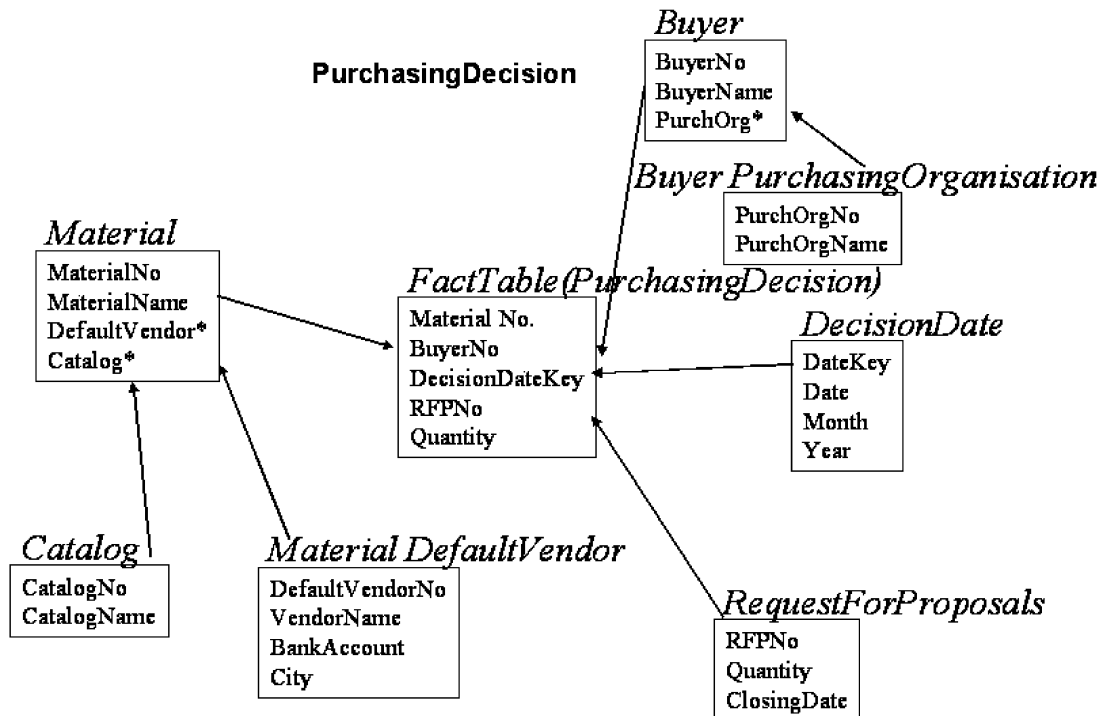


Fig. D3. Multidimensional representation of the Purchasing case study, by the Golfarelli, Maio and Rizzi method.

## References

- [1] M. Golfarelli, S. Rizzi, B. Vrdoljak, Data Warehouse Design from XML Sources, In: Proceedings of DOLAP 2001, Atlanta, USA, 2001, pp. 40–47.
- [2] R. Kimball, The Data Warehouse Toolkit, Wiley, 1996.
- [3] K. Glassey, Business intelligence is a smart move, IEEE IT Professional 1 (5) (1999) 77–79.
- [4] S. Chaudhuri, U. Dayal, An overview of data warehousing and OLAP technology, ACM SIGMOD Record 26 (1) (1997) 65–74.
- [5] M. Golfarelli, D. Maio, S. Rizzi, The Dimensional Fact Model: a conceptual Model for Data Warehouses, Int. J. Cooperative Inform. Syst. 7 (2&3) (1998) 215–247.
- [6] C. Sapia, M. Blaschka, G. Hofling, B. Dinter, Extending the E/R model for the Multidimensional Paradigm, In: Proceedings of ER Workshops 1998, Singapore, 1998, pp. 105–116.
- [7] N. Tryfona, F. Busborg, J. G. B. Christiansen, StarER: A Conceptual model for data warehouse design, in: Proceedings of DOLAP 1999, Washington DC, USA, 1999, pp. 3–8.
- [8] V. Gopalkrishnan, Q. Li, K. Karlapalem, Star/Snow Flake Schema Driven Object-Relational Data Warehouse Design and Query Processing Strategies, Lecture Notes in Computer Science 1676 (1999) 11–22.
- [9] OMG, Unified Modeling Language Home Page, Retrieved From: <[www.uml.org](http://www.uml.org)>, 2006.
- [10] E. Medina, J. Trujillo, A Standard for Representing Multidimensional Properties: The Common Warehouse Metamodel (CWM), In: Proceedings ADBIS 2002. Bratislava, Slovakia, 2002, pp. 232–247.
- [11] J. Trujillo, M. Palomar, J. Gomez, I.Y. Song, Designing Data Warehouses with OO Conceptual Models, IEEE Comput. 34 (12) (2001) 66–75.
- [12] D.T. Chang, J.D. Poole, Extending UML for data warehousing and business common warehouse metamodel, in: Proceedings of the OMG's First Workshop on "UML in the .com Enterprise," Palm Springs, USA. Retrieved from: <<http://www.cwmforum.org/UMLDWBI.pdf>>, 2000.
- [13] J. Poole, D. Chang, D. Tolbert, D. Mellor, Common Warehouse Metamodel—An Introduction to the Standard of Data Warehouse Integration, OMG Press, Wiley, 2002.
- [14] D.L. Moody, M.A.R. Kortink, From enterprise models to dimensional models: a methodology for data warehouse and data mart design, in: Proceedings of DMDW 2000, Stockholm, Sweden, 2000, pp. 5.1–5.12.
- [15] M. Boehnlein, A.U. Ende, Deriving initial data warehouse structures from the conceptual data models of the underlying operational information systems, in: Proceedings of DOLAP, Kansas City, USA, 1999, pp. 15–21.
- [16] M. Golfarelli, D. Maio, S. Rizzi, Conceptual design of data warehouses from E/R schemes, In: Proceedings of 31st HICSS (7), Hawaii, USA, 1998, pp. 334–343.
- [17] M. Golfarelli, S. Rizzi, Designing the data warehouse: key steps and crucial issues, J. Comput. Sci. Inform. Manage. 2 (3) (1999) 1–14.
- [18] B. Husemann, J. Lechtenborger, G. Vossen, Conceptual data warehouse design, in: Proceedings of DMDW 2000, Stockholm, Sweden, 2000, pp. 6.1–6.11.
- [19] C. Phipps, K. Davis, Automating data warehouse conceptual schema design and evaluation, in: Proceedings of DMDW 2002, Toronto, Canada, 2002, pp. 23–32.

- [20] P. Chen, The Entity-relationship model—toward a unified view of data, *QACM Trans. Database Syst.* 1 (1) (1976) 9–36.
- [21] C. Kaldeich, J. Oliviera e Sa, Data warehouse methodology: a process driven approach, in: *Proceedings of CAiSE 2004*, 2004, pp. 536–549.
- [22] B. List, J. Schiefer, A.M. Tjoa, G. Quirchmayr, Multi-dimensional business process analysis with the process warehouse, in: W. Abramowicz, J. Zurada (Eds.), *Knowledge Discovery for Business Information Systems*, Kluwer Academic Publishing, 2000, pp. 211–227.
- [23] M. Boehnlein, A.U. Ende, Business process oriented development of data warehouse structures, in: *Proceedings of Data Warehouse 2000*, Physica Verlag, Friedrichshafen, Germany, 2000, pp. 3–21.
- [24] O.K. Ferstl, E.J. Sinz, SOM modeling of business systems, in: P. Bernus, K. Mertins, G. Schmidt (Eds.), *Handbook on Architectures of Information Systems—International Handbook on Information Systems*, Springer, Berlin, 1998, pp. 339–358.
- [25] H. Kim, T.H. Lee, S. Lee, J. Chun, Automated data warehousing for Rule-based CRM Systems, in: *Proceedings of Australasian Database Conference (ADC) 2003*, Adelaide, Australia, 2003, pp. 67–73.
- [26] D. Dori, R. Feldman, A. Sturm, Transforming an operational system model to a data warehouse model: a survey of techniques, in: *Proceedings of the IEEE International Conference on Software—Science, Technology & Engineering (SwSTE)*, Hertzlia, Israel, 2005, pp. 47–56.
- [27] D. Dori, *Object-Process Methodology: a Holistic Systems Paradigm*, Springer Press, 2002.
- [28] D. Dori, I. Reinhartz-Berger, A. Sturm, OPCAT—a bimodal CASE tool for object-process based system development, *Proceedings of ICEIS (3) (2003)* Angers, France.
- [29] SAP, SAP BW Business Warehouse—Introduction, retrieved from: <[http://www.thespot4sap.com/Articles/SAP\\_BW\\_Introduction.asp](http://www.thespot4sap.com/Articles/SAP_BW_Introduction.asp)>, 2006.
- [30] OMG, XML Metadata Interchange (XMI) Specification, retrieved from: <<http://www.omg.org/technology/documents/formal/xmi.htm>>, 2005.
- [31] M. Golfarelli, S. Rizzi, I. Cella, Beyond data warehousing: what's next in business intelligence?, in: *Proceedings of DOLAP 2004*, Washington DC, USA, 2004, pp. 1–6.
- [32] E.J. Sinz, Das Strukturierte Entity-Relationship-Modell (SER-Modell), *Ang. Inform.* 30 (5) (1988) 191–202.