A Comprehensive Framework on Multidimensional Modeling

Oscar Romero and Alberto Abelló

Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
{aabello,oromero}@essi.upc.edu

Abstract. In this paper we discuss what current multidimensional design approaches provide and which are their major flaws. Our contribution lays in a comprehensive framework that does not focus on *how* these approaches work but *what* they do provide for usage in real data warehouse projects. So that, we do not aim at comparing current approaches but set up a framework (based on four criteria: the role played by end-user requirements and data sources, the degree of automation achieved and the quality of the output produced) highlighting their drawbacks, and the need for further research on this area.

1 Introduction

Developing a data warehousing system is never an easy job, and raises up some interesting challenges. One of these challenges focuses on modeling multidimensionality. OLAP tools are conceived to exploit the data warehouse for analysis tasks based on the multidimensional (MD) paradigm and therefore, the data warehouse must be structured according to the MD model. Lots of efforts have been devoted to MD modeling, and several models and design methods have been developed and presented in the literature. Consequently, we can nowadays design a MD conceptual schema, create it physically and later, exploit it through the model algebra or calculus (implemented in the exploitation tools).

MD modeling was first introduced by Kimball in [9]. Kimball's approach was well received by the industry and also introduced the first method to derive the data warehouse *logical* schema. Similar to traditional information systems modeling, Kimball's method is *requirement-driven*: it starts eliciting business requirements of an organization and through a step-by-step guide we are able to derive the MD schema. Only at the end of the process data sources are considered to map data from sources to target.

In short, Kimball's approach follows a traditional modeling approach (i.e., from requirements), but it set down the principles of MD modeling. MD modeling is radically opposite to OLTP systems modeling: the data warehouse conceptual schema is directly derived from the organization operational sources and provides a single, detailed, integrated and homogenized view of the business domain. Consequently, the data warehouse can be thought as a strategic view of the

O. De Troyer et al. (Eds.): ER 2011 Workshops, LNCS 6999, pp. 108–117, 2011.

organization data and for this reason, and unlike most information systems that are designed from *scratch*, the organization data sources must be considered as first-class citizens in the data warehouse design process. This major additional requirement has such interesting consequences so much so that it gave rise to a new research topic and up to now, several MD modeling methods have been introduced in the literature. With the perspective of time, we may now highlight those features that drew the attention of the community. The evolution of the modeling methods introduced in the literature pivots on a crucial aspect: the dichotomy requirements versus data sources (and how to deal with it).

In this paper we discuss what current approaches provide and which are their major flaws. Our contribution lays in a comprehensive framework that does not focus on *how* these approaches work but *what* they do provide. Importantly, note that by no means we aim at comparing current approaches but providing a comprehensive, general picture on MD modeling and identify what is (yet) missing on this area. The criteria used for this analysis can be summarized as follows: the role played by end-user requirements and data sources for each method, the degree of automation achieved and the quality of the output produced (i.e., which MD concepts and features do they really consider). The use of this criteria is justified by the conclusions drawn by a previous, exhaustive analysis of current design methods that can be found in [17].

The paper is structured as follows. Section 2 summarizes briefly our previous research on MD design and highlights how this area evolved with time. Next, Section 3 provides a detailed, comprehensive discussion of what can be achieved by using current approaches in real projects. We wrap up the discussion pointing out the main flaws that still need to be addressed in order to better support the data warehouse design.

2 Multidimensional Modeling

In this section we introduce the background of MD modeling. Our objective here is to provide an insightful view of how this area evolved with time. The interested reader is addressed to [17] for details.

Shortly after Kimball introduced his ad hoc modeling method for data warehouses [10], some other methods were presented in the literature (e.g., [4,6,2,7,12]). Like Kimball's method, these methods were originally regarded as step-by-step guides to be followed by a data warehouse expert who start gathering the end-user requirements. However, unlike Kimball's work, they give more and more relevance to the data sources. Involving the data sources in these approaches means that it is compulsory to have well-documented data sources (e.g., with up-to-date conceptual schemas) at the expert's disposal but it also entailed two main benefits: on the one hand, the user may not know all the potential analysis contained in the data sources and analyzing them we may find unexpected potential analysis of interest to the user; on the other hand, we should assure that the data warehouse can be populated with data available within the organization.

As said, to carry out these approaches manually it is compulsory to have well-documented data sources, but in a real organization, the data sources

documentation may be incomplete, incorrect or may not even exist [6] and, in any case, it would be rather difficult for a non-expert designer to follow these guidelines. Indeed, when automating this process is essential not to depend on the expert's ability to properly apply the method chosen and to avoid the tedious and time-consuming task (even unfeasible when working over large databases) of analyzing the data sources.

In order to solve these problems, several new methods automating the design task were introduced in the literature [14,21,8]. These approaches work directly over relational database logical schemas. Thus, despite they are restricted to a specific technology, they get up-to-date data that can be queried and managed by computers. They also argue that restricting to relational technology makes sense since nowadays it is the most widely used technology for operational databases. About the process carried out, these methods follow a *data-driven* process focusing on a thorough analysis of the data sources to derive the data warehouse schema in a reengineering process. This process consists of techniques and design patterns that must be applied over the data sources schema to identify data likely to be analyzed from a MD perspective.

Nevertheless, a requirement analysis phase is crucial to meet the user needs and expectations [3,5,22,15,11,1]. Otherwise, the user may find himself frustrated since he / she would not be able to analyze data of his / her interest, entailing the failure of the whole system. Today, it is assumed that the ideal scenario to derive the data warehouse conceptual schema embraces a hybrid approach (i.e., a combined data-driven and requirement-driven approach) [22]. Then, the resulting MD schema will satisfy the end-user requirements and it will have been conciliated with the data sources simultaneously.

According to [22], MD modeling methods may be classified within a *demand-driven*, a *supply-driven* or a *hybrid* framework:

- Supply-driven approaches (SDAs): Also known as *data-driven*, start from a detailed analysis of the data sources to determine the MD concepts in a reengineering process.
- Demand-driven approaches (DDAs): Also known as *requirement-driven* or *goal-driven*, focus on determining the user MD requirements (as typically performed in other information systems) to later map them onto data sources.
- Hybrid approaches: Combine both paradigms to design the data warehouse from the data sources but bearing in mind the end-user requirements.

Each paradigm has its own advantages and disadvantages. Carrying out an exhaustive search of dimensional concepts among all the concepts of the domain (like SDAs do) has a main benefit with regard to those approaches that derive the schema from requirements and later conciliate them with the data sources (i.e., DDAs): in many real scenarios, the user may not be aware of all the potential analysis contained in the data sources and, therefore, overlook relevant knowledge. Demand-driven and current hybrid approaches do not consider this and assume that requirements are exhaustive. Thus, knowledge derived from the sources not depicted in the requirements is not considered and discarded.

As a counterpart, SDAs tend to generate too many results (since they overlook the MD requirements, they must apply their design patterns all over the data sources) and mislead the user with non relevant information. Furthermore, DDAs (or demand-driven stages within a hybrid approach) are not automated whereas supply-driven stages tend to facilitate their automation. The main reason is that demand-driven stages would require to formalize the end-user requirements (i.e., translate them to a language understandable by computers). Unfortunately, most current methods handle requirements mostly stated in languages (such as natural language) lacking the required degree of formalization. Thus, matching requirements over the data sources must be performed manually. However, the time-consuming nature of this task can render it unfeasible when large databases are used.

In general, most approaches do not automate the process and just present a set of steps (i.e., a guideline) to be followed by an expert in order to derive the MD schema. Mainly, these methods introduce different patterns or heuristics to discover concepts likely to play a MD role and to carry out these approaches manually it is compulsory to have well-documented data sources at the expert's disposal. This prerequisite is not easy to fulfill in many real organizations and in order to solve this problem, current automatable methods directly work over relational databases (i.e., getting up-to-date data). To our knowledge, only three exceptions exist to this rule [20,19,13], which automate the process from ER schemas (the first one) and ontologies (the other two). Consequently, all these methods (or stages within hybrid approaches) follow a supply-driven paradigm and thus, rely on a thorough analysis of the sources.

All in all, *DDAs assume that requirements are exhaustive*, whereas *SDAs rely on discovering as much MD knowledge as possible*. As a consequence, *SDAs generate too many results*. Furthermore, *current automatable methods follow a SDA*, whereas *current DDAs overlook the process automation*, since they tend to work with requirements at a high level of abstraction. Finally, all *current hybrid approaches follow a sequential approach with two well-differentiated steps*: the supply-driven and the demand-driven stages. Each one of these stages, however, suffers from the same drawbacks as pure SDAs or DDAs do.

2.1 The State of the Art in a Nutshell

Previous experiences in the data warehouse field have shown that the data warehouse MD conceptual schema must be derived from a hybrid approach: i.e., by considering both the end-user requirements and the data sources, as first-class citizens. Like in any other system, requirements guarantee that the system devised meets the end-user necessities. In addition, since the data warehouse design task is a reengineering process, it must consider the underlying data sources of the organization: (i) to guarantee that the data warehouse must be populated from data available within the organization, and (ii) to allow the end-user discover unknown additional analysis capabilities.

Nowadays, we may find several methods for supporting the data warehouse conceptual design but, all of them, start from very different assumptions that

make them hardly comparable. For example, some approaches claim to fully automate the design task, but they do so by overlooking the end-user requirements in a fully SDA (and thus, making the user responsible for manually filtering the results obtained according to his / her needs). Similarly, exhaustive DDAs claim to derive high-quality outputs, but they completely overlook the task automation. For this reason, every approach fits to a narrow-ranged set of scenarios and do not provide an integrated solution for every real-world case, which, in turn, makes the data warehouse designers to come up with ad hoc solutions for each project. For example, we cannot follow the same approach in a scenario where the end-user requirements are clear and well-known, and in a scenario in which the end-user requirements are not evident or cannot be easily elicited (e.g., this may happen when the users are not aware of the analysis capabilities of their own sources). Clearly, this is the major flaw of current approaches, which do not suit well for the wide range of real projects a designer could meet. Interestingly, it has already been pointed out [18] that, given a specific design scenario, the necessity to provide requirements beforehand is smoothed by the fact of having semantically rich data sources. In lack of that, requirements gain relevance to extract the MD knowledge from the sources. In both cases, we can still achieve an acceptable degree of automation and output quality, as discussed later on.

3 The Big Picture

To overcome the situation above discussed, we aim to establish a clear framework in which to place the most relevant design methods introduced in the literature. This comprehensive picture of the state of the art will help to identify the major drawbacks of current approaches. As earlier introduced in this paper, the data warehouse design task must consider (i) the end-user requirements and (ii) the data sources. Furthermore, we also aim to analyze the (iii) automation degree achieved and (iv) the quality of the output produced. In the following, we rate the most relevant methods introduced in the literature with regard to these four criteria. However, we do not intend to classify nor rank approaches (for a detailed comparison on MD design methods, see [17]) but to identify, at first sight, the assumptions made by each kind of approach and moreover, analyze the consequences of the process proposed regarding the automation degree achieved and the quality of the outputs produced. In short, identify trends and flaws that should be addressed in the future.

Consider Figures 1 and 2. Axes x and y represent the assumptions of each method; i.e., the use of the requirements and the data sources in each approach. The x axis measures how important requirements are in the approach, and if the method proposes to formalize them somehow, to facilitate their analysis. The y axis assesses how important the analysis of the data sources is for the approach, and if detailed patterns are provided to exploit them. Finally, axes z measure either the automation degree achieved (see Figure 1) and the quality of the output produced (see Figure 2) regarding the assumptions made by the method (i.e., axes x and y). In the 3D-space formed, every approach is identified

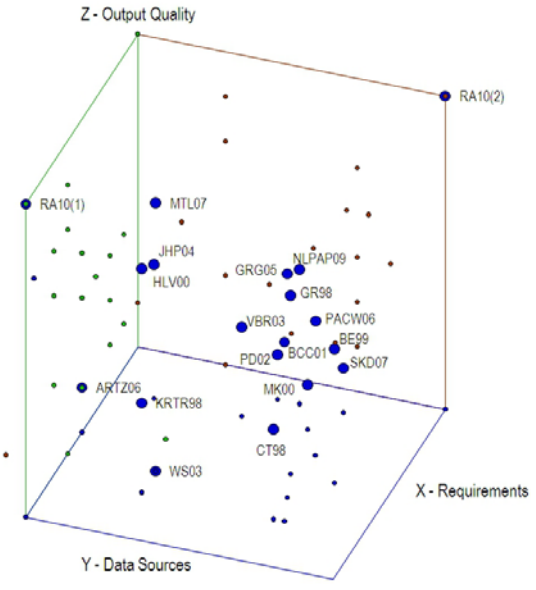

Fig. 1. Output quality analysis

as a rhombus labeled with the first initial of each author plus the year of the bibliographical item it represents. Furthermore, for the sake of understandability, we provide the projection of each rhombus in the three planes (green points for the XZ plane projections; blue points for the XY plane and red points for the XZ plane). Each approach is placed in the 3D-space according to the conclusions extracted from [17]. The first conclusion is that the duality requirements / data-sources is clearly shown in both figures, as SDAs and DDAs are placed in opposite axis (to better appreciate it, check the plane projections of each point).

Requirements Specificity: DDAs integrate the end-user requirements, which lead the whole process by exploiting the knowledge of the requirements. Therefore, the quality and expressiveness of the input requirements must be high. On the contrary, at the beginning of the process, SDAs do not need the end-user requirements to work. Indeed, results provided by SDAs, are eventually shaped by the end-user needs a posteriori. In this latter step, the end-user just state his / her requirements by choosing his / her concepts of interest regarding the results provided. Therefore, the user would even be able to state them on-the-fly regarding the output presented to the user by SDAs.

Data Source Expresiveness: SDAs lead the process from a thorough analysis of the data sources and, in general, they ask for high quality inputs capturing the data sources (i.e., relational schema, ER diagram, domain ontology, etc.). In case of inputs at the conceptual level, a mapping between the conceptual schema and the sources as well as means to access the data sources at the instance level are also required. Regarding DDAs, the quality of the inputs is not that relevant

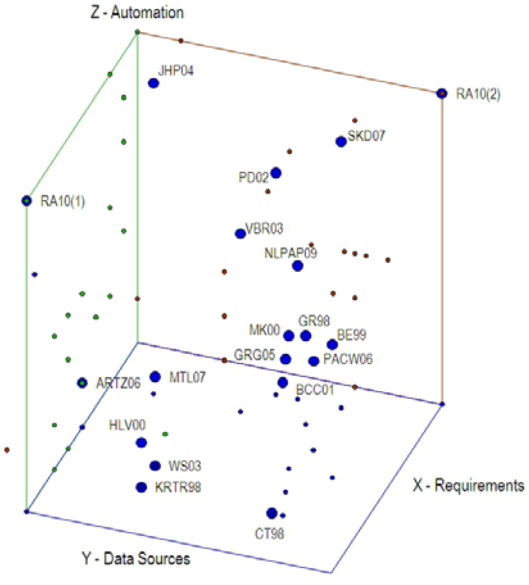

Fig. 2. Automation analysis

given that the requirements provide the lack of semantics captured in the sources. Thus, they could even handle not well-formed data sources (e.g., denormalized sources).

Automation: The first figure shows that the automation degree achieved, in the general case, is medium or low. Only 6 approaches automate the design task up to a fair degree. Regarding DDAs, new techniques to automatically manipulate requirements during the design phase are needed. In this sense, [16] sets a basis on this direction but it can only deal with relational sources. Oppositely, SDAs achieve an interesting degree of automation, but most of them happen not to be useful in practice due to the big set of assumptions made. For example, the kind of sources (normally, only relational sources are allowed but this is clearly unsatisfactory nowadays, where unstructured data and the Web is a relevant source of knowledge) or additional ones (such as relational schemas in, at least, 3NF). Furthermore, filtering techniques based on objective evidences are a must. SDAs tend to generate too many results. Consequently, they unnecessarily overwhelm users with blindly generated combinations whose meaning has not been analyzed in advance. Eventually, they put the burden of (manually) analyzing and filtering results provided onto the designer's shoulder, but the time-consuming nature of this task can render it unfeasible when large data sources are considered. To our knowledge, only [19] filters out results obtained prior to show the results to the user. Furthermore, all these facts directly affect the computational complexity of SDAs.

Quality Output: Both SDAs and DDAs are able to extract valuable knowledge from the requirements / data sources but only a few of them deal with concepts

such as *factless facts*, *MD space definitions*, *aggregate measures* and *semantic relationships* between the MD concepts identified. However, this is not tied to the kind of framework used but a decision made by the authors. Indeed, both can produce quality outputs if the appropriate inputs are provided. This is obviously sound, since, given the same scenario, we would expect to obtain the same result regardless of the approach chosen. The difference between both approaches is just on *how* we obtain the final result depending on our current inputs and if they provide enough semantics. Thus, DDAs are exhaustive regarding requirements so in case of disposing of quality requirements a DDA suits better. However, if requirements are not clear but we dispose of quality data sources then a SDA is mandatory. For example, DDAs can compute *derived measures* or *concept specializations* not explicitly captured in the data sources, but present in the end-user requirements. However, this kind of measures and specializations can only be identified by SDAs if they are captured in the input representation of the data sources (and thus, only if they are of enough quality). Analogously, this also happens regarding *semantic relationships* between MD concepts, *aggregate measures* and *MD space definitions*. The case of the *factless facts*, however, is slightly different. DDAs might identify them by means of requirements, but in SDAs, some kind of *quality function* used to identify facts would be needed [19].

4 Discussion and Conclusions

Several methods for supporting the data warehouse modeling task have been provided. However, they suffer from some significant drawbacks, which need to be addressed. In short, DDAs assume that requirements are exhaustive (and therefore, do not consider the data sources to contain alternative interesting evidences of analysis), whereas SDAs (i.e., those leading the design task from a thorough analysis of the data sources) rely on discovering as much MD knowledge as possible from the data sources. As a consequence, SDAs generate too many results, which misleads the user. Furthermore, the design task automation is essential in this scenario, as it removes the dependency on an expert's ability to properly apply the method chosen, and the need to analyze the data sources, which is a tedious and time-consuming task (which can be unfeasible when working with large databases). In this sense, current automatable methods follow a SDA, whereas current DDAs overlook the process automation, since they tend to work with requirements at a high level of abstraction. Indeed, this scenario is repeated regarding SDA and DDA stages within current hybrid approaches, which suffer from the same drawbacks than pure DDA and SDA approaches.

Consequently, previous experiences in this field have shown that the data warehouse MD conceptual schema must be derived from a truly hybrid approach: i.e., by considering both the end-user requirements and the data sources, as first-class citizens. Currently, several methods (i.e., detailed design approaches) and dissertations (i.e., high level discussions highlighting the necessities in each real scenario) for supporting the data warehouse design task have been introduced in the literature, but none of them provides an integrated and automated solution

embracing both aspects. On the one hand, dissertations about how the design task must be adapted to every real-world scenario provide an insightful idea of how to proceed in each case. However, they fail to provide detailed algorithms to undertake this task (thus, ad hoc solutions are needed). On the other hand, detailed methods introduced tend to focus on a narrow-ranged set of scenarios. For example, today, it is assumed that the approach to follow in a scenario where the end-user requirements are clear and well-known is completely different from that in which the end-user requirements are not evident or cannot be easily elicited (for example, this may happen when the users are not aware of the analysis capabilities of their own sources). Similarly, the necessity to provide requirements beforehand is smoothed by the fact of having semantically rich data sources. In lack of that, requirements gain relevance to extract the MD knowledge from the sources. Indeed, a combined and comprehensive framework to decide, according to the inputs provided in each scenario, which is the best approach to follow, is missing. This framework should be built considering that:

- If the end-user requirements are well-known beforehand, we can benefit from the knowledge captured in the data sources, but we should guide the design task according to requirements and consequently, we will be able to work and handle semantically poorer data sources. In other words, providing high-quality end-user requirements, we can guide the process and overcome the fact of disposing of bad quality (from a semantical point of view) data sources.
- As a counterpart, a scenario in which the data sources available are semantically richer, the approach should be guided by a thorough analysis of the data sources, which eventually will be properly adapted to shape the output result and meet the end-user requirements. In this context, disposing of high-quality data sources we can overcome the fact of lacking of very expressive end-user requirements.

Acknowledgements. This work has been partly supported by the Ministerio de Ciencia e Innovación under project TIN2008-03863.

References

1. Annoni, E., Ravat, F., Teste, O., Zurfluh, G.: Towards multidimensional requirement design. In: DaWaK 2006. LNCS, vol. 4081, pp. 75–84. Springer, Heidelberg (2006)
2. Böhnlein, M., vom Ende, A.U.: Deriving Initial Data Warehouse Structures from the Conceptual Data Models of the Underlying Operational Information Systems. In: Proc. of 2nd Int. Wksp on Data Warehousing and OLAP, pp. 15–21. ACM, New York (1999)
3. Bonifati, A., Cattaneo, F., Ceri, S., Fuggetta, A., Paraboschi, S.: Designing Data Marts for Data Warehouses. ACM Trans. Soft. Eng. Method 10(4), 452–483 (2001)
4. Cabibbo, L., Torlone, R.: A Logical Approach to Multidimensional Databases. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 183–197. Springer, Heidelberg (1998)

5. Giorgini, P., Rizzi, S., Garzetti, M.: Goal-oriented Requirement Analysis for Data Warehouse Design. In: Proc. of 8th Int. Wksp on Data Warehousing and OLAP, pp. 47–56. ACM Press, New York (2005)
6. Golfarelli, M., Maio, D., Rizzi, S.: The Dimensional Fact Model: A Conceptual Model for Data Warehouses. Int. Journal of Cooperative Information Systems 7(2-3), 215–247 (1998)
7. Hüsemann, B., Lechtenbörger, J., Vossen, G.: Conceptual Data Warehouse Modeling. In: Proc. of 2nd Int. Wksp on Design and Management of Data Warehouses, p. 6. CEUR-WS.org (2000)
8. Jensen, M.R., Holmgren, T., Pedersen, T.B.: Discovering Multidimensional Structure in Relational Data. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, pp. 138–148. Springer, Heidelberg (2004)
9. Kimball, R.: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, Inc., Chichester (1996)
10. Kimball, R., Reeves, L., Thornthwaite, W., Ross, M.: The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses. John Wiley & Sons, Inc., Chichester (1998)
11. Mazón, J., Trujillo, J., Lechtenborger, J.: Reconciling Requirement-Driven Data Warehouses with Data Sources Via Multidimensional Normal Forms. Data & Knowledge Engineering 23(3), 725–751 (2007)
12. Moody, D., Kortink, M.: From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. In: Proc. of 2nd Int. Wksp on Design and Management of Data Warehouses. CEUR-WS.org (2000)
13. Nebot, V., Llavori, R.B., Pérez-Martínez, J.M., Aramburu, M.J., Pedersen, T.B.: Multidimensional integrated ontologies: A framework for designing semantic data warehouses. J. Data Semantics 13, 1–36 (2009)
14. Phipps, C., Davis, K.C.: Automating Data Warehouse Conceptual Schema Design and Evaluation. In: Proc. of 4th Int. Wksp on Design and Management of Data Warehouses., vol. 58, pp. 23–32. CEUR-WS.org (2002)
15. Prat, N., Akoka, J., Comyn-Wattiau, I.: A UML-based Data Warehouse Design Method. Decision Support Systems 42(3), 1449–1473 (2006)
16. Romero, O., Abelló, A.: Automatic Validation of Requirements to Support Multidimensional Design. Data & Knowledge Engineering 69(9), 917–942 (2010)
17. Romero, O., Abelló, A.: A Survey of Multidimensional Modeling Methodologies. Int. J. of Data Warehousing and Mining 5(2), 1–23 (2009)
18. Romero, O.: Automating the Multidimensional Design of Data Warehouses. Ph.D. thesis, Universitat Politécnica de Catalunya, Barcelona, Spain (2010), http://www.tdx.cat/handle/10803/6670
19. Romero, O., Abelló, A.: A Framework for Multidimensional Design of Data Warehouses from Ontologies. Data & Knowledge Engineering 69(11), 1138–1157 (2010)
20. Song, I., Khare, R., Dai, B.: SAMSTAR: A Semi-Automated Lexical Method for Generating STAR Schemas from an ER Diagram. In: Proc. of the 10th Int. Wksp on Data Warehousing and OLAP, pp. 9–16. ACM, New York (2007)
21. Vrdoljak, B., Banek, M., Rizzi, S.: Designing Web Warehouses from XML Schemas. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2003. LNCS, vol. 2737, pp. 89–98. Springer, Heidelberg (2003)
22. Winter, R., Strauch, B.: A Method for Demand-Driven Information Requirements Analysis in DW Projects. In: Proc. of 36th Annual Hawaii Int. Conf. on System Sciences, pp. 231–239. IEEE, Los Alamitos (2003)