RC 22044 (98938) 24 April 2001 **Computer Science**

IBM Research Report

Prototyping Data Warehouse Systems

Thanh N. Huynh
Institute of Software Technology
Vienna University of Technology

Josef Schiefer
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Prototyping Data Warehouse Systems

Thanh N. Huynh
thanh@ifs.tuwien.ac.at
Institute of Software Technology
Vienna University of Technology

Josef Schiefer
josef.schiefer@us.ibm.com
Watson Research Center, IBM

Abstract. The process of developing and evolving complex data warehouse systems is intrinsically exploratory in nature. Data warehouse projects are notoriously difficult to manage and many of them end in failure. One explanation for the high failure rate is the lack of understanding of the requirements and missing proof-of-concepts for a decision support for the knowledge workers of an organization. In this paper, we present a prototyping approach for data warehouse environments. We show how prototyping can support an incremental and iterative requirement development. We introduce different types of prototypes and illustrate their applicability for data warehouse systems. Finally, we show how to develop data warehouse prototypes with the BEDAWA tool, which supports a rapid development of prototypes by automatically generating the mass sample data for a data warehouse system.

1. Introduction

Prototypes can be used for data warehouse systems to explain and answer open questions of stakeholders. The idea behind prototyping is to cut down on the complexity of implementation by eliminating parts of the full system. It is a technique to reduce the risks of failed data warehouse implementation or stakeholder dissatisfaction. A prototype can be used as early feedback from the users ensuring that the data warehouse team properly understands the requirements and knows how best to implement them.

Even with extensive requirement elicitation, analysis, and documentation practices, some portions of the data warehouse requirements might still be uncertain or unclear to either the stakeholders or the data warehouse team. If these problems are not corrected, an expectation gap between the stakeholders will result. Prototyping allows the envisioned data warehouse system to become tangible, brings use cases to life, and closes the gaps to the different understanding of requirements. Furthermore, users are generally more willing to try out a prototype than to read the complete requirements specification.

Any prototype is designed to focus on only certain aspects of the system, compromising or ignoring other aspects. There are three major dimensions of compromises *system functionality, system attributes, and user interface* [4].

Data warehouse prototypes can focus on one or more of these dimensions, while ignoring the others. For instance, some prototypes focus purely on performance

(which represents a system attribute), ignoring user interface and functionality issues. Other prototypes might be developed to discover an optimal user interface for executives, approximating other aspects only to the degree needed to allow it to serve as driver for sponsorship.
The remainder of this paper is organized as follows. In section 2, we discuss the contribution of this paper and related work. In section 3, we present prototyping activities for data warehouse environments and shows how they are interwoven with the requirement process. In section 4, we discuss different types of prototypes. In section 5, we introduce the BEDAWA prototyping tool and show, how it can be used to generate representative sample data for data warehouse systems. Finally, in section 6 we present our conclusion.

2. Contribution and Related Work

Floyd provides in [3] a high-level characterization of prototyping approaches, tools and techniques, and a discussion of problems and prospects of prototyping. Floyd's high-level characterization includes the distinction between horizontal and vertical prototyping. His characterization of prototyping approaches is based on a distinction between exploratory, experimental and evolutionary prototyping.

In this paper, we discuss different type of prototypes for data warehouse systems by using Floyd's prototyping characterization. Furthermore, we also introduce the idea of using prototyping scenarios, which are a combination of horizontal and vertical prototypes.

[17] argues that using measurable objectives (benchmarks) for user performance and attitudes provide an objective way to assess the quality of the prototype. In [5], Huynh proposes a tool, namely BEDAWA, which is able to generate statistical sound, familiar, complete sample data for data warehouse and OLAP systems. The tool allows building sample data for benchmarking applications or various types of prototypes.

[1] introduces the DBGEN tool, which can be used to generate sample data for AS3AP or TPC-D benchmark in flat files. [16] and [7] present an approach for generating sample data to evaluate database systems. However, the presented approach generates sample data manually, which makes it less useful for prototyping purposes.

[10] introduces the easyREMOTE^{DWH} approach, a comprehensive requirements engineering framework for data warehouse systems. easyREMOTE^{DWH} includes a requirements process, which allows an iterative and incremental requirements development. We use this model as foundation and show, how prototyping activities are interwoven with other requirement activities.

To our knowledge, no approach for prototyping has been published yet, which considers the characteristics of data warehouse systems (large, evolving system, heterogeneous user community, complex data structures, large amount of data etc.).

Traditional prototyping tools lack the ability to support the generation of complex data schemas and large amounts of data, which is typical for data warehouse environments.

3. Requirements Process and Prototyping Activities

In this section we discuss, how prototyping activities are interwoven with the requirements process. Figure 1 shows an extract of the requirement activities in the easyREMOTE^{DWH} process [10], and how they are linked by their deliverables. The deliverables are shown as moving from one activity to the next. In the "Requirements Gathering, Elicitation" step data warehouse requirements are discovered, which are made rigorous by writing them down in a prescribed manner. Afterwards, the requirements are inspected in the "Requirements Refinement" to determine if they are accurate enough to be added to the final requirement specification draft. Any rejects are sent back to the originator, who probably takes the requirements back to the requirements gathering and elicitation activity for clarification and further explanation.

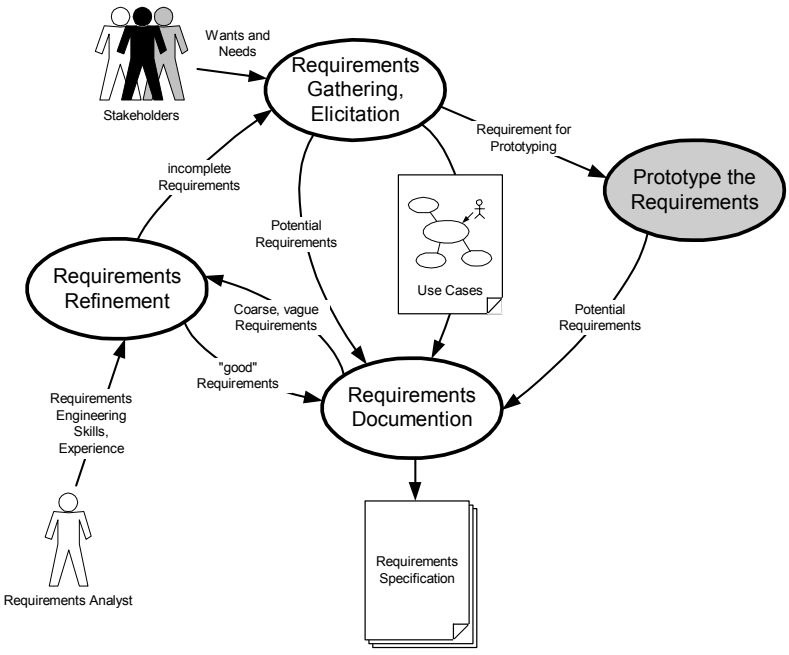

Figure 1: Requirements Process for Data Warehouse Systems (adapted from [10])

If requirements are still uncertain or unclear to either the stakeholders or the data warehouse implementation team, the development of prototypes can facilitate the management of stakeholder expectations (e.g. by a proof-of-concept) or technical

concerns (e.g. by a proof-of-performance). One key point in Figure 1 is that the requirements process is iterative and incremental rather than linear. The elicitation, prototyping, refinement, and documentation activities are iterated a number of times. During each iteration new information emerges which may necessitate the modification of already acquired information.

Figure 2 shows the different stages of developing a prototype of a data warehouse system. The input for the prototyping might be a single requirement or a complete use case.

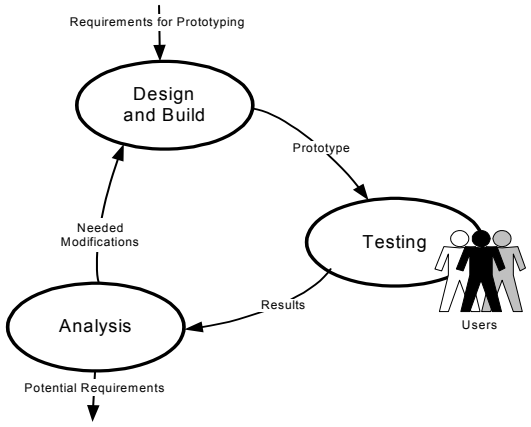

Figure 2: Prototyping Activities

Design and Build. Designing is mapping the world of the user into the prototype. Design is also the activity of deciding which data warehouse requirements should be modeled and simulated with the prototype, and what should be achieved with the prototype. Furthermore it is important to determine, which type of prototype should be built. After the design stage the data warehouse implementation team builds the modeled prototype.

Testing in the User Environment. Testing involves having the data warehouse users use the prototype as a simulation of their tasks. Users will give feedback of the prototype, where they describe their problems and experience during their work with the prototype. The feedback should also include the results of usability tests.

Analyzing the Results. In this stage the feedback of the users is analyzed. The analysis will uncover new potential requirements for the data warehouse system. If the results are not satisfying, modifications of the prototype and running more tests should be considered.

4. Prototyping Types

4.1 Horizontal vs. Vertical Prototypes

Horizontal prototypes are prototypes where the user-interface is implemented, but the functionality is missing, or simulated. Horizontal prototypes reduce the level of functionality and result in a user interface surface layer, while vertical prototypes reduce the number of features and implement the full functionality of those chosen (see Figure 3) [3].

Horizontal Prototypes. Horizontal prototypes display the facades of user interface screens from the data warehouse system, possibly allowing some navigation between them, but they do not show real data or contain little or no real functionality. The information that appears in response to a data warehouse query is faked or static, and report contents are hard-coded. A horizontal prototype does not actually perform any useful work, although it looks like it does.

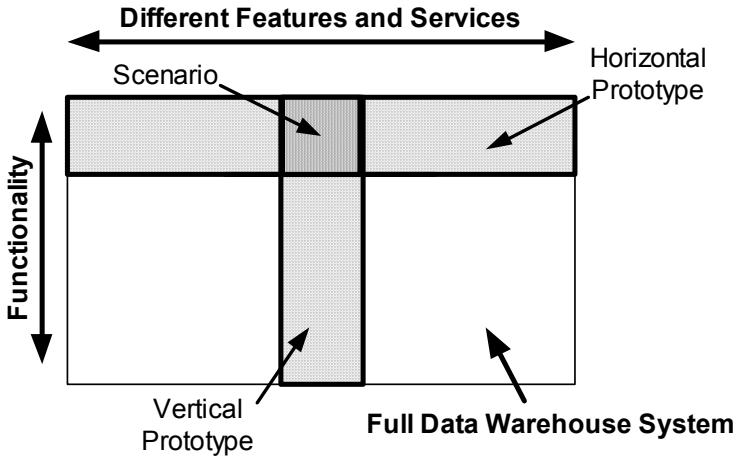

Figure 3: Horizontal Prototypes, Vertical Prototypes, and Scenarios

The simulation is often enough to give the users a feeling for the data warehouse system and lets them judge whether any functionality is missing, wrong, or unnecessary. The prototype represents the concept to the data warehouse team of how a specific requirement or use case might be implemented. The user's evaluation of the prototype can point out alternative courses for a use case, new missing process steps, previously undetected exception conditions, or new ways to visualize information.

Vertical Prototypes. Vertical prototypes are also known as structural prototypes or proof of concepts. They implement a slice of the data warehouse functionality. Vertical prototypes are developed, when uncertainty exists whether a proposed architectural approach is sound or when certain data warehouse policies should be optimized (i.e. access policies, security policies, data quality policies), or the data

warehouse schema should be evaluated (i.e. performance evaluation), or critical timing requirements (i.e. time for data warehouse queries) should be tested. Vertical prototypes are used more to reduce risks during the data warehouse design and implementation than for requirements development. They are generally constructed with the proposed tools of the data warehouse system (database, data staging tools, metadata management tools etc.) in the operating environment to make the results meaningful. Prototype tools, like the BEDAWA tool, can help to construct realistic vertical prototypes.

Scenarios. Scenarios can be seen as a combination of horizontal and vertical prototyping by reducing both the level of functionality and the number of features and services. Since scenarios are small and cheap, they can be applied and changed more frequently. Therefore, scenarios are a way of getting quick and frequent feedback from users. Scenarios can be implemented as paper mock-ups [11], [12] or in simple prototyping environments that may be easier to learn than more advanced programming environments. This is an additional savings compared to more complex prototypes requiring the use of advanced software tools.

4.2 Throwaway vs. Evolutionary Prototypes

Before a prototype is built, the decision has to be made, whether the prototype will be discarded after evaluation or will eventually evolve into a portion of the final data warehouse system. This decision must be explicit and well communicated.

Throwaway Prototypes. The throwaway prototype is most appropriate when you face uncertainty, ambiguity, incompleteness, or vagueness in the data warehouse requirements [2]. These issues have to be resolved to reduce the risk of proceeding with the data warehouse implementation. Because throwaway prototypes are discarded after they have served their purpose, they should be built as quickly and cheaply as possible. When throwaway prototypes are built, solid software construction techniques can be ignored. The emphasis is on a quick implementation and modification over robustness, reliability, performance, and long-term maintainability. For this reason, you must not allow the code from a throwaway prototype to migrate into a final data warehouse system. Development teams often tend to recycle existing throwaway prototypes. A good way to avoid this problem is to use different tools and a different programming language for the prototype construction.

Evolutionary Prototypes. Evolutionary prototypes provide a solid architectural foundation for building the data warehouse system incrementally as the requirements become clearly defined over time. In contrast to the throwaway prototyping, an evolutionary prototype must be built with robust, high-quality code from the beginning. Therefore, an evolutionary prototype takes longer to build than a throwaway prototype that addresses the same functionality. As evolutionary prototype must be designed for easy growth and frequent enhancement, it is important to

emphasize the system architecture and solid design principles. For data warehouse systems, evolutionary prototypes are particularly interesting for simulating data models. If a tested data model fulfills the requirements for robustness, extensibility, and performance, it can be used as the foundation for the final data warehouse implementation.

5. BEDAWA - Prototyping Tool

When building a prototype for a data warehouse system, one major problem has to be solved - the generation of data for the data warehouse. It can very difficult to use production data for this purpose, since the data model of the data warehouse system can be very different from the data model of production systems, and consequently, data transformations would be necessary. Once a data basis is established, any prototype can be built using this data.

The BEDAWA tool offers support for this problem. It allows developers of data warehouse prototypes to produce sample data suitable for their prototype implementation. By specifying the characteristics of the sample data and by testing the sample data generation results, they are able to repeatedly develop sample data for their needs. In this section we give an overview of the BEDAWA approach and show how to generate mass sample data for prototyping purposes.

5.1 Introduction of the BEDAWA Approach

The lack of sample data for data warehouse or OLAP systems usually makes it difficult for organizations to evaluate, demonstrate or build prototypes for their systems. However, the generation of representative sample data for data warehouses is a challenging and complex task. Difficulties often arise in producing familiar, complete and consistent sample data on any scale. Producing sample data manually often causes problems, which can be avoided by an automatic generation tool producing consistent and statistical plausible data.

The sample data generation process is usually a complex, iterative process that begins with a modeling step, and ends with the generation of the desired sample data. To get representative sample data that is familiar, consistent, scalable, large amount and that reflects various degrees of freedom, the BEDAWA tool is based on a statistical model that will be introduced in the next sections. The statistical correctness of the model provides a framework to define relationships between dimensions and facts in the context of a star schema, the most popular schema for designing and building a data warehouse [8]. The tool provides abilities to define and generate sample data of any size in order to reflect a real world situation. Furthermore, the tool is able to integrate existing external data sources for the sample data generation.

5.2 Statistical Model of BEDAWA

According to the statistical point of view, relationship between dimensions and a fact can be presented as follows:

$$\mathbf{y = f(\delta_1, \delta_2, ..., \delta_x) + \varepsilon} \qquad \textbf{(1)}$$

where:

y	: a value of the fact
$\delta_1, \delta_2, ..., \delta_x$	: numbers presenting the effect of dimension members to the fact y
ε	: error value

That means, based on relationship between facts and dimensions of a fact table, a statistical model can be applied to present that relationship stated in formula (1). Besides the calculation of fact values, the statistical model includes a distribution process, which takes care of the task to distribute dimension members in the fact table according a proper statistical distribution.

For example, a fact value can be created by using a linear model as follows [13]:

$$\mathbf{y_{ijk...} = \mu + c_i\alpha_i + c_j\beta_j + c_k\gamma_k + ... + \varepsilon_{ijk...}} \qquad \textbf{(3)}$$

where:

$\alpha, \beta, \gamma, ...$	:	fact effect value sets of dimensions D_α, D_β, D_γ that have effect on the fact y, the dimensions have numbers of elements in their domains are n, m, p, ... respectively,
$c_i, c_j, c_k, \ldots$	:	scalar constants.
i, j, k, ...	:	index numbers that are in range [1..n], [1..m], [1..p], ..., respectively,
μ	:	average mean of the fact,
$\alpha_i, \beta_j, \gamma_k, ...$	:	fact effect values of elements i^{th}, j^{th}, k^{th} of dimension fact effect value sets $\alpha, \beta, \gamma, ...$, respectively,
$\varepsilon_{ijk...}$	:	error value.
$y_{ijk...}$	:	a value of the fact.

5.3 Sample Data Generation Process

Figure 4 shows the sample data generation process of BEDAWA. The first two steps (grayed boxes) are considered as preparation steps. The major purpose of these steps is getting statistical parameters for the sample data. The next steps cover the modeling and the design of the sample data. In this stage all definitions for the sample data are completed which are required to generate the sample data automatically. The statistical parameters and data schemas of the data warehouse are used to model the

sample data architecture [5]. In the sample data generation step, the modeled sample data are automatically generated.

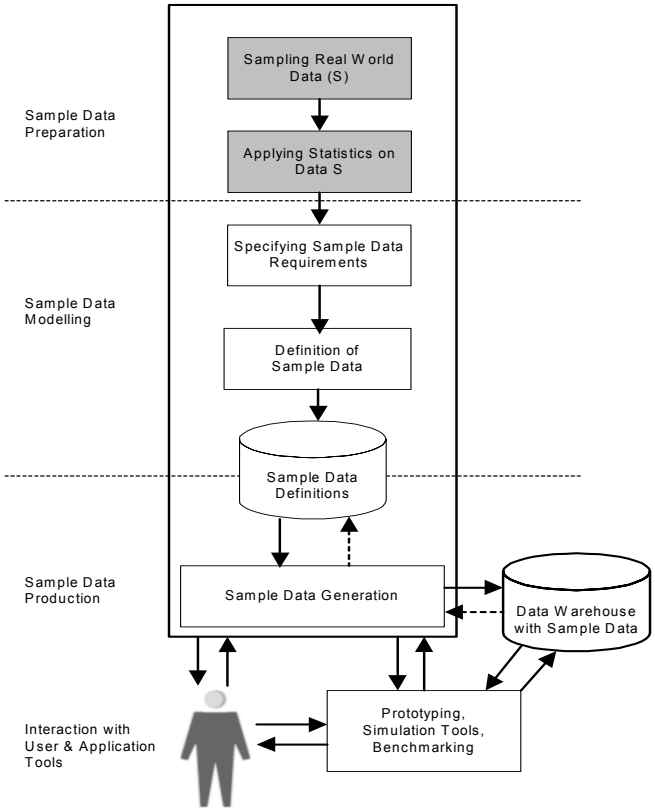

Figure 4: Process for Generating Sample Data for a Prototype

Users or applications are able to control the sample data generation. That means, depending on the result of a run of a sample data generation, possible iterations of the presented steps can be necessary. If for instance the result of the sample data production does not fulfill the requirements for a prototype, the user will have to go back to the sample data modeling stage to model the sample data correctly. The user iterates between these steps until the produced sample data is satisfying. A further explanation of the sample data generation process is beyond the scope of this paper. A detailed description can be found in [6].

5.4 Generating Sample Data for prototypes

Builders of horizontal prototypes for data warehouse systems need the complete data structures of the data warehouse with or without data. On the other hand, for vertical prototypes, only a part of data structures is needed, however with a realistic amount and quality of data. By using the BEDAWA tool, both types of prototypes can be built. The tool provides a rich set of functionality for modeling the data structures of a data warehouse system and for describing the quality of the desired sample data.

Following we show how to built a prototype with the star schema shown in **Figure 5**. This star schema is the well-known grocery sample introduced in [8].

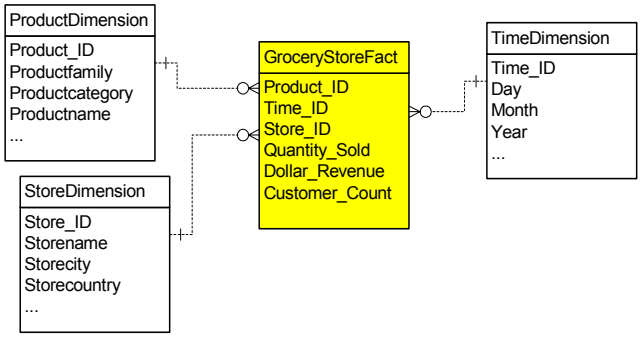

Figure 5: Star schema

The generation process of this star schema includes the following steps:

1) Definition of *data sources*. Data sources are used to identify any necessary data for the sample data generation of the prototype.
2) Definition of *dimensions*. Dimension definitions describe the dimension structures of the modelled sample data. For our example, we define the dimensions ProductDimension, StoreDimension and TimeDimension. The dimension data can be either automatically generated or manually copied from available data sources.
   Figure 6 shows the definition of the StoreDimension. A list of attributes is defined, which is described by name, data type, size, and number of data items. Furthermore, each dimension can be constructed based on various hierarchies, e.g., for TimenDimension dimension we can define two hierarchies as follows: day → month → year or dayOfWeek → Week → year.
3) Definition of *statistical parameters* for the dimensions. The statistical parameters include distributive information, and fact effect values. Fact effect values

quantify each relationship between a dimension and a fact. Distributive information is used to specify the distribution of dimension members in the fact tables for the sample data. The distribution can be statistical distribution or user-defined distribution parameters.

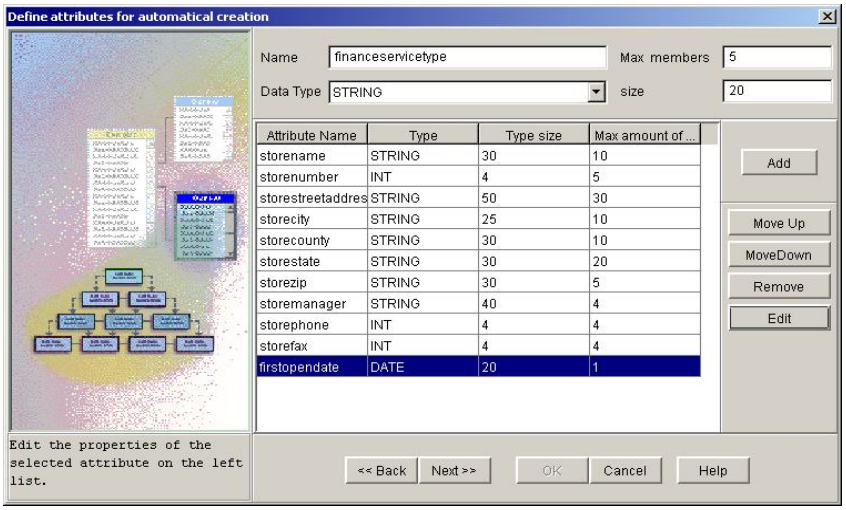

Figure 6: Defining store dimension

4) Definition of the *facts* for the fact table. Each fact is presented by a statistical model, which represents the relationship between fact values and members of dimension components effecting to the fact. For our example, we defined following three facts illustrated in Table 1.

Fact Name	Fact Definition	Error value
F(quatity_sold)	253 + FactEffect(time2_fe1) + FactEffect(product_fe1) + FactEffect(promotiondimension_fe1)	Random in [-10,10]
F(dollar_revenue)	1453 + FactEffect(time2_fe2) + FactEffect(product_fe2) + FactEffect(promotiondimension_fe2) + FactEffect(storedimension_fe1)	Normal distribution $\mu = 10$ $\sigma = 2$
F(customer_count)	42 + FactEffect(time2_fe3) + FactEffect(promotiondimension_fe3)	Random in [-5, 7]

Table 1: Fact Definitions

5) Defining the *fact table schema*. A fact table consists of various facts defined in the previously step and dimensions, which should be included in the fact table. E.g., the Grocery_Store fact table of our example includes the 3 facts quantity_sole, dollar_revenue, and customer_count and the 3 dimensions ProductDimension, StoreDimension, and TimeDimension.
6) *Generation of sample data*. In this step the user specifies the amount of sample data to be created and starts the generation process. Depending on the type of prototype (horizontal vs. vertical), the users can decide how large the fact table should get.

As in the previous steps shown the BEDAWA sample data generating process supports a flexible specification of the data warehouse metadata (dimension attributes, dimension hierarchies, fact, fact schema etc.) and business data. As result, such sample data is a sound basis for building prototypes for a proof-of-concept or proof-of-performance.

6. Conclusion

The expectation of organizations that more or better requirement and development tools would be a remedy to all problems of building data warehouse systems cannot be fulfilled. Case studies about prototyping in large industry projects show that the central problem to be solved is the difference between application knowledge and information processing knowledge [9]. This difference is clearly visible at the gap between the expectation of stakeholders towards the envisioned system and the resulting data warehouse system.

In this paper, we stressed the importance of building effective prototypes for closing this gap. We have shown how prototyping activities are part of the requirement process and how they facilitate iterative and incremental requirements development. We discussed different types of prototypes, by using the high-level characterization of Floyd. We introduced the BEDAWA tool as prototyping tool for generating sample data for data warehouse systems. The BEDAWA provides a statistical model for the specification of sample data and allows the generation of sample data of various amount and type for prototyping and benchmarking purposes.

References

[1] Bitton, D.; Orji, C., Program Documentation for DBGEN, a test database generator, University of Illinois at Chicago.
[2] Davis, Alan M., Software Requirements: Objects, Functions, and States, PTR Prentice Hall

[3] Floyd, C. A, A Systematic Look at Prototyping.
In: Bude et al. (eds) approaches to Prototyping Springer Verlag, 1984
[4] Goldman, N.M.; Narayanaswamy, K., Software Evolution through Iterative Prototyping 14th Int. Conf. on SW Eng., IEEE, 1992
[5] Huynh, T.; Nguyen, B.; Schiefer, J.; Tjoa, A M., BEDAWA - A Tool for Generating Sample Data for Data Warehouses. Data Warehousing and Knowledge Discovery, 2nd Int'l Conf., Dawak 2000, Greenwich, UK., Springer-Verlag.
[6] Huynh, T.; Nguyen, B.; Schiefer, J.; Tjoa, A M., Representative Sample Data for Data Warehouse Environments ADVIS2000, Turkey
[7] Informix Redbrick Warehouse, TPC-D and its relevance, Informix Corporation
[8] Kimball, R. (1996). The Data Warehouse Toolkit, John Wiley & Sons Inc.
[9] Lichter, H.; Schneider-Hufschmidt, M.; Zuellighoven, H. Prototyping in Industrial software Projects - Bridging the Gap Between Theory and Practice, IEEE Trans. on SE, Vol 20, No 11, pp 825-832, 1993
[10] List, B.; Schiefer, J.; Tjoa, A M., Use Case Driven Requirements Analysis for Data Warehouse Systems, Data Warehousing 2000, Friedrichshafen
[11] Nielsen, J., Prototyping user interfaces using an object-oriented hypertext programming system, Proc. NordDATA'89 Joint Scandinavian Computer Conference Copenhagen, Denmark
[12] Nielsen, J., Paper versus computer implementations as mockup scenarios for heuristic evaluation, Proc. INTERACT'90 3rd IFIP Conf. Human-Computer Interaction Cambridge, UK
[13] Rao, C. R.; Toutenburg, H., Linear Models least Squares and Alternatives, Springer-Verlag New York, Inc. 1995.
[14] Schach, S. R., Classical and Object-Oriented Software Engineering, 3rd, IRWIN.
[15] Schiefer, J.; Tjoa, A M., Generating Sample Data for Mining and Warehousing Proc. Int. Conference on Business Information System Springer-Verlag, BIS 99, Poznan
[16] Turbyfill, C.; Bitton, D., AS3AP-An ANSI SQL Standard Scalable and Portable Benchmark for Relational Database Systems. The Benchmark Handbook. J. Gray, Morgan Kaufmann Publishers.
[17] Whiteside, J.; Bennett, J.; Holtzblatt, K., Usability Engineering: Our experience and Evolution. In: Handbook of Human-Computer Interaction Amsterdam, Elsevier, 1988