MODELING THE PHYSICAL DESIGN OF DATA WAREHOUSES FROM A UML SPECIFICATION

Sergio Luján-Mora, Juan Trujillo
Department of Software and Computing Systems
University of Alicante
Alicante, Spain
email: {slujan,jtrujillo}@dlsi.ua.es

ABSTRACT
A Data Warehouse (DW) is a complex information system mainly used to support strategy decisions. During the last few years, several approaches have been proposed to model different aspects of a DW. However, few efforts have been dedicated to the modeling of the physical design (i.e. the physical structures that will host data together with their corresponding implementations) of a DW from the early stages of a DW project. In this paper, we present a proposal for the modeling of the physical design of DWs by using the *component diagrams* and *deployment diagrams* of UML. With these diagrams, we can anticipate important physical design decisions that may shorten the overall development time of a DW such as replicating dimension tables, vertical and horizontal partitioning of a fact table, the use of particular servers for certain ETL processes and so on. The approach presented in this paper complements our previous works for the conceptual and logical design of DWs, and therefore, to the best of our knowledge we provide the first global proposal, based on the Unified Modeling Language (UML), that allows us to cover all main design phases of DWs, from the conceptual modeling phase until the final implementation.

KEY WORDS
data warehouses, configuration, deployment, component, UML, physical design

1 Introduction

A Data Warehouse (DW) [1] is a complex information system mainly used for strategic decision-making by using OLAP (*On-Line Analytical Processing*) applications, *data mining* and/or *knowledge discovery* techniques. Although there have been several methods and approaches in the last years focused on the modeling of different parts of DWs (conceptual and logical modeling, modeling ETL processes, etc.); to the best of our knowledge, there is not any standard method or model that allows us to model all aspects of a DW.

So far, most of the research efforts in designing and modeling DWs has been focused on the development of Multidimensional (MD) data models [2], while the interest on the physical design of DWs has been very poor (see related work in Section 2). Nevertheless, an outstanding physical design is of a vital importance and highly influences the overall performance of the DW [3] and the ulterior maintenance.

In DWs, as in any other software project, once the conceptual and logical design has been accomplished, we have to deal with the physical design that implements the corresponding specification. Nevertheless, in DWs and mainly due to the large volumen of data that they manage, we normally face with a high number of implementation problems such as the storage of fact tables in different hard disks, copying the same table, vertical and horizontal partitioning and so on. Due to the idiosyncrasy of DWs, we can adopt several decisions regarding the physical design from the early stages of a DW project (in which final users, analysts, designers, and administrators participate). We believe that these decisions will normally shorten the total development time of the DW. Please, take into account that we are not saying to accomplish the conceptual modeling of a DW taking into consideration physical issues, instead we argue to mode physical aspects and ulterior implementations together with the conceptual modeling of the DW from the early stages of a DW project.

In previous works, we have dealt with the modeling of different aspects of a DW by using UML [4]: multidimensional modeling [5, 6], modeling of the ETL processes [7], etc. Furthermore, we have also proposed a method to properly design all aspects of a DW [8]. In this paper we present, within the context of our method to design DWs, a proposal to accomplish the physical design of DWs from early stages of a DW project. To accomplish this, we propose the use of the *component diagrams* and *deployment diagrams* of UML. Component diagrams show the physical storage of the database (tablespaces, partitions, etc.), whereas deployment diagrams show the hardware configuration that is used for the database and applications. Both *component* and *deployment* diagrams must be defined at the same time by DW designers and people who will be in charge of the ulterior implementation and maintenance. This is mainly due to the fact that, while the former know how to design and build a DW, the latter have a better knowledge in the corresponding implementation and the real hardware and software needs for the correct functioning of the DW.

The modeling of the physical design of a DW from the early stages of a DW project with our proposal provides us many advantages:

- We deal with important aspects of the implementation before we start with the implementation process itself, and therefore, we can shorten the total development time of the DW. This is mainly due to the fact that, after the conceptual modeling has been accomplished, we can have enough information to take some decisions regarding the implementation of the DW structures such as replicating dimension tables or making the vertical or horizontal partitioning of a fact table.
- We have a rapid feedback if we have a problem with the DW implementation as we can easily track a problem to find out its main reasons.
- It facilitates the communication between all people involved in the design of a DW since all of them use the same notation (based on the UML) for modeling different aspects of a DW. Moreover, making sure that the crucial concepts mean the same to all groups of people and is not used in different ways is critical.
- It help us choose both hardware and software on which we intend to implement the DW. This also allows us to compare and evaluate different configurations based on user requirements.
- It allows us to verify that all different parts of the DW (fact and dimension tables, ETL processes, OLAP tools, etc.) perfectly fit together.

The rest of the paper is organized as follows. In Section 2, we briefly comment other works that have dealt with the physical design and/or deployment of a DW. In Section 3, we briefly introduce our overall method to design all aspects of a DW. In Section 4, we describe our proposal for using both component and deployment diagrams for the physical design of DWs. Finally, in Section 5, we present our conclusions and main future works.

2 Related Work

So far, both the research community and companies have devoted few effort to the physical design of DWs from the early stages of a DW project, and incorporate it within a global method that allows us to design all main aspects of DWs.

In [9], authors deal the lifecycle of a DW and propose a method for the design, development and deployment of a DW. In this book, we can find a chapter devoted to the planning of the deployment of a DW and authors recommend us documenting all different deployment strategies. However, authors do not provide a standard technique for the formal modeling of the deployment of a DW.

In [10], authors deal with the design of a DW from the conceptual modeling up to its implementation. They propose the use of non-standard diagrams to represent the physical architecture of a DW: on one hand, to represent data integration processes and, on the other hand, to represent the relationship between the *enterprise data warehouse* and the different *data marts* that are populated from it. Nevertheless, these diagrams represent the architecture of the DW from a high level, without providing different levels of detail of the ulterior implementation of the DW.

In [11], several aspects of a DW implementation are discussed. Although in this book, other aspects of a DW implementation such as the paralelism, the partitioning of data in a RAID (*Redundant Array of Inexpensive Disk*) system or the use of a distributed database are tackled, authors do not provide a formal or standard technique to model all these aspects.

Finally, in [12], we find that one of the current open problems regarding DWs is the lack of a formal documentation that covers all design phases and provides multiple levels of abstraction (low level for designers and people devoted to the corresponding implementation, and high level for final users). The author argues that this documentation is absolutely basic for the maintenance and the ulterior extension of the DW. In this work, three different detail levels for DWs are proposed: *data warehouse level*, *data mart level* and *fact level*. At the first level, the use of the deployment diagrams of UML are proposed to document a DW architecture from a high level of detail. However, these diagrams are not integrated at all with the rest of techniques, models and/or methods used in the design of other aspects of the DW.

We argue that there is a still a need for providing a standard technique that allows us to model the physical design of a DW from early stages of a DW project. Therefore, in this paper we present a proposal for the modeling of the physical design of DWs by means of UML. Our proposal is totally integrated in an overall approach that allows us to cover other aspects of the DW design such the conceptual or logical design of the DW or the modeling of ETL processes.

3 Data Warehouse Design Framework

The architecture of a DW is usually depicted as various layers of data in which data from one layer is derived from data of the previous layer [13]. Following this consideration, we consider that the development of a DW can be structured into an integrated framework with five stages and three levels that define different diagrams for the DW model, as summarized in Table 1.

In previous works, we have presented some of the diagrams and the corresponding profiles for the different stages and levels presented in Table 1: *Multidimensional Profile* [5, 6] for the *Client Conceptual Schema* (CCS), the *ETL Profile* [7] for the *ETL Process* and the *Exporting Process*, and the *Data Mapping Profile* [14] for the *Data Mapping* (DM) between the *Source Conceptual Schema* (SCS) and the *Data Warehouse Conceptual Schema* (DWCS), and

- **Stages**: we distinguish five stages in the definition of a DW:
  - Source, that defines the data sources of the DW, such as OLTP systems, external data sources (syndicated data, census data), etc.
  - Integration, that defines the mapping between the data sources and the DW.
  - Data Warehouse, that defines the structure of the DW.
  - Customization, that defines the mapping between the DW and the clients' structures.
  - Client, that defines special structures that are used by the clients to access the DW, such as data marts (DM) or OLAP applications.
- **Levels**: each stage can be analyzed at three different levels or perspectives:
  - Conceptual: it defines the DW from a conceptual point of view.
  - Logical: it addresses logical aspects of the DW design, such as the definition of the ETL processes.
  - Physical: it defines physical aspects of the DW, such as the storage of the logical structures in different disks, or the configuration of the database servers that support the DW.
- **Diagrams**: each stage or level requires different modeling formalisms. Therefore, our approach is composed of 15 diagrams, but the DW designer does not need to define all the diagrams in each DW project: for example, if there is a straightforward mapping between the *Source Conceptual Schema* (SCS) and the *Data Warehouse Conceptual Schema* (DWCS), the designer may not need to define the corresponding *Data Mapping* (DM). In our approach, we use the UML [4] as the modeling language, because it provides enough expressiveness power to address all the diagrams. As the UML is a general modeling language, we can use the UML extension mechanisms (stereotypes, tag definitions, and constraints) to adapt the UML to specific domains.

Table 1. Data warehouse design framework: summary

between the DWCS and the CCS. Finally, in this paper we present the *Database Deployment Profile*, for modeling a DW at a physical level.

The different diagrams of the same DW are not independent but overlapping: they depend on each other in many ways. For example, changes in one diagram may imply changes in another, and a large portion of one diagram may be created on the basis of another diagram. For example, the DM is created by importing elements from the SCS and the DWCS.

4 Data Warehouse Physical Design

In Section 3, we have briefly described our design method of DWs. Within this method, we use the component and deployment diagrams to model the physical level of DWs. To achieve this goal, we propose the following five diagrams, which correspond with the five stages presented in Table 1:

- *Source Physical Schema* (SPS): it defines the physical configuration of the data sources that populate the DW.
- *Integration Transportation Diagram* (ITD): it defines the physical structure of the ETL processes that transform and load data in the DW. This diagram relates the SPS and the next diagram.
- *Data Warehouse Physical Schema* (DWPS): it defines the physical structure of the DW itself.
- *Customization Transportation Diagram* (CTD): it defines the the physical structure of the exportation processes from the DW to the specific structures employed by clients. This diagram relates the DWPS and the next diagram.
- *Client Physical Schema* (CPS): it defines the physical configuration of the structures employed by clients in accessing the DW.

The SPS, DWPS, and CPS are based on the UML component and deployment diagrams, whereas ITD and CTD are based on the deployment diagrams.

The five proposed diagrams use an extension of UML that we have called *Database Deployment Profile*, which is formed by a series of stereotypes, tagged values and constraints. Due to the lack of space, we do not include in this paper the formal definition of this extension.

Throughout the rest of this paper, we are going to use an example to introduce the different diagrams we propose. In this example, the final users need a DW that contains the daily sales of a company. There exist two data sources: the sales server, which contains the data about transactions and sales, and the CRM (*Customer Relationship Management*) server, which contains the data about the customers who buy products. In Figure 1, we show the *Data Warehouse Logical Schema* (DWLS), which represents the logical model of the DW. In this example, a ROLAP (*Relational OLAP*) system has been selected for the implementation of the DW, which means the use of the relational model in the logical design of the DW. In Figure 1, five classes adorned with the stereotype ≪Table≫ are showed: Customers, Periods, Products, and Stores are represented by means of the icon of the stereotype, whereas the table Sales appears with the icon of the stereotype inside

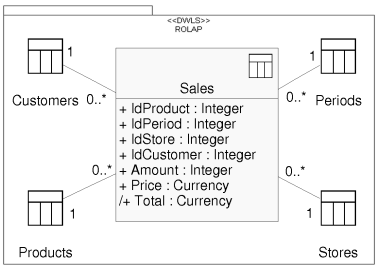

Figure 1. Logical model (ROLAP) of the data warehouse

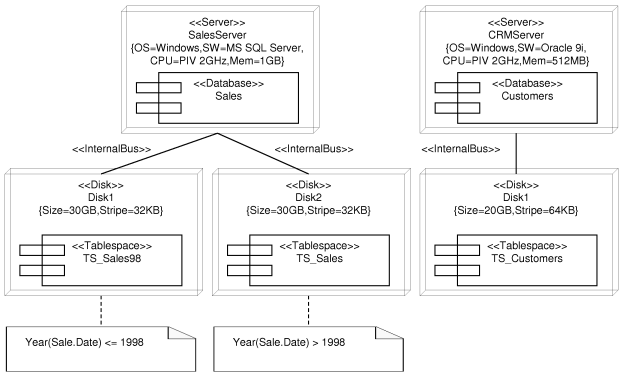

Figure 2. Source Physical Schema: deployment diagram

the typical representation of a class in UML. In order to avoid a cluttered diagram, we only display the attributes of Sales: the attributes IdProduct, IdPeriod, IdStore and IdCustomer are the foreign keys that connect the fact table with the dimension tables, whereas the attributes Amount, Price, and Total (derived attribute) represent the measures of the fact table.

4.1 Source Physical Schema

The SPS describes the origins of data of the DW from a physical point of view. In Figure 2, we show the SPS of our example, which is formed by two servers called SalesServer and CRMServer; for each one of them, the hardware and software configuration is displayed. The first server hosts a database called Sales, whereas the second server hosts a database Customers.

In our *Database Deployment Profile*, when the storage system is a RDBMS (Relational Database Management System), we make use of the *UML for Profile Database* [15] that defines a series of stereotypes like ≪Database≫ or ≪Tablespace≫. Moreover, we have defined our own set of stereotypes: in Figure 2, we can see the stereotypes ≪Server≫ that defines a computer that performs server functions, ≪Disk≫ to represent a physical disk drive and ≪InternalBus≫ to define the type of communication between two elements. Whenever we need to specify additional information in a diagram, we make use of the UML notes to incorporate it. For example, in Figure 2 we have used two notes to indicate how the data is distributed into the two existing tablespaces.

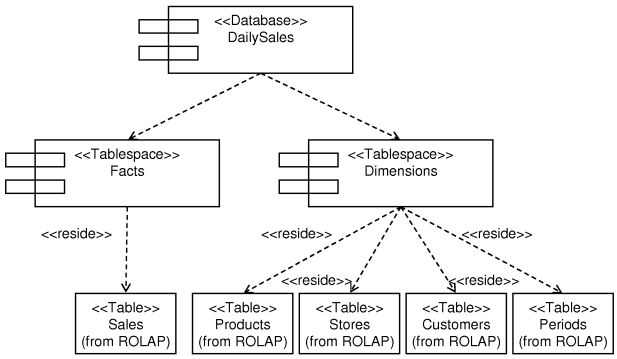

Figure 3. Data Warehouse Physical Schema: component diagram

4.2 Data Warehouse Physical Schema

The DWPS shows the physical aspects of the implementation of the DW. This diagram is divided up into two parts: the component and deployment diagrams. In the first diagram, the configuration of the logical structures used to store the DW is shown. For example, in Figure 3, we can observe that the DW DailySales is formed by two tablespaces called Facts and Dimensions: the first tablespace hosts the table Sales and the second tablespace hosts the tables Products, Stores, Customers, and Periods. Below the name of each table, the text (from ROLAP) is included, which indicates that the tables have been previously defined in a package called ROLAP (Figure 1).

In the second diagram, the deployment diagram, different aspects relative to the hardware and software configuration are specified. Moreover, the physical distribution of the logical structures previously defined in the component diagrams is also represented. For example, in Figure 4, we can observe the configuration of the server that hosts the DW.

4.3 Integration Transportation Diagram

The ITD defines the physical structure of the ETL processes used in the loading of data in the DW from the data sources. On the one hand, the data sources are represented by means of the SPS and, on the other hand, the DW is represented by means of the DWPS. Since the SPS and the DWPS have been defined previously, in this diagram we do not have to define them again, but they are imported.

For example, the ITD for our running example is shown in Figure 5. On the left hand side of this diagram, different data source servers are represented: SalesServer

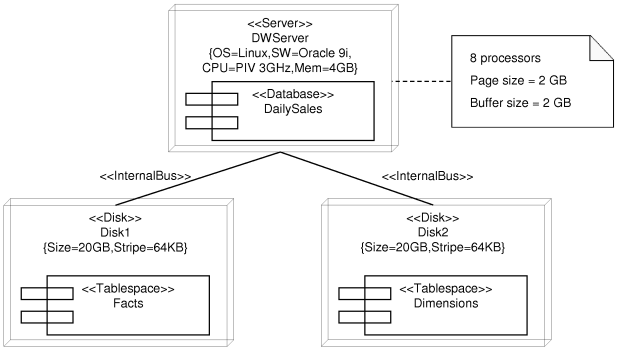

Figure 4. Data Warehouse Physical Schema: deployment diagram

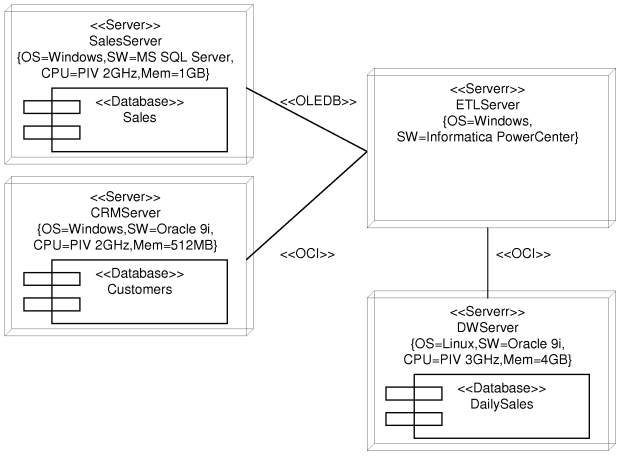

Figure 5. Integration Transportation Diagram: deployment diagram

and CRMServer, which have been previously defined in Figure 2; on the right hand side, the DWServer, previously defined in Figure 4, is shown. In this figure, the ETLServer is introduced, an additional server that is used to execute the ETL processes. This server communicates with the rest of the servers by means of a series of specific protocols: OLEDB to communicate with SalesServer because it uses Microsoft SQLServer and OCI (*Oracle Call Interface*) to communicate with CRMServer and DWServer because both of them use Oracle. The configuration of a server is defined by means of tagged values: OS, SW, CPU, etc.

4.4 Client Physical Schema

The CPS defines the physical structure of the specific structures that are used by the clients to access the DW. Diverse configurations exist that can be used: exportation of data to *data marts*, use of an OLAP server, etc. In our example, we

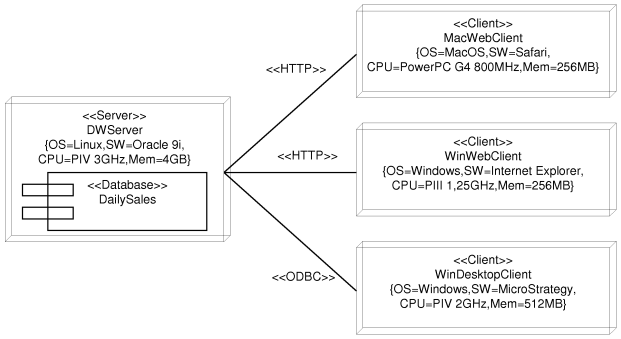

Figure 6. Customization Transportation Diagram: deployment diagram

have chosen a client/server architecture and the same DW server provides access to data for the clients. Therefore, we do not need to define a specific structure for the clients.

4.5 Customization Transportation Diagram

The CTD defines the exportation processes from the DW towards the specific structures used by the clients. In this diagram, the DW is represented by means of the DWPS and clients are represented by means of the CPS. Since the DWPS and the CPS have been previously defined, in this diagram we do not have to define them again, but they are directly imported.

For example, in Figure 6, the CTD of our running example is shown. On the left hand side of this diagram, part of the DWPS, which has been previously defined in Figure 4, is shown; on the right hand side, three types of clients who will use the DW are shown: a Web client with operating system Apple Macintosh, a Web client with operating system Microsoft Windows and, finally, a client with a specific desktop application (MicroStrategy) with operating system Microsoft Windows. Whereas both Web clients communicate with the server by means of HTTP (*HyperText Transfer Protocol*), the desktop client uses ODBC (they *Open Connectivity Database*).

5 Conclusions and Future Work

In this paper, we have presented an adaptation of the component and deployment diagrams of UML for the modeling of the physical design of a DW. One of the advantages of this proposal is that these diagrams are not used in an isolated way, instead they are used together with other diagrams that we use for the modeling of other aspects of a DW (conceptual and logical design, modeling of ETL processes, etc.) in the context of our overall method for designing DWs.

Thanks to the use of the component and deployment diagrams, a DW designer can specify both hardware and

software, and middelware needs for a DW project. The main advantages provided by our approach are as follows:

- It is part of an integrated approach in which we use different diagrams -always following the same standard notation based on UML, for modeling all main aspects of a DW.
- Traceability of the design of a DW, from the conceptual model up to the physical model.
- Reducing the overall development cost (time and money) as we accomplish implementation issues from the early stages of a DW project, becasue modifying these aspects in ulterior design phases may result in increasing the total cost of the project.
- Different levels of abstraction by providing different levels of details for the same diagram.

Regarding future works, we are currently working in the development of a global method, based on the Unified Process [16], to design all main aspects of DW (conceptual and logical design of the DW itself, modeling of ETL processes, atc.), including the deployment and component diagrams presented in this paper.

Acknowledgements

This work has been partially supported by the HIDRA project (TIC2001-3530-C02-02) from the Spanish Ministry of Education and Science.

References

[1] W.H. Inmon. *Building the Data Warehouse*. QED Press/John Wiley, 1992. (Last edition: 3ª, John Wiley & Sons, 2002).

[2] A. Abelló, J. Samos, and F. Saltor. A Framework for the Classification and Description of Multidimensional Data Models. In *Proc. of the 12th Intl. Conf. on Database and Expert Systems Applications (DEXA'01)*, pages 668–677, Munich, Germany, September 2001.

[3] M. Nicola and H. Rizvi. Storage Layout and I/O Performance in Data Warehouses. In *Proc. of the 5th Intl. Workshop on Design and Management of Data Warehouses (DMDW'03)*, pages 7.1–7.9, Berlin, Germany, September 2003.

[4] Object Management Group (OMG). Unified Modeling Language Specification 1.5. Internet: http://www.omg.org/cgi-bin/doc?formal/03-03-01, March 2003.

[5] S. Luján-Mora, J. Trujillo, and I. Song. Extending UML for Multidimensional Modeling. In *Proc. of the 5th Intl. Conf. on the Unified Modeling Language (UML'02)*, volume 2460 of *Lecture Notes in Computer Science*, pages 290–304, Dresden, Germany, September 2002.

[6] S. Luján-Mora, J. Trujillo, and I. Song. Multidimensional Modeling with UML Package Diagrams. In *Proc. of the 21st Intl. Conf. on Conceptual Modeling (ER'02)*, volume 2503 of *Lecture Notes in Computer Science*, pages 199–213, Tampere, Finland, October 2002.

[7] J. Trujillo and S. Luján-Mora. A UML Based Approach for Modeling ETL Processes in Data Warehouses. In *Proc. of the 22nd Intl. Conf. on Conceptual Modeling (ER'03)*, volume 2813 of *Lecture Notes in Computer Science*, pages 307–320, Chicago, USA, October 2003.

[8] S. Luján-Mora and J. Trujillo. A Comprehensive Method for Data Warehouse Design. In *Proc. of the 5th Intl. Workshop on Design and Management of Data Warehouses (DMDW'03)*, pages 1.1–1.14, Berlin, Germany, September 2003.

[9] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite. *The Data Warehouse Lifecycle Toolkit*. John Wiley & Sons, 1998.

[10] V. Poe, P. Klauer, and S. Brobst. *Building a Data Warehouse for Decision Support*. Prentice-Hall, 2 edition, 1998.

[11] W. Giovinazzo. *Object-Oriented Data Warehouse Design. Building a star schema*. Prentice-Hall, 2000.

[12] S. Rizzi. Open problems in data warehousing: eight years later. In *Proc. of the 5th Intl. Workshop on Design and Management of Data Warehouses (DMDW'03)*, Berlin, Germany, September 2003.

[13] M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis. *Fundamentals of Data Warehouses*. Springer-Verlag, 2 edition, 2003.

[14] S. Luján-Mora, P. Vassiliadis, and J. Trujillo. Data Mapping Diagrams for Data Warehouse Design with UML. In *Proc. of the 23rd Intl. Conf. on Conceptual Modeling (ER'04)*, Lecture Notes in Computer Science, Shanghai, China, November 2004.

[15] E.J. Naiburg and R.A. Maksimchuk. *UML for Database Design*. Object Technology Series. Addison-Wesley, 2001.

[16] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Object Technology Series. Addison-Wesley, 1999.