Executive Information Systems, Inc.

Dimensional Object Modeling

By

Joseph M. Firestone, Ph.D.

White Paper No. Seven

April 30, 1998

Introduction: Dimensional Object Modeling and Dimensional Data Modeling

An object modeling approach offers advantages in supporting Dimensional Data Modeling (DDM) of data warehouses and data marts. The current approach to making the basic decisions in producing a DDM is a pragmatic one. The pragmatic approach has had considerable commercial success [1], but it still

- makes tight coupling of strategic goals and objectives to the DDM result a matter of art, rather than a product of an explicit method or procedure,
- results in a model composed of passive containers for data attributes, rather than components that combine both data and behavior,
- does not place DDM within a broader framework for integrating data and process -- that is, the pragmatic approach is too data-centric, at a time when data warehousing is concerned with integrating a complex diversity of server-based decision support system functions.

While following an object modeling approach will not remove the element of art from the process, the traceability of requirements from strategic goals and objectives, through business processes, through sub-processes, through use cases, and through entity objects to the key decisions in constructing a DDM, is likely to greatly improve the validity of practice in the DDM area, in the sense of ensuring that DDMs reflect organizational goals and objectives. Also, an object modeling approach to DDM will by its very nature result in a collection of components combining both data and behavior. Finally, an object modeling approach can be more easily integrated within a distributed objects/components framework of enterprise-wide decision support. Such frameworks, represented by CORBA and DCOM are increasingly seen as the primary trend in organizational IT development, because they allow continued use of legacy systems and incremental progress toward solution of the islands of information problem.

Let’s now:

- examine the nature of DDM and DOM,
- develop the argument for tight coupling of strategic goals and objectives to the DDM through an object modeling approach, and
- discuss the advantages of the DOM approach in more detail.

Dimensional Data Modeling

DDM is the favorite modeling technique in data warehousing. In DDM, a model of tables and relations is constituted with the purpose of optimizing decision support query performance in relational databases, relative to a measurement or set of measurements of the outcome(s) of the business process being modeled. In contrast, conventional E-R models are constituted to (a) remove redundancy in the data model, (b) facilitate retrieval of individual records having certain critical identifiers, and (c) therefore, optimize On-line Transaction Processing (OLTP) performance.

Practitioners of DDM have approached developing a logical data model by selecting the business process to be modeled and then deciding what each individual low level record in the "fact table" (the grain of the fact table) will mean. The fact table is the focus of dimensional analysis. It is the table dimensional queries segment in the process of producing solution sets. The criteria for segmentation are contained in one or more "dimension tables" whose single part primary keys become foreign keys of the related fact table in DDM designs. The foreign keys in a related fact table constitute a multi-part primary key for that fact table, which, in turn, expresses a many-to-many relationship. [2]

In a DDM further, the grain of the fact table is usually a quantitative measurement of the outcome of the business process being analyzed. While the dimension tables are generally composed of attributes measured on some discrete category scale that describe, qualify, locate, or constrain the fact table quantitative measurements.

The decision about the grain of the fact table is crucial to DDM, because once it is selected, the dimensions that will relate to it can usually be easily specified, their attributes identified, and their respective grains determined. So it is difficult to overestimate the importance of selecting the fact table grain, or of making this selection on the basis of a procedure providing for a tight coupling between grain selection and user requirements.

This tight coupling can be achieved by selecting a fact table grain and associated numerical measurement attribute(s) that will facilitate planning, monitoring and evaluation of the extent to which past, present, planned, and forecast business outcomes will fail, meet, or exceed strategic and tactical goals and objectives. That is, the choice of fact table grain should flow from strategic and tactical goals and objectives and the need to measure actual outcomes and business performance against them.

Dimensional Object Modeling

Before describing dimensional object modeling, it will be useful to provide a brief framework of concepts from Object-Oriented Software Engineering (OOSE). These include: business process, sub-process, external user, business system actor, business system use case, information system actor, information system use case, class, object, relation, object type, object modeling, and component.

A Business Process is a sequence of interrelated activities that transforms inputs into positively or negatively valued outputs. Processes are value streams in that they are oriented toward producing value for the enterprise.

A business process is directed by organizational goals and objectives. It is driven by a variety of sub-processes, use cases, and tasks, whose collective purpose is to achieve goals and objectives. Some horizontal processes will involve similar or even the same task sequences across organizational and business types. Others will be unique to an organizational or business type. For example, Selling, Marketing, Financing & Investing, Accounting, Knowledge Management, Customer Care, Supervising, and Recruiting are examples of processes sharing common elements across industries. Agent Servicing, Agency Servicing, and New Business Underwriting, on the other hand are business processes specific to the insurance industry.

The sub-processes of any business process are: Planning, Acting, Monitoring, and Evaluating. Planning means setting goals, objectives, and priorities, making forecasts as part of prospective analysis, performing cost/benefit assessments as part of prospective analysis, and revising or reengineering a business process. Acting means performing the business process or any of its components. Monitoring means retrospectively tracking and describing the business process. Evaluating means retrospectively assessing the performance of the business process as a value stream.

Business Processes are used by actors who drive processes and sub-processes by performing use cases and tasks. **An External User** of a business system, is an individual or organization outside the logical boundary of the business area being modeled, who uses a business process or system. For example, a customer is a user external to the General Motors business system. In simply naming the user we imply no particular role or set of actions in connection with the business area. The user is not an abstraction except in the general sense that any concept is an abstraction.

A Business System Actor is a particular coherent cluster of activities adopted by a User in relation to a Business System or Process. These structured sets of activities, or roles played by users, are what we mean by Business System Actors. The actor concept is an abstraction from the basic notion of user. And note that unlike the user concept, it refers to a type of internal entity, a business system role.

A Business System Use Case is defined by Jacobson [3] as "A sequence of transactions in a system whose task is to yield a result of measurable value to an individual actor of the business system." A use case may also be composed of multiple transaction sequences or tasks. A behaviorally-related set of business use cases, in turn, constitutes a business process, and therefore extends over the four sub-processes. Figure One shows the relationships of business processes, sub-processes, use cases, and tasks (transaction sequences) to one another.

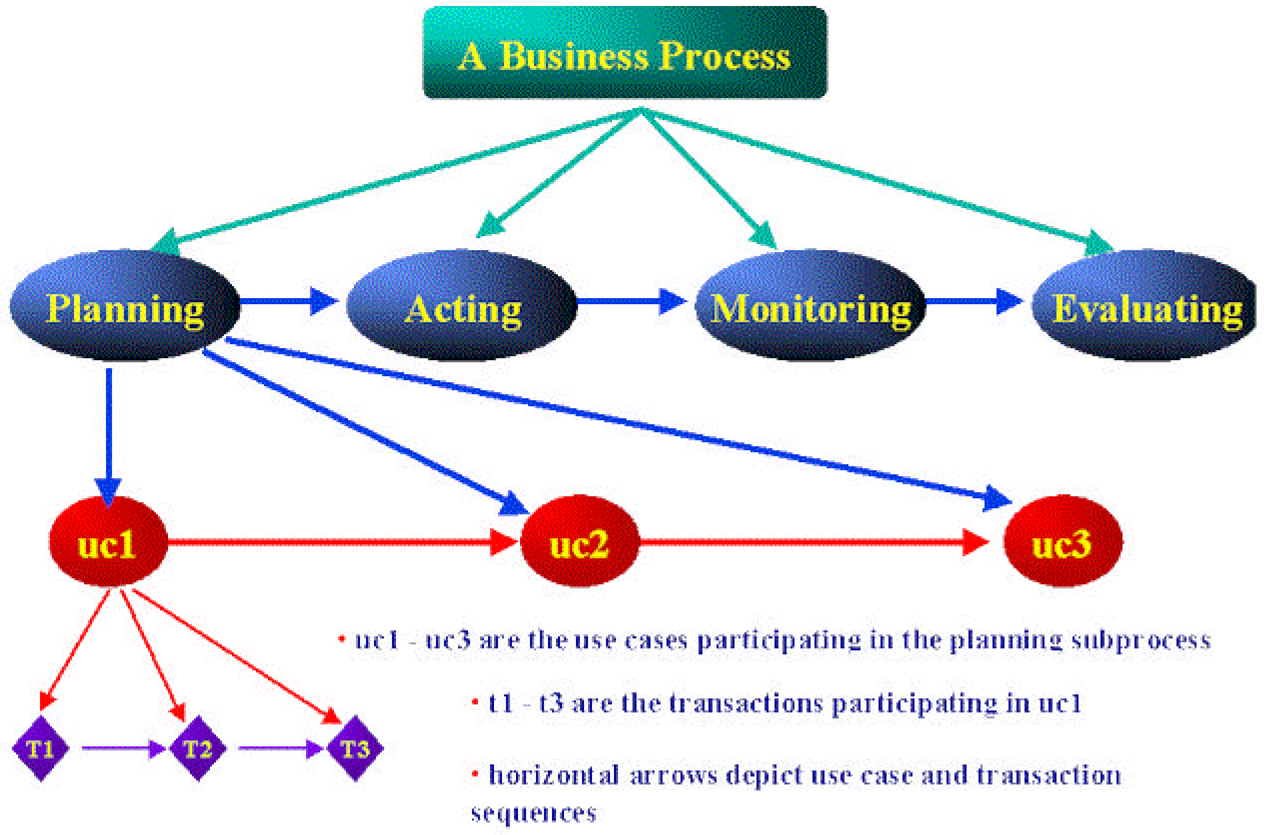
A Business Process
Planning
Acting
Monitoring
Evaluating
uc1
uc2
uc3
T1
T2
T3
• uc1 - uc3 are the use cases participating in the planning subprocess
• t1 - t3 are the transactions participating in uc1
• horizontal arrows depict use case and transaction sequences

Figure One -- The Hierarchy of Business Processes, Subprocesses, Use Cases And Transaction Sequences

A business process may be supported by a sub-process or subsystem called an information system. Actors who are external to the logical boundary of the information system and who play coherent roles in the business system are potential **Information System Actors.**

An Information System Use Case, in turn, is defined by Jacobson [4] as "A behaviourally related sequence of transactions performed by an actor in a dialogue with the system to provide some measurable value to the actor." A behaviorally related set of information system use cases, in turn, constitutes an information systems application supporting a business process through its use cases. This application may or may not extend over the four sub-processes, depending on its scope. Figure Two shows the relationships of business and information system use cases in an application.

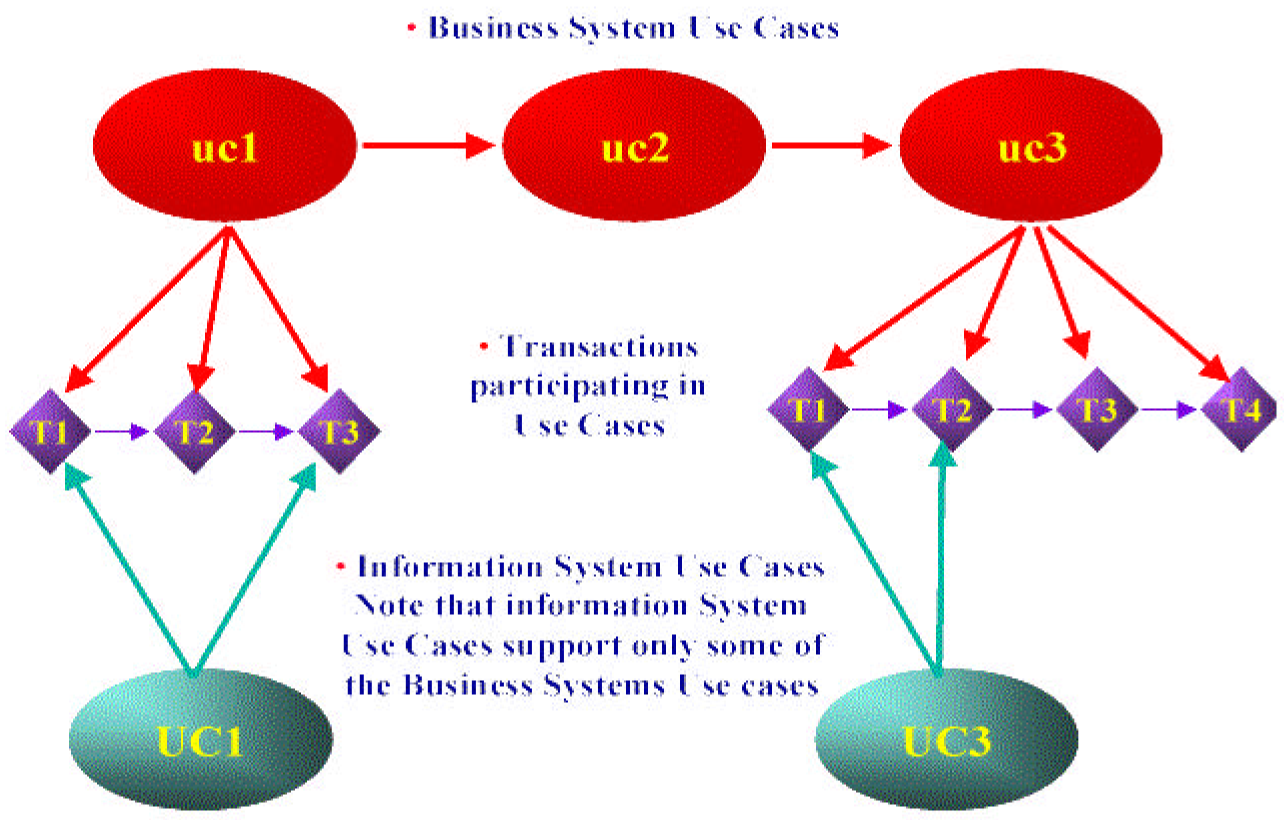
• Business System Use Cases
uc1
uc2
uc3
• Transactions participating in Use Cases
T1
T2
T3
T1
T2
T3
T4
• Information System Use Cases Note that information System Use Cases support only some of the Business Systems Use cases
UC1
UC3

Figure Two -- Relationships of Business System Use Cases to Information System Use cases

A use case is intended to accomplish some tactical objective of an actor, or to aid in accomplishing a tactical objective. The use case concept focuses attention on the user's viewpoint about what the system is supposed to give back in response to the user's input. That is, it is supposed to give back a response or output, and that output will have value relative to a hierarchy of tactical and strategic objectives and goals. Figure Three illustrates the connection between a use case and a hierarchy of goals and objectives.

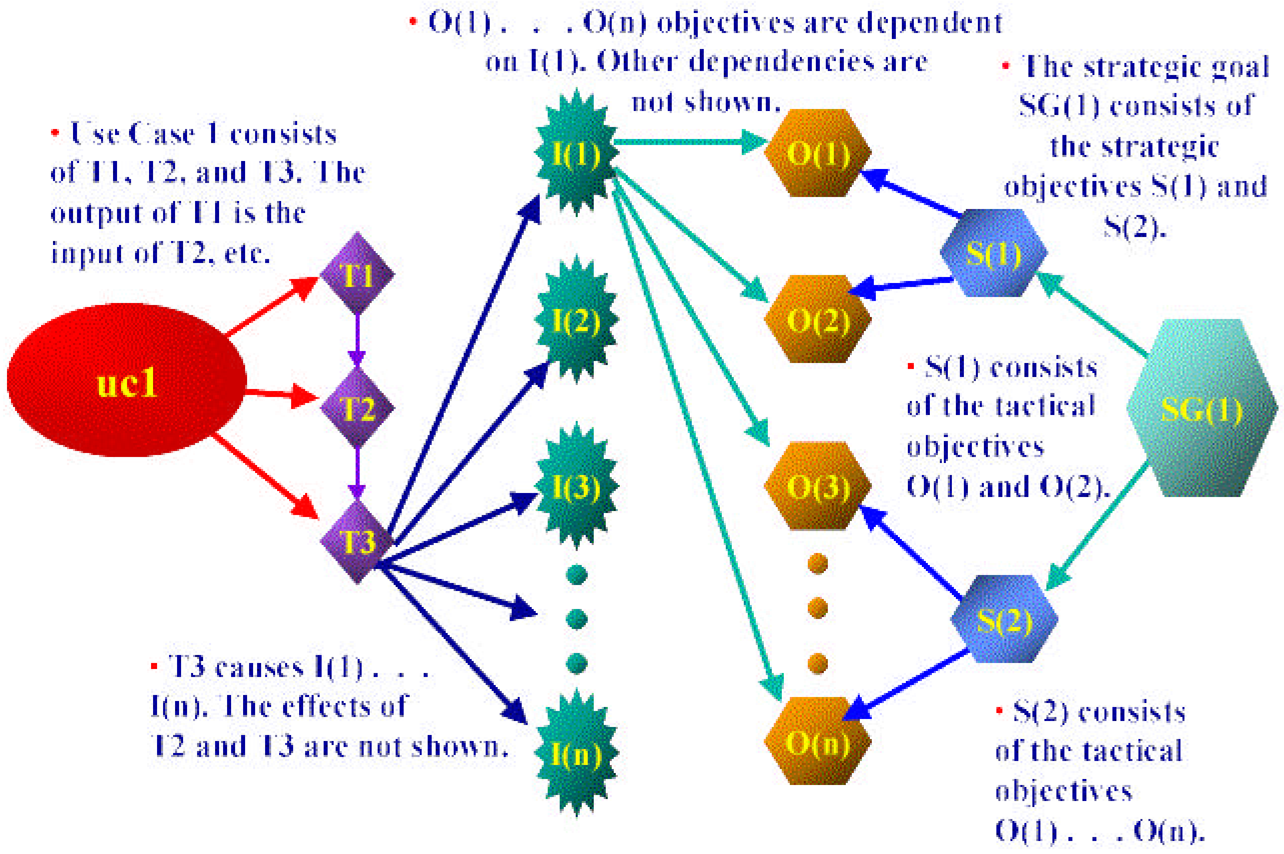
• O(1) . . . O(n) objectives are dependent on I(1). Other dependencies are not shown.
• The strategic goal SG(1) consists of the strategic objectives S(1) and S(2).
• Use Case 1 consists of T1, T2, and T3. The output of T1 is the input of T2, etc.
uc1
T1
T2
T3
I(1)
I(2)
I(3)
I(n)
O(1)
O(2)
O(3)
O(n)
S(1)
S(2)
SG(1)
• S(1) consists of the tactical objectives O(1) and O(2).
• T3 causes I(1) . . . I(n). The effects of T2 and T3 are not shown.
• S(2) consists of the tactical objectives O(1) . . . O(n).

Figure Three -- Connecting a Use Case to Goals and Objectives

Thus, the use case provides a view of the system from the viewpoint of its users, and their business purposes. If we build the system on a foundation of use cases specified by users, and let these lead us to objects and interactions, and given that objects defined at higher levels are traceable down to the level of implemented code, it follows that the whole system architecture will be determined by the use cases and ultimately by the users who specified them. If we want to change the system: to incrementally implement or improve it, to redesign it, or to re-engineer it; we can determine new requirements by asking users which use cases they want to see changed, deleted, or added, and we can determine the schedule of iterative development by asking the users about their priorities among the various use cases they have identified.

An **object** is a uniquely identifiable unit of analysis of which properties may be predicated. The properties of an object are its attributes, its behavior or operations, and its methods (the internal activity of an object implementing its behavior). A relation between two objects is a property of the object pair, and is itself an object. Examples of relations are aggregation, association, or inheritance.

An **object type** is a description of a set of objects (instances) sharing the same attributes, operations, and relationships. Object types have sub-types. Also, **classes** are implementations of types in software. So, objects are instances of classes as well as types.

The objects mentioned above, therefore, when generalized, also are examples of object types, and, when implemented in software, of classes. Purchase order is another example of an object type, as is document. Automobile Purchase Order is a subtype of Purchase Order.

A well-known abstract typology of objects is that of interface, control, and entity objects [5]. **Interface objects** are those that manage inputs to, and outputs from, an information system. **Control Objects** coordinate object interactions in one or more use cases. **Entity Objects** manage information, resources, and access to them within an information system. All of the examples of object types just given are entity object subtypes. Figure Four provides an illustration of an entity object.

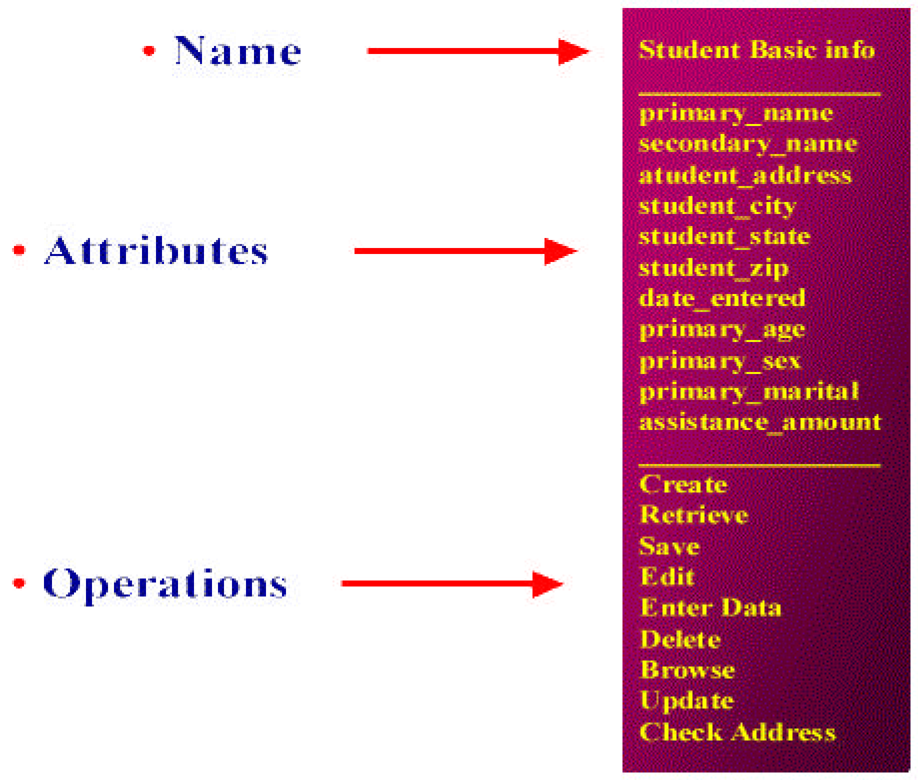

Figure Four -- An Entity Object with Name, Attributes, and Operations

More generally, object types encapsulate data and therefore have attributes. Objects are instances of object types whose attributes have values. Objects both perform and have operations performed on them e.g., products please peoples' taste buds, products are shipped. Objects also encapsulate methods that perform their operations, and these may be specified in code.

Another special type of object is a component. A **component** is a large grained object. It may be composed of clusters of tightly-related small- or smaller-grained objects. Or it may be a procedural legacy application that has been "wrapped," in a package providing an object interface and encapsulation to the legacy application.

Components are now enjoying great popularity as the foundation of true rapid application development, and distributed data warehousing is often characterized as a component-based business application. Sometimes analysts are at pains to distinguish components from objects, because objects were associated with earlier difficult and slow implementations of O-O systems, while components appear to have a very promising future in providing software reuse. In any case, it is clear that components are just large-grained objects, and concepts introduced here to describe objects and their interactions apply to components, as well.

According to the approach just outlined, business processes produce value streams, and sub-processes, and use cases contribute to value and to strategic and tactical goals and objectives within the analytic hierarchy. Also, use cases are performed by entity, control, and interface objects, and object modeling formulates relationships between such objects in the Object Model.

Focusing for present purposes on entity objects alone, an object modeling approach to data warehousing and data marts utilizing dimensional analysis, would proceed from Use Case Modeling to identification and modeling of entity objects containing attributes specifying quantitative measures of the outcome of the actions taken to implement business use cases. If the use case modeling is done carefully, it helps the analyst to arrive at objects encapsulating quantitative measures that are indicated by, and closely coupled with, the use cases, business sub-processes, and business processes that are, in turn, tied to strategic goals and objectives. So, the analyst would emerge from object specification with a set of classes identifying data warehouse and/or data mart outcome measures. Figure Five shows the chain of analysis that precedes selection of the fact table grain in an object modeling approach.

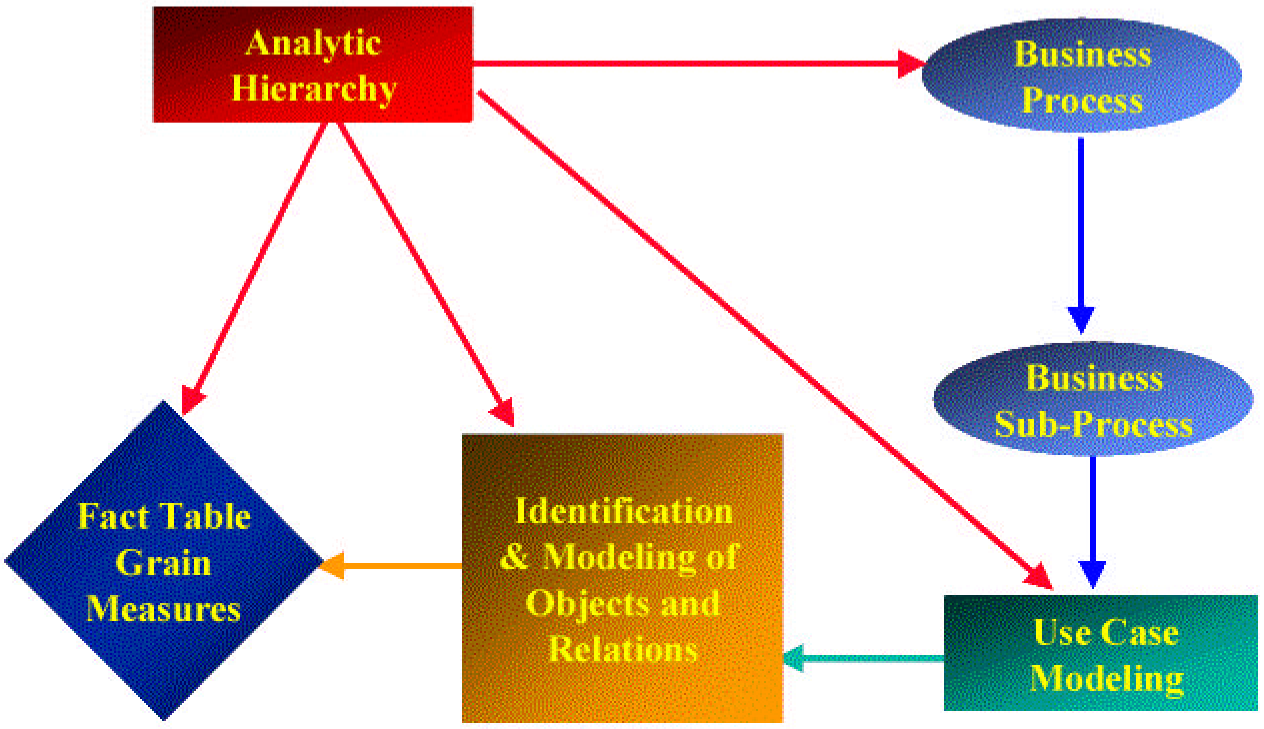
Analytic Hierarchy
Business Process
Business Sub-Process
Fact Table Grain Measures
Identification & Modeling of Objects and Relations
Use Case Modeling

Figure Five -- The Chain of Analysis Preceding Selection of the Grain of the Fact Table

If the purpose of the application is to provide a high performance decision support system, object modeling would proceed to model the associations of classes having outcome attributes, with other (dimensional) classes that locate, identify, describe, classify, subset, or otherwise characterize the instances of the outcome classes. This activity produces an entity object model in which outcome entity objects (n-ary association classes) are related to dimensional entity objects -- in other words, a Dimensional Object Model (DOM). In Figure Six, the outcome entity is an association class named Academic Success containing educational outcome attributes. The members of the association are the classes Student, Project, Status, Household, School, and Time.

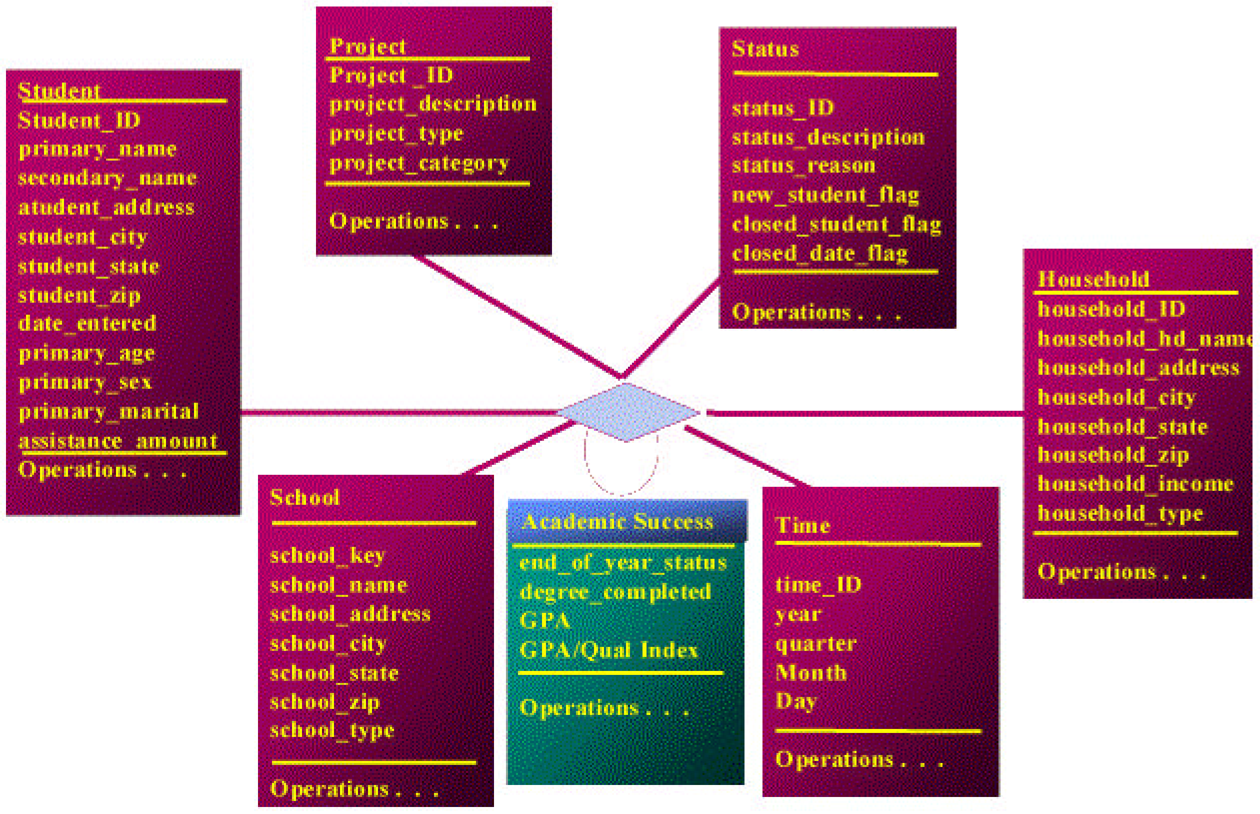

Figure Six -- A Dimensional Object Model

But entity object types are not equivalent to entities in an E-R model. Nor is a DOM equivalent to a DDM. Entity objects are characterized by behavior providing for access to data, and perhaps for computations of various kinds, and they encapsulate the methods that produce this behavior in response to external stimuli. Relational entities, on the other hand, represent tables, purely passive containers for data, and since they are not objects, are independent of behavior (operations) and methods.

Still, entity object classes are similar to entities in that they are associated with instances (objects) that have attributes with values, and *entity object classes can be mapped onto the entities of a DDM, while object relations can be mapped onto DDM relations, by implementing the required foreign keys and retaining unique object entity IDs in corresponding DDM entities.* In short, this suggests an approach to DDM through use cases, and dimensional object modeling of object types and relations, followed by a mapping of the resulting DOM onto a DDM to specify the Logical (dimensional or star schema) Model for storage in a relational database. Figure Seven depicts mapping a DOM onto a DDM (a simple star schema).

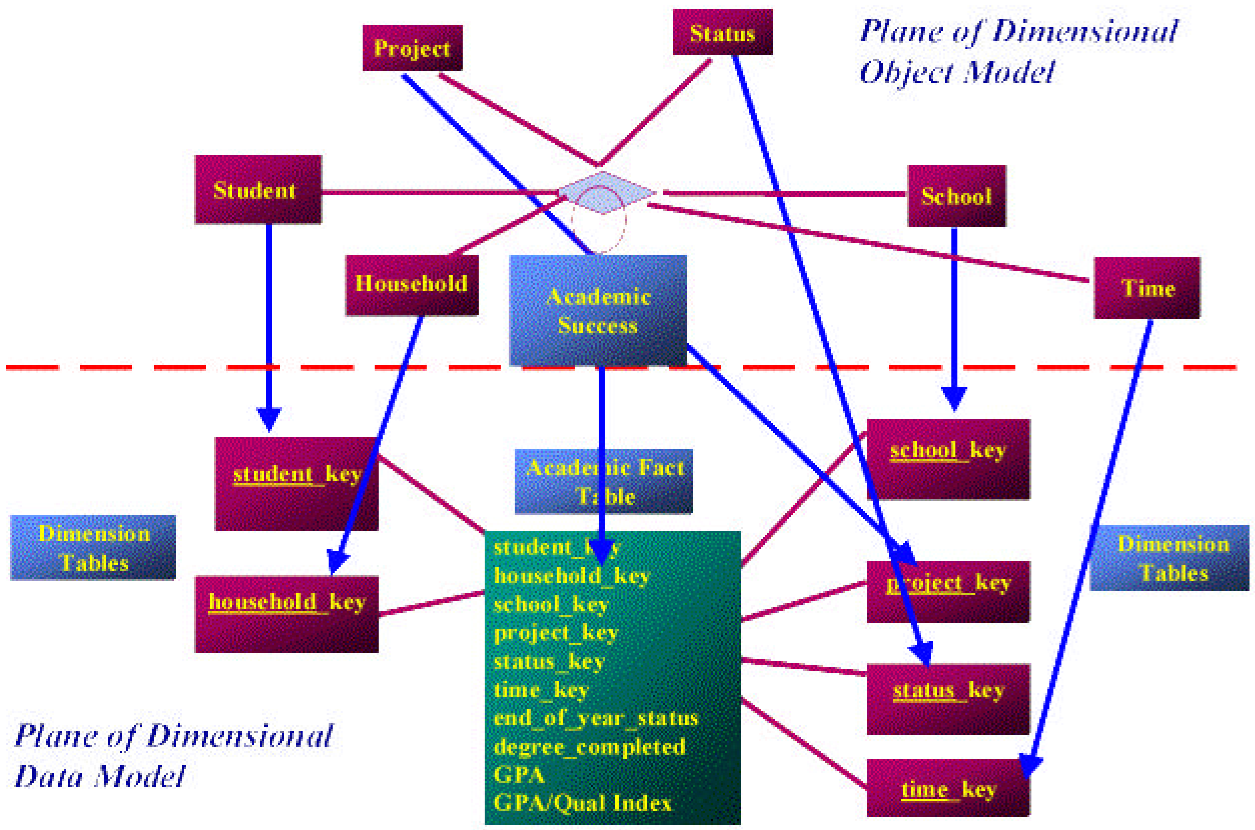

Figure Seven -- Mapping a DOM onto a DDM

In the figure the arrows from each entity object to each relational entity represent a one-one mapping of the data aspect of entity objects to corresponding entities. That is, classes identify entities, and their attributes correspond one-to-one with entity attributes. Dimensional Object IDs are mapped directly to Dimensional Entity primary keys. The association class in the DOM is mapped to the Fact Table in the DDM, which is at the center of the star schema. Finally, the Object IDs become the candidate foreign keys comprising the composite primary key of the Fact Table.

The example should make clear that one can formulate dimensional data models by working through an OOSE approach, arriving at Dimensional Object Models and mapping them onto entities to arrive at Dimensional Data Models. This procedure for arriving at DDMs through OOSE, is an alternative to the pragmatic approaches currently being employed, and ought to be preferred because of its systematic character and connection to the enterprise's hierarchy of goals and objectives.

Advantages of a DOM Approach

If the final logical and physical models to be specified are DDMs, why do we need to approach them by using an approach like the DOM approach just outlined? Why not just construct the DDM and skip the preliminaries? Here are four reasons.

First, a DOM approach provides a tighter conceptual association or coupling between strategic goals and objectives and the DDM construct. The result is that the DDM is more likely to be valid in the sense that it is the DDM users need to support their decisions.

Second, an approach through DOM provides an object layer to a data warehousing application unifying behavior and data within its object components. This unification directs the modeler toward explicit consideration of the specific object behavior that the data warehouse design should incorporate into entity objects, and by doing so opens up the potential for data warehousing to provide increased decision support functionality to users.

Third, the DOM approach provides conceptual consistency with a distributed objects approach and tighter enterprise-wide integration of data warehouses, data marts, operational data stores, Data Mining Servers, staging areas, DSS Servers, Web Servers and the increasingly diverse set of application servers constituting the data warehousing architecture.

And Fourth, the DOM approach provides a much better correspondence with the historic purposes of DSS and the more recent OLAP orientation to business intelligence than does the DDM approach. It does this because the DDM approach focuses only on data and its proper structuring to maximize performance in segmenting the database. But there is much more to the requirements of DSS and OLAP than this. In the DDM approach additional DSS requirements must be addressed in an ad hoc manner, or by using an additional conceptual approach to address such requirements. DOM, in contrast, because it addresses behavior, process, and levels of conceptual abstraction, as well as data structures, provides a unified approach to DSS and OLAP and therefore is the natural approach for those who want to use data warehousing to fulfill each of these orientations. I will explore each of these advantages in more detail below.

Tight Coupling -- Validity

Approaches to DDM normally exhibit only a loose association with strategic goals and objectives of the business area being modeled. Data warehousing practitioners may ask users about the mission of the organizational subdivision they work in, or about how they measure success, or about the major business processes of their business. Based on answers to questions on these subjects and others about data resources and existing systems, a decision will be made about the grain of the fact table, the necessary dimension tables, or the information packages that will define a data warehouse or data mart.

Such procedures are ad hoc, and do not provide a tight coupling between a business's goals and objectives and its DDM or data warehouse. They do little to guarantee that the data warehouse resulting from this procedure will be relevant to the gap between goals and actuality that motivated its construction.

In contrast, the DOM approach outlined, begins with a specification of business goals and objectives, moves on to business processes, sub-processes and use cases that will fulfill them, and then, based on the use cases, specifies objects that are modeled to support these use cases, sub-processes, processes, and ultimately the goals and objectives. In short, the DOM approach provides a much clearer justification for the relevance of the object model to the problem the users actually have. It supports the validity of the data warehouse application in the sense that the data warehouse construct will be *relevant* to the problem it was designed to solve.

Unification Of Behavior And Data In Object Components

A pure DDM approach is fundamentally an approach in which tables are associated with SQL-standard methods to support set-oriented processing of data for the purpose of returning result sets in response to SQL language queries. When applied to data warehousing, this approach supports efficient segmentation of the tables that are the targets of such queries. In turn such segmentation can support some data analysis expressed in reports based on the segmented data, and can support extraction of response sets for use by data mining servers and modeling data marts.

An approach through DOM, on the other hand, provides an object layer to a data warehousing application unifying behavior and data within its object components. This unification directs the modeler toward explicit consideration of the specific methods that the data warehouse design should incorporate into entity objects. In a data warehousing situation employing a relational database for persistent storage, these will exceed the methods offered by relational databases in functionality. This is true because the entity objects involved will encapsulate both the SQL-standard methods for data manipulation and retrieval, and additional methods relevant to object layer DSS and OLAP support.

This openness of the entity object layer to inclusion of additional methods as part of the process of object modeling, in contrast to the standardized SQL-tuple layer's restrictive nature, needs also to be emphasized. We don't at this point know all of the methods that should be included in the object layer for facilitating DSS and OLAP. But it is an advantage of the DOM approach that it is open to whatever methods it proves useful to add, and also that it provides a rigorous theoretical context, namely the object modeling context, for adding them.

Consistency With A Distributed Objects Approach And Tighter Enterprise-Wide Integration

DOM is part of a broader distributed objects approach to enterprise-wide systems integration. The general trend of software technology is toward development and application of this type of approach to solve the islands of information problem in enterprise-wide information systems. While the existence of this trend is not by itself a compelling reason to apply a distributed objects approach to data warehousing, it does mean that pressure will exist to adapt data warehousing practice to increasingly prevalent distributed object-based systems. Already, reports have begun to surface of corporations adopting O-O technology standards, and of this decision impacting the direction and orientation of data warehousing projects.

Apart from the existence of the trend however, the development of data warehousing itself suggests that enterprise-wide data warehousing systems will need to be integrated with distributed object processing layers running on top of relational and other forms of legacy persistent storage mechanisms. This follows from the proliferation of data warehouses, Relational data mart (ROLAP) servers, Multidimensional data mart (MOLAP) servers, Vertical technology OLAP (VT-OLAP) servers, Operational Data Stores (ODSs), Data Mining servers, Transaction servers, Web servers, ETT servers, and other application servers.

Currently, the prevailing architectural approach to this diversity is ad hoc systems integration. But this cannot be expected to be a lasting long-term approach because it is not standards based, and lacks reusability. The alternative is distributed O-O -based systems integration using either the DCOM or CORBA, or some combination of these two standards.

In this context, it is worth mentioning the Enterprise Data Mart Architecture (EDMA) approach, being developed by Hackney [6], among others. The point of this data warehousing approach is to pursue a strategy of implementing data marts incrementally throughout the enterprise, while relying on an EDMA specifying (a) enterprise subject areas, (b) common business dimensions, (c) a common repository of business metrics, (d) a common set of business rules for calculating common metrics and identities, (e) a common set of source systems of record, and (f) a common semantics. While this concept of EDMA does not require an object layer to implement it, an object repository functioning within a distributed objects architecture is a particularly effective way of implementing such an EDMA. Indeed, an enterprise standard requiring registration of new data marts or other components within an existing CORBA-based distributed objects network would allow evaluation of the new component for consistency with the standard established by the existing object repository.

Better Correspondence With The Purposes Of OLAP And DSS

Finally, as a consequence of the advantages already discussed, the DOM approach has the advantage of providing a better foundation for achieving goals of the DSS and OLAP constructs that data warehousing systems are intended to implement. It does this because the DDM approach focuses only on data and its proper structuring to maximize performance in segmenting the database. But there is much more to the requirements of DSS and OLAP than this.

While there is no agreement on the definition and characterization of DSSs, the following is a representative definition that relies heavily on Ralph Sprague's [7] characterization which presents manager's, builder's, and toolsmith's views of DSS. A Decision Support System (DSS) is an *adaptive information system* containing three primary distinguishable components. These include: (1) a database of values of attributes (field values) ranging over cases or units of analysis (records); (2) a modelbase containing the analytical or statistical models supporting measurement, evaluation, monitoring, and forecasting; and (3) a software system for manipulating data and models to perform data management and model maintenance and to produce new models, model revisions, analyses, queries, reports, predictions and forecasts. The data content of a DSS is contained in its database; but its intelligence is contained in its modelbase and in the ability of its software system to learn and to adapt to new data inputs.

Using this or other even remotely similar definitions of DSS, it follows that DDM does not provide a model of a DSS, but only of its database. It does nothing to provide the modelbase necessary for a DSS. It leaves the modelbase to data mining, analytical modeling or other data warehousing related functions. DOM however, supports specification of measurement modeling, prediction, planning, forecasting, and evaluation methods as part of the object modeling process. This is a radical difference from DDM and suggests an entirely new and as yet unexploited modeling process.

A similar point to the above applies to OLAP. Let's take Nigel Pendse's and Richard Creeth's ***F***ast ***A***nalysis of ***S***hared ***M***ultidimensional ***I***nformation (**FASMI**) definition of OLAP [8] as the basis of discussion, since it seems to reflect a greater industry consensus than earlier definitions.

In terms of FASMI, OLAP-consistent modeling needs to support:

- delivery of most responses to queries in (F) five seconds, with simple queries coming back in less than one second, and all but the very few most difficult queries taking no longer than 20 seconds;
- access to records one needs to perform a variety of analyses (A), including
    - database segmenting (or subsetting according to the criteria specified in a query, also known as "dicing"),
    - rotating (also known as "data slicing,") to examine a different view of the multidimensional data being queried without having to reassemble the view from more basic data,
    - aggregating or disaggregating multidimensional data to display higher or lower levels in an analytic hierarchy such as time periodicity, geography, or business/social/ government organizational hierarchy (known as "rolling up" or drilling down).
    - predictive modeling,
    - time series analysis,
    - measurement modeling combining database attributes (to develop good measures of important abstractions such as corporate or government performance, customer satisfaction, strength of customer bonding, and many other properties not adequately measured by a single database attribute or variable),
    - nonlinear, even fuzzy, causal and structural modeling combining measurement and causal models (to develop impact modeling and further refine predictive modeling),
    - short- and long-term forecasting,
    - automated exploratory data analysis (data mining) to aid Knowledge Discovery in Databases (KDD),
    - validation analysis (see my White Paper, "Data Mining and KDD" for an explanation of why validation of data mining is necessary) of patterns

discovered through data mining;
- a multidimensional (M) conceptual view of the data in the application;
- a comprehensive organization of all the data (I) that may be needed to achieve KDD.

While DOM supports all of these OLAP requirements through methods specification and integration with its related distributed objects framework, DDM fails to support the last seven analysis (A) requirements ranging from predictive modeling to validation. The point for both OLAP and DSS orientations is that DDM is a partial modeling approach which must be supplemented with many additional approaches, while DOM is a comprehensive approach that integrates data, methods, and behavior into the modeling process.

This paper has focused on an approach to data warehousing in which the primary persistent storage mechanisms remain relational databases. Since this is the case, I hope the point is clear that the proposal is to add DOM to the arsenal of data warehousing, not to remove DDM. The end result of the proposed approach is a better, more valid DDM, implementation of data warehouses and data marts in an RDBMS, and integration of RDBMSs in an enterprise-wide distributed object framework providing an integrative architecture for DSS and OLAP processing.

References

[1] As evidenced by the proliferation of books, seminars, conferences, institutes, projects, newsletters, journal articles, and mailing lists on the subject, as well as survey results reporting that upwards of 90% of U.S. corporations either plan or have data warehouses or data marts.

[2] Ralph Kimball, The Data Warehouse Toolkit (New York: John Wiley & Sons, Inc., 1996), Pp. 15-16

[3] Ivar Jacobson, Maria Ericsson, and Agneta Jacobson, The Object Advantage: Business Process Reengineering with Object Technology (Reading, MA: Addison-Wesley, 1995), P. 343

[4] Ibid.

[5] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Overgaard (Reading, MA: Addison-Wesley, 1992), Pp. 169-187

[6] Douglas Hackney, Understanding and Implementing Successful Data Marts (Reading, MA: Addison-Wesley, 1997), Pp. 52-54, 183-84, 257, 307-309

[7] Ralph H. Sprague, Jr. "A Framework for the Development of Decision Support Systems," MIS Quarterly, 4, no. 4 (December, 1980), 1-26

[8] "What is OLAP?" The OLAP Report, revised February 19, 1998, @http://www.olapreport.com/fasmi.htm

Biography

Joseph M. Firestone is an independent Information Technology consultant working in the areas of Decision Support (especially Data Marts and Data Mining), Business Process Reengineering and Database Marketing. He formulated and is developing the idea of Market Systems Reengineering (MSR). In addition, he is developing an integrated data mining approach incorporating a fair comparison methodology for evaluating data mining results. Finally, he is formulating the concept of Distributed Knowledge Management Systems (DKMS) as an organizing framework for the next business "killer app." You can e-mail Joe at eisai@home.com.

[Up] [Data Warehouses and Data Marts: New Definitions and New Conceptions]
[Is Data Staging Relational: A Comment]
[DKMA and The Data Warehouse Bus Architecture]
[The Corporate Information Factory or the Corporate Knowledge Factory]
[Architectural Evolution in Data Warehousing]
[Dimensional Modeling and E-R Modeling in the Data Warehouse]
[Dimensional Object Modeling] [Evaluating OLAP Alternatives]
[Data Mining and KDD: A Shifting Mosaic]
[Data Warehouses and Data Marts: A Dynamic View]
[A Systems Approach to Dimensional Modeling in Data Marts]
[Object-Oriented Data Warehousing]