Based on nearly 10 years of data modeling experience, in 1995 I published Data Model Patterns: Conventions of Thought in the United States (cf. Hay, 1995). It was simply a compendium of standard data models I had built for standard business situations that I encountered. Two years later, Silverston et al. (1997) published The Data Model Resource Book: A Library of Logical Data and Data Warehouse Designs, which contained their versions of similar models. Fowler (1997) added an object-oriented flavor to the set of common data modeling patterns with Analysis Patterns: Reusable Object Models.

In all cases, we were writing from practical experience, describing models that we found ourselves recreating over and over again for various (mostly American) clients. None of us was working from an academic background in computer science, and none of us saw any particular reason to do academic research on the subject. Both the need for these models and the shape of the models themselves were relatively self-evident to all of us.

And we have been successful both at selling books and at marketing ourselves as conceptual data modeling consultants.

Mr. Silverston’s and my modeling approach was derived from work done in the UK, and we both have traveled some, but our orientation, at least, was clearly toward the American and UK environments.

Imagine my surprise, then, to come across the articles in this volume, which presented an extensive bibliography of academic works primarily from Germany and Austria on the subject of data model patterns—here called “reference models.”

The chapters here go beyond what Mr. Silverston, Mr. Fowler and I have done in three ways:

First, several of the chapters address process models as well as data models. In my experience, it is more difficult to identify generic processes beyond the first few levels of a function decomposition diagram, and I for one found data models more interesting as a way to
understand the subtle nuances of what makes a company what it is. Still, addressing process
and/or function models is clearly a worthwhile effort.

Second, several of the chapters address the problem of managing pattern configurations. As I
describe more informally in my book, it is often the case that there isn’t a single “right” answer
for a generic model. It is often the case, however, that there are reasonable alternatives to
select from when using the patterns to build a specific enterprise’s model. The management
of these alternate patterns is a task that has not been well addressed in the past.

Finally, and perhaps most important, several chapters specifically address the question of
how to evaluate the quality of reference models. This, of course, is a continuation of the
more general question of how to evaluate the quality of models in general. The industry
thus far has not been as articulate as it could be in addressing either question.

What is a “Reference Model”

The term “reference model” as it is used here is problematic. Different articles describe
very different things under this title.

Reference Data or All Data?

In the United States, the meaning of the term “reference model” is different from the one
used in this volume. There it refers to any data model that portrays only reference data—the
entity classes that describe an organization’s configuration. These include primarily the “___
type” entity classes, such as EVENT TYPE, ORDER TYPE, ACTIVITY TYPE and so forth, and not any
transaction data elements. While both our patterns and some of the reference models de-
scribed here make extensive use of these entity classes, they also include, in a generic sense,
the transactional data types as well, such as ORDER, EMPLOYMENT, ACTIVITY and so forth.

That is, our patterns represent a generic version of all sides of an enterprise data model,
not just the reference data.

Data or Process?

It is clear that different practitioners address different dimensions of the business. The two
primary dimensions are data and activities. Mr. Silverston and I have addressed data in our
patterns for the simple reason that it is harder to find universal processes that are common
to all businesses. Or at least harder to find any in sufficient detail to be interesting.

In this volume, both data and process models are addressed, to varying degrees. In at least
one case, the models focus on aggregate functions more than detailed processes, but in oth-
ers, process models are included. In some cases, only process models and no data models
are discussed.

Ovidiu Noran’s article on “Reference Models in Enterprise Architecture” asserts that “The
reuse of EA [enterprise architecture] knowledge [involves] identifying commonalities in
enterprise models (EMs) and grouping them accordingly. Their common features can then
be abstracted into a partial enterprise model (PEM) or reference enterprise model.” In his
discussion, he does not specify whether such models are of data or process. Indeed, later in the article he asserts that “Reference models represent reusable templates for human roles (organisational), processes (common functionality) or technology (resources, e.g., IT).” This, in turn, supports an “enterprise engineering tool,” which in turn is used to build an “enterprise model,” but is not shown to be a template for that model per se.

Dr. Noran’s overall point, however, is about a “generalized enterprise reference architecture, that is, an architecture that can model activities involved in the implementation of a project spanning over a part or the entire life of an entity.” That is, rather than addressing specific patterns in data or process structure, he is concerned with creating a template that describes the system development effort overall. His model is in fact a meta model addressing the nature and artefacts of the development process itself.

**Detail or Overall Functions?**

This raises a separate question about the definition of “reference model.”

In these articles, there is considerable variety in the level of detail involved. In the article presenting a reference model for industrial enterprises, only a high-level function breakdown is presented, with emphasis on describing the overall functions involved. Along with that is representation of bills of material, routing and equipment as data categories, but without any data model details. By contrast, the article describing retail reference models contained both basic process and data models.

**Description or Prescription**

An ongoing controversy in the United States, that was alluded to only obliquely here, is whether a data model is *descriptive* or *prescriptive*. Is it simply a description of the nature of an enterprise, or is it a blueprint for building new systems. In the United States, at least, the term “data model” is used in several different contexts to mean different things. Some view data modeling as being equivalent to database design. While it is important for the model to reflect the business world that systems will address, the model itself is the design of database artifacts. (IBM Rational’s product, for example, a few years ago added a “data modeling” facility, which is nothing other than a tool for modeling relational database design.) As such, the model is prescriptive.

I and others, meanwhile, contend that a data model (well, a “conceptual” data model, anyway) should focus entirely on the business at hand as a first step in the development process. It should be done completely without regard for the technologies that might be used to implement its findings. In this sense, a data model is descriptive.¹

Graeme Simsion has written a lot on this subject. See, for example, his article in *The Data Administration Newsletter*, “You’re Making It Up: Data Modeling Analysis or Design” (Simsion, 2006).

The answer in the data modeling world at large is, of course, both. In the data world, a conceptual model describes the underlying structure of a business in terms of the data it
uses. A logical (some call it the physical) data model then describes either table and column or object class structures for a system design. Similarly, in the process world, an “essential data flow diagram” describes business processes without regard for the technologies that might be invoked to carry them out (McMenamin & Palmer, 1984). A “to be” process model, then, describes the automated or manual mechanisms that will be implemented to carry out the processes.

The reader’s view of the models being addressed affect his or her view of the way to approach reference models. Reference models can be created for all of these views. A conceptual reference model is concerned with generic concepts that will be made particular in describing actual enterprises. A general concept might be “facility, which in the petroleum industry means any collection of physical equipment configured to carry out a function. Specific company (conceptual) models might then describe the configuration of their facilities in more detail. In the logical model, there might be a “facility” table in a piece of commercial software, with the ability to tailor it as necessary for each customer installation.

Logical reference (data) models, on the other hand, are more specific database designs, but they have been created with the idea that as designs they will be reused.

Most of the articles in this volume beg this question by not addressing the actual modeling in great detail, but at least one (“configuration management for reference models”) did go so far as to assert that what the authors considered a “conceptual model” is “…the result of a construct done by a modeler, who examines the elements of a system for a specific purpose such as redesign of an organization or the development of an information system at a given point of time with a specific language.”

Managing the Modeling Process and the Models

Several of the chapters discuss the process of developing models. Ostensibly, they are about developing reference models, but the steps they describe apply to the creation of any model: interviews, draft models, review of the models and so forth. In fact, in my experience, it is rare to set out to develop reference models in isolation. Both Mr. Silverston and I had over ten years experience developing enterprise models for clients before we wrote our books describing the models we inferred from our experiences. Fowler’s (1997) Analysis Patterns similarly came from his actual modeling experiences.

As Gamma et al. (1995) describe the origin of their Design Patterns book, “Design patterns capture solutions that have developed and evolved over time. Hence they aren’t the designs people tend to generate initially. They reflect untold redesign and recoding as developers have struggled for greater reuse and flexibility in their software” (p. xi).

One measure of quality that came up more than once in these articles was the extent to which a reference model is used by those who develop more specific enterprise models.

This is similar to the general data model quality criterion that a model is only “good” if it is used as the basis for the design of systems. In this case, though, the question is whether a reference model has been used as the basis to create other models. This is less compelling, since, if a reference model is derived from a specific model, by definition it already is sufficient for that situation. If it is then generalized enough to cover a second situation, by definition it now is suitable for at least two situations. If it is then generalized for a third
situation, it is pretty likely that the generalization should now cover not only those three, but indeed most situations.

The heart of this process, however, is that for developing the specialized models in the first place—the processes described by the authors here apply.

The issue of configuration management is more difficult. Between the most general and the most concrete specific model there are families of models that deal with variations in business situations. These could vary in the level of generality, the granularity of sub-types or language. Underlying patterns still apply, but in applying them to specific situations, it is necessary to make decisions between alternatives.

Several articles described this problem, although it is not clear to me that any of them provided a clear solution. Technology for managing models is still relatively primitive, which means that at best, the skill of the modeler in selecting from his or her intellectual toolkit will be key to success for some years to come.

---

**Technology**

This raises another issue: Technology. This is problematic in three areas:

The first problem is that, as an industry, we still haven’t agreed on modeling notations. In the field of entity/relationship modeling, there is the original Chen (1997) notation, information engineering (developed by Finkelstein (1989) and James Martin), SSADM (cf. Eva 1994, promoted by the Oracle Corporation, and Barker (1990)) and UML (Rumbaugh et al. (1998)). In addition, the technique originally developed in the Netherlands called NIAM (cf. Nijsen, 1976) and promoted in the United States as Object Role Modeling (cf. Halpin, 2001) takes a different approach. Each of these has its promoters, and, indeed, each is particularly well suited for different purposes. That UML is becoming popular in the object-oriented community does not mean that it will be accepted readily by those already proficient with Information Engineering. As one who is particularly prejudiced in favor of a particular technique, I think I can speak for all equally prejudiced practitioners when I say that I am not going to give it up willingly.

Note that this is also an issue in the area of function and process modeling. The data flow diagrams of the 1970s have been largely replaced by business process diagrams, IDEF0 diagrams and use cases. Each of these has its promoters as well.

This will undoubtedly be a problem for some years.

The second problem, which is related to the first, is one of the tools available for creating models. Of those on the market, very few are able to accommodate more than one notation. A few can convert a conceptual entity/relationship model into a design UML model, but even this is not a smooth process. The notation that prevails may be determined by the modeling technology that prevails.

The third area is the technology used to convert data models into database or object designs. In this regard, the technology is relatively mature, although changes in development practices are making it difficult for tool vendors to keep up. As business rules become more important, the tools that allow specification of rules and generation of systems from those
specifications will become progressively more important. But they don’t necessarily link to the CASE tools.

**Market for Reference Models**

In reading these essays, the question prominent in my mind is: Is the IT industry ready for all of this? It is clear that many years of research are behind what is written here, with extensive thought having gone into the best ways to both create and deploy reference models.

My personal experience, however, makes me wonder about the willingness of industry to make use of this research.

First, there is the question of industry’s willingness to adopt modeling techniques at all. In my experience, there are many companies that think this is a good idea in principle, but are not prepared to invest even in the amount of time and resources that would be required to carry out a simple modeling exercise. They want to see something tangible in the way of program code as quickly as possible.

The current movement toward “agile” development techniques is in response to what some people consider too much effort spent on modeling at the expense of actually producing systems. Techniques such as “extreme programming” advocate simply developing a small piece of code, passing it to the user for evaluation and then modifying and adding to it as necessary. Drawing models is seen as a waste of time. This is a movement that is meeting resistance, but it is growing.

Many companies (and their information technology departments) are not sophisticated when it comes to modeling, so considerable salesmanship is required just to be allowed to do it in the first place.

In this context, reference models are valuable because a consultant who has them in his head can make the case that a tailored enterprise model can be completed in a matter of a few months instead of the year or more that people expect. But this presupposes the models already exist. The elaborate processes of developing them, as described in the essays contained here, would already have already taken place.

As mentioned, much of what is written here applies to data modeling in general, not just to the development of reference models. It is important to interview subject matter experts, gain consensus for the model and provide a means for maintaining it. These are not issues just for reference models.

One thing missing from the essays here is discussion of the skill of the modeler. Surveys have shown that given the same problem, ten data modelers will come up with ten different models (Hobermann, 2006; Shanks et al., 1993). This makes the question of data model quality problematic. I contend, however, that, given a particular level of abstraction, the modelers will come up with the same model. At any particular level of abstraction, the contents of a model are ultimately constrained by the reality of the things being modeled.

Where the data modeler’s skill comes into play is in determining an appropriate level of abstraction. The more abstract the model, the more general it is and the more adaptable it is to variations in the business’s situation, but the farther it is from the concrete language of the business. The more concrete it is, the more familiar and understandable it is to the people who will be using it, but the more vulnerable it is to changes in the business.
In 1999, I attempted to package the models in *Data Model Patterns: Conventions of Thought* into a product that would allow a customer to simply read the models into a CASE tool, and then modify them to meet its particular requirements. It included documentation that the customer could also tailor. The product was not a commercial success. No one was confident enough of their own abilities to make use of such a tool.

On the other hand, in the years since then, my consulting business has thrived as people invite me to produce company-specific enterprise data models—which I can do quickly because I have the patterns in my head.

This leads to a question about the need for the extensive project management and configuration management techniques described here for reference models. A good reference model is a view of the enterprise that is most effective when it resides in the mind of a data modeler. It is the skill of the data modeler that determines the usefulness—the success—of that model. The project management skills are required for the rest of the project.

## Conclusion

Clearly the disciplines described by the academic articles in this volume are long overdue their implementation in real companies. As mentioned, however, much of what is prescribed here represents simple good practice in the development of any kind of model.

The development of data model (and process model) *reference patterns*, however, must fundamentally remain an inductive process, based on extensive experience with real, specific models.

*David C. Hay*

*Essential Strategies, Inc., USA*

## References


---

**Endnote**

This is why, by the way, I vigourously objected to the concept of “object-oriented Analysis.” To me, analysis of requirements is just that, and should be done without regard for whether the findings are implemented with object-oriented technology, COBOL programs, or something else.

---

**Dave Hay**, a veteran of the information industry since the days of punched cards, paper tape and teletype machines, has been producing data models to support strategic information planning and requirements analysis since the mid-1980s. He has worked in a variety of industries including, among others, power generation, clinical pharmaceutical research, upstream and downstream oil production and processing, forestry, banking, foster care and broadcast. He is the founder and president of Essential Strategies, Inc., a consulting firm dedicated to helping clients define corporate information architecture, identify requirements and plan strategies for the implementation of new systems, including data warehouses. More recently, the company has become deeply immersed in the problems of defining, specifying and implementing metadata repositories. Hay’s background in philosophy and data modeling leave him perfectly situated to participate in the new/old field of semantics. His work with clients is usually concerned with defining an organization’s semantics. A pioneer in the use of standard data models for standard business situations, he is the author of the book, *Data Model Patterns: Conventions of Thought*. He has brought his considerable experience in requirements analysis into play in writing *Requirements Analysis: From Business Views to Architecture*. His latest book, *Data Model Patterns: A Metadata Map*, describes a comprehensive schema encompassing all aspects of metadata. He is a member of DAMA International, the International Oracle User’s Group, the Oracle Development Tools User Group, as well as local chapters of both. He has spoken numerous times at events sponsored by these groups and others. Hay’s bachelor’s degree in philosophy is from Claremont McKenna College, and he has an MBA in quantitative analysis from New York University.