

# Logical Data Models for SOA Information Exchange

By  
Steven Lemmo  
CTO & Founder  
[ObjectRiver Inc.](#)

## Introduction

In the computer industry the same patterns keep emerging with new names and different advocates. Model Driven Architecture is promoted as though it is a new revelation for modeling business information, when in reality it is just a new name for an established process. Data Architects have been defining harmonized logical data models for at least 15 years.

So why are SOA implementations struggling to define canonical business information for exchanging data? Many SOA implementations are re-inventing the wheel because they do not leverage work that has already been done in other parts of the organization.

## SOA Data Exchange

The term “uncoupling” is popular in SOA discussions. The goal of uncoupling is to separate the SOA implementation from databases, applications, and physical devices. This allows the implementation to remain neutral for the purposes of interoperability. The way to achieve this uncoupling is to work with the business to define the generic data constructs needed to describe new business processes. These generic data constructs must be robust enough to satisfy multiple back-end applications, yet harmonized to allow application-independent interoperability. Uncoupling is the true promise of SOA, where new business processes are written and packaged to leverage existing application functionality.

The foundation of the uncoupling is to reverse engineer the business into a harmonized data model that can be used to define generic business processes. This is why most SOA advocates preach the business top-down approach for designing enterprise SOAs.

## Guerilla SOA

Many vendors will have you believe that the business top-down approach requires a major investment in SOA infrastructure. You need to buy a repository, enterprise service bus, testing tools, and development environment to build your SOA architecture. You're also going to need to manage, monitor, secure, audit, and authorize business services. Many businesses are buying infrastructure, assuming that this will make them successful in defining a robust set of services.

In reality, the entire aforementioned infrastructure does not help you work from the business top-down to define a harmonized data model for building services.

The current blogs about Guerilla SOA speak about creating SOA at the department level, by defining interoperable services that can be built without expensive infrastructure. They go as far as to say that you should be uncoupled from your SOA infrastructure! They are not saying that you don't need infrastructure, because as the number of services increases infrastructure will be needed. Guerilla SOA speaks more to the design of well thought out services without the complications of large expensive infrastructure getting in the way of, or dictating, the design.

Guerilla SOA may be a little controversial because some think that departmental initiatives might create disjointed services that are not readily reusable. On the other hand, if the barrier to creating services is lowered, the better designed services will rise to the surface and succeed. We believe Guerilla SOA could be wildly successful if the services are developed in conjunction with a harmonized SOA data exchange model. This will promote departmental development of services that use the same underlying vocabulary.

## **Logical Data Models**

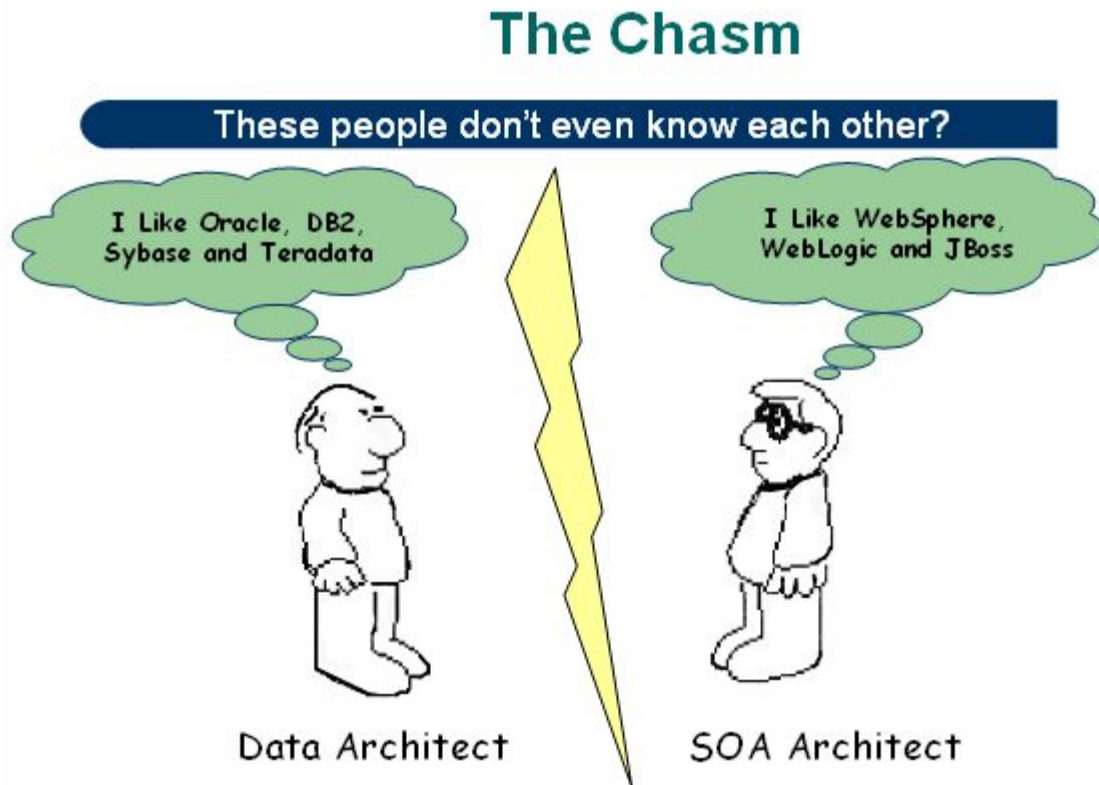
Logical Data Modeling (LDM) is the well accepted practice of designing business data models that are independent of application and physical database implementations. LDMs are conceived by Data Architects, Business Analysts and end users who analyze business processes and information for the purpose of defining a harmonized (logical) data model that represents the business. The models are designed from the business top-down, and the business information is not influenced by applications or database implementation. Free of these complications, assumptions and constraints, the LDM is a harmonized, uncoupled data model.

Logical data models are so uncoupled that you can purchase complete LDMs from third party companies. These purchased data models are specific to vertical industries. The LDMs are uncoupled from the company, and can be quickly adapted to represent the company requirements. LDMs also contain type definitions called domains, and master code definitions which validate fields within the model. This is exactly how industry standards like ACORD are created, to define the data exchange format of Life Insurance Policies among Customers, Vendors and Financial Institutions.

Logical Data Models are typically defined by the business and act as the contract for creating a physical database implementation. What if the LDM was used to persist data to the network instead of a database? This is exactly the kind of change control that is needed for exchanging information with an SOA. By using the LDM, the business can create a contract for the SOA data exchange layer. This contract will help with versioning and let the business define the system from the top-down.

## The Chasm

If this is so obvious why hasn't somebody figured this out yet?



Let's start with two groups of people who couldn't have a more diverse set of technical skills, data architects and SOA architects. They typically don't work in the same building, or even in the same state. One has been working on harmonizing business information for years, while the other is just discovering the process and does not know that the problem may already have been solved.

The reason these people are disconnected is they don't agree on the modeling technique. The data architect uses relational data modeling tools to describe business information, while the SOA architect uses object modeling tools to describe business objects. The problem that both groups are struggling with is how to represent their information in a canonical format for purposes of exchange within an SOA implementation.

The solution is that both parties need to represent their information in a hierarchical format for the purpose of exchanging business information with XML. What the SOA architects don't know is that relational data, when normalized, contains natural hierarchies that can represent business objects.

## Logical Business Objects (LBO)

Any normalized logical data model can be broken down into a series of hierarchies of the relational tables in the data model. Using primary or alternate keys to determine the hierarchies, you can typically find about 15-25 business objects within any given data model. These are "logical business objects" because they are hierarchies of entities that are linked through defined relationships. The following table is an example of the business objects found in data models from three vastly different industries

<b>Retail</b>	<b>Financial Security Master</b>	<b>Insurance</b>
Contact	Security	Person
Location	Security_Type	Organization
ContractRelation	Exchange	Agreement
RetailSale	Industry	Policy
Contract	Issuer	Claim
Order	IssuerGroup	Location
Product	SecurityGeography	Contact
Inventory	RatingCodeAssociation	Agent
Account		
Institution		
Depository		

**Figure 1. Business Objects found within industry data models.**

The business objects discovered in these data models correspond to the nouns in the business vocabulary. If you were to eavesdrop on analysts from one of these industries, you would hear these nouns in their business conversations.

These newly defined LBOs can be used as business objects for exchange between applications in an SOA implementation. Because they are derived from the logical data model, they are independent from the implementation of applications and databases. Think of the LBOs as defining the data structure for the network instead of the physical database.

The business objects, domain types, and master code values that the LDM defines can all be used as parameters when defining services. The logical data model can serve as the SOA data contract for creating enterprise services.

## SOA Software stack

The SOA software stack is mind boggling to most people because they tend to confuse the infrastructure with the actual services. In the following diagram, we break out the infrastructure from the actual service implementation.

The diagram defines three layers (represented in blue) of the SOA implementation that are used as a best practice for defining enterprise business services.

- Logical Business Objects**  
 The bottom layer defines logical business objects, master codes, and domain types that were discovered from the LDM. All of these artifacts are from the LDM, with strict versioning controlled by the business.
- Uncoupled Business Services**  
 These are the actual definitions of the services. This layer is capable of defining types, records, and additional exchange data information, but its primary purpose is to define the actual services within the business processes.
- SOA Orchestration / Business Process**  
 The promise of any large SOA implementation is to have a library of services that address many logical business objects, such as Customer, Product, Order, Invoice, etc. The services are combined to create long-running business processes. When packaged together, they can represent a new application that leverages existing applications.

## ESB / SOA Stack

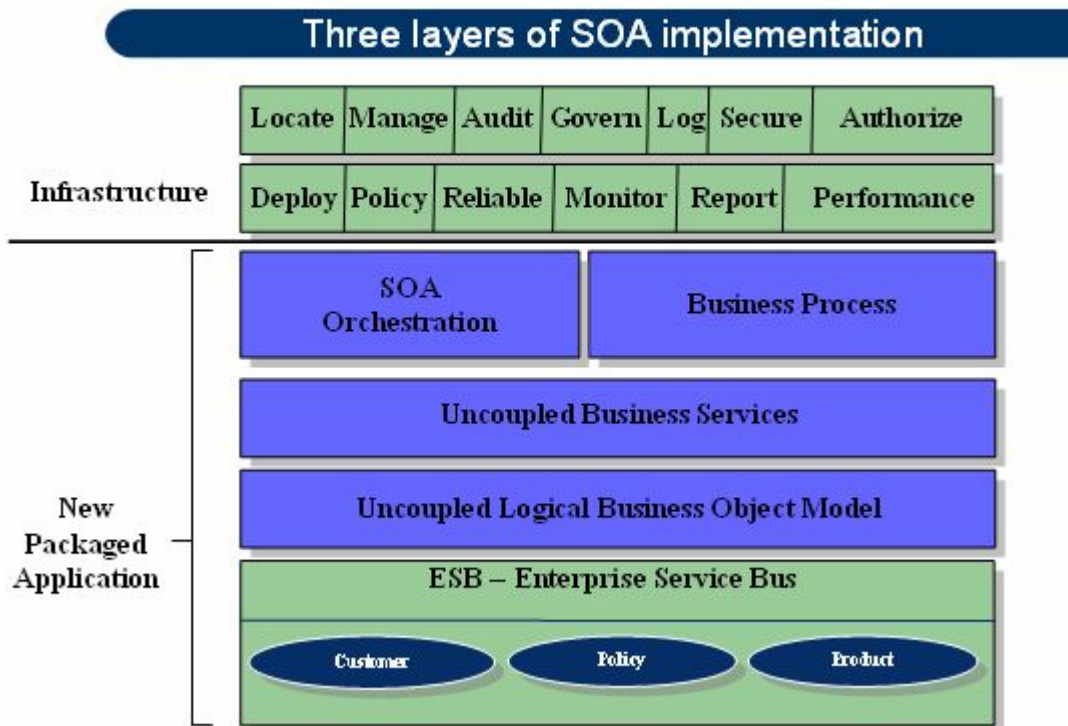


Figure 2. SOA Software Stack.

Think of Figure 2 as an application that runs on top of a bus architecture. Newly defined business processes are created by combining finer grained uncoupled business services. Logical business objects from the LDM serve as the exchange data standard for interoperability. It is important to notice that the logical business object layer is uncoupled from the business services layer. They are defined by different groups, and are separate contracts for the data vs. the services.

## **Summary**

Harmonizing business information is a well defined process called Logical Data Modeling. LDMs contain hierarchies that represent logical business objects. The logical business objects can form the contract for a SOA data exchange layer.

Many articles, blogs, and papers promote the idea of aligning business data in conjunction with developing an SOA implementation. The LDM will formalize the SOA data layer, which will allow the development of interoperable services.

Using the LDM as the contract for SOA data exchange allows us to model logical business services. Since the data is uncoupled, it is easier to define uncoupled, or logical, services. Logical services can be orchestrated to create flexible new applications for the agile business.