Towards a Model-Based Characterization of Data and Services Integration

Paul C. Brown, Principal Software Architect, TIBCO Software Inc.

Overview

Data and services integration is an ongoing practical challenge in the IT world. In most situations today, it is very much an ad-hoc exercise that relies heavily on human interpretation to get the job done. This is a shame, because there is a lot of model-based standards work that, if appropriately combined and applied, could be leveraged to both represent human understanding and potentially automate many error-prone manual development activities.

This paper examines data and services integration from three perspectives: data, services, and service utilization context. Each perspective explores the realities that are encountered in the field.

Data Realities

Structure of Information: The information encountered in building real-world solutions generally has an inherent network structure (Figure 1), yet most of the representations used for inter-component communications are tree-structured (Figure 2a). These tree structures are subsets of the inherently network-structured information. Complicating things, different interfaces often require different tree-structured subsets from the same core network of information (Figure 2b).

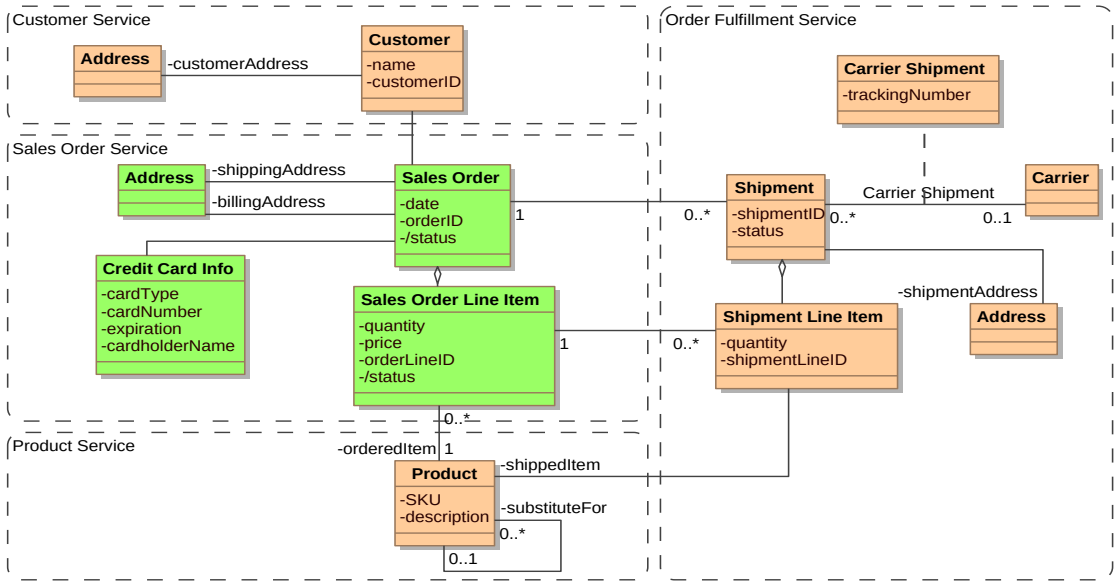
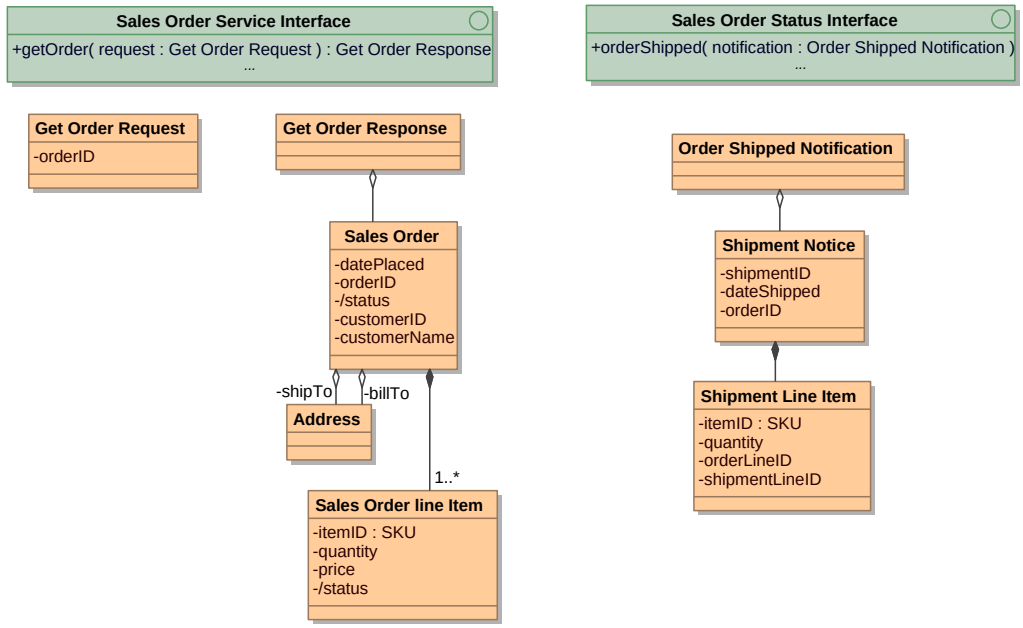


Figure 1: Inherently Network-Structured Information



a) getOrder() Operation

b) orderShipped() Operation

Figure 2: Two Tree-Structured Subsets at Interfaces

This dichotomy extends to the technical representations of information used in solutions. Some of these representations are inherently tree-structured, including XML, JSON, and file structures. Others, such as database schema, handle network structures gracefully. The reality is that solutions need to deal with both kinds of representations and to relate the information between them.

Canonical Data Structure Myth: One Size Does Not Fit All. The goal of canonical data structures is to minimize the number of representations in use. Often this desire is interpreted to mean a single data representation, but in reality this approach rarely works. Consider the representations of the Sales Order in the previous figures. In Figure 2a the entire Sales Order is represented, while in Figure 2b just the orderID appears as a field in the Shipment Notice. Typically a solution will require at least three representations of an entity: a full representation, a minimal representation (i.e. an identifier), and a human-readable minimal representation (i.e. an identifier and a human-readable name or description). These three representations, of course, need to be related to one another.

Vocabularies Vary: It is common for different parts of the business to use different terminology for similar (if not identical) concepts. The sales organization refers to products, often identified by SKU's. Manufacturing refers to assemblies and packaged products. Service refers to field-replicable units. In practice, it is unrealistic to try to drive to a single set of terminology across an entire enterprise. Efforts to do so never converge, and thus never produce useful results for the enterprise. The reality is that terminology varies, and the various terms need to be related to one another.

Information is Often Distributed and Replicated: Of necessity, information is often distributed across different components and portions of the information may be replicated (Figure 3). Such situations require the ability to refer to the different instances of the information and the service design necessary to maintain the consistency of the information.

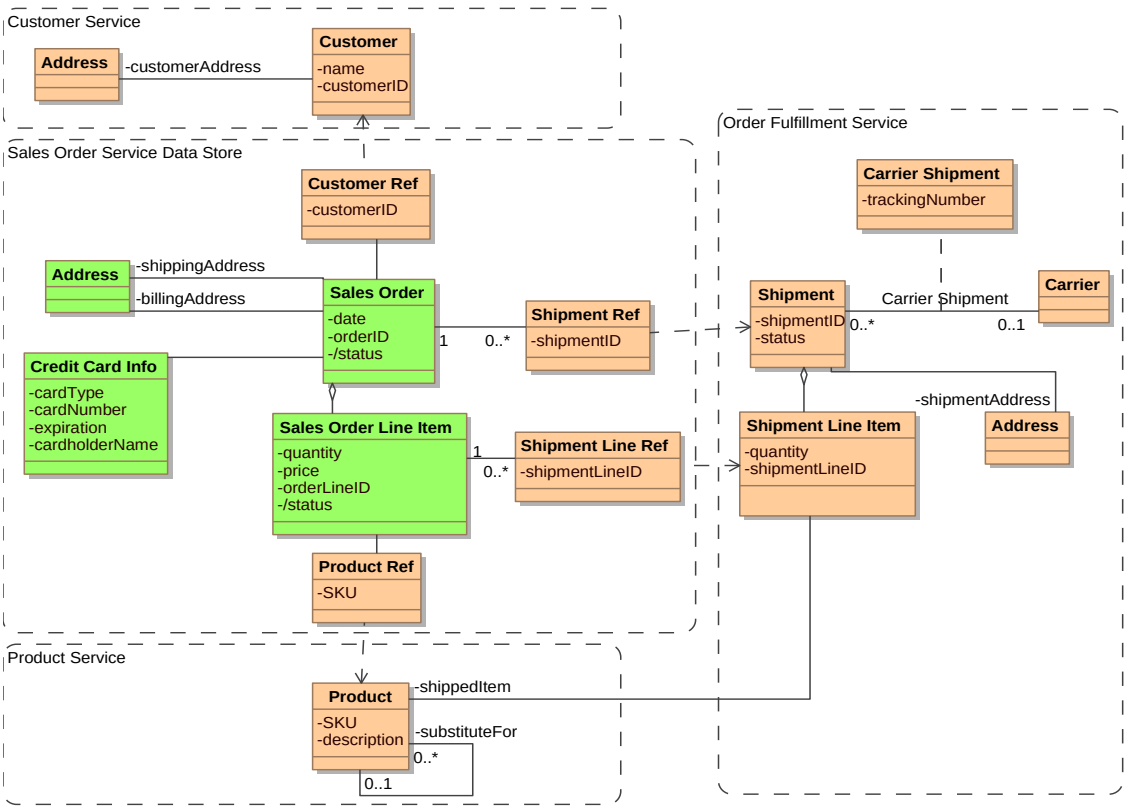


Figure 3: Replicated Information in Storage

Information Evolves: The set of information relevant to an enterprise evolves over time. While good analysis and design result in most changes being additive in nature, nevertheless it must be possible to relate old and new versions of data structures.

Service Realities

Vocabularies Vary: Service and service operation naming terminology tends to vary even more than for information concepts and relationships. This terminology tends to reflect the usage of the service operations in the business processes that they support. Some terms (create, read, update, and delete) tend to be heavily overloaded: their meaning can only be made clear by considering the context in which the term is being used.

Many Service Operations are Not Pure Functions: Most high-value services manage things (generally information). Thus defining the semantics of their operations requires relating the operation's data structures not only to each other (functionally) but also to the information statefully retained by the service (or its underlying services).

Many Operations are Not Independent (Orthogonal): The operations of a service often have dependencies on one another. Some of these constraints can be expressed as simple relationships between operations, such as the constraint that an item cannot be read, modified, or deleted before it has been created. Others, however, require more information to express. Consider the cancellation of a Sales Order, when business rules dictate that the order cannot be cancelled after it has shipped (a reference to state information).

Services often contain cached data: Services often contain cached information – information for which other components are the systems of record. The Sales Order Service's references to Customer, Shipment, and Product in

Figure 3 are examples of this. In such cases it is necessary to be able to represent the fact that the returned information is not the system-of-record's golden version, but another copy that may be inconsistent under some circumstances. Beyond this, in a complete service design it must be possible to represent the cache maintenance strategy (a process) and the interfaces required to support it.

Services Evolve: As business requirements change, services evolve. To support this, it must be possible to characterize different versions of interfaces and to relate the versions to one another.

Service Utilization Context Realities

Usage Scenarios: Services provide value only when they become a working part of one or more business processes. As such, it is necessary to indicate the manner in which the service operations are intended to integrate into the overall business process. When this participation involves multiple service interactions, it may be necessary to represent the interactions at different levels of detail. Figure 4 shows such an overview for the Sales Order Service participating in an order-to-delivery scenario. There is a need to show the required behavior under different execution circumstances and, in the case of reusable services, different execution contexts.

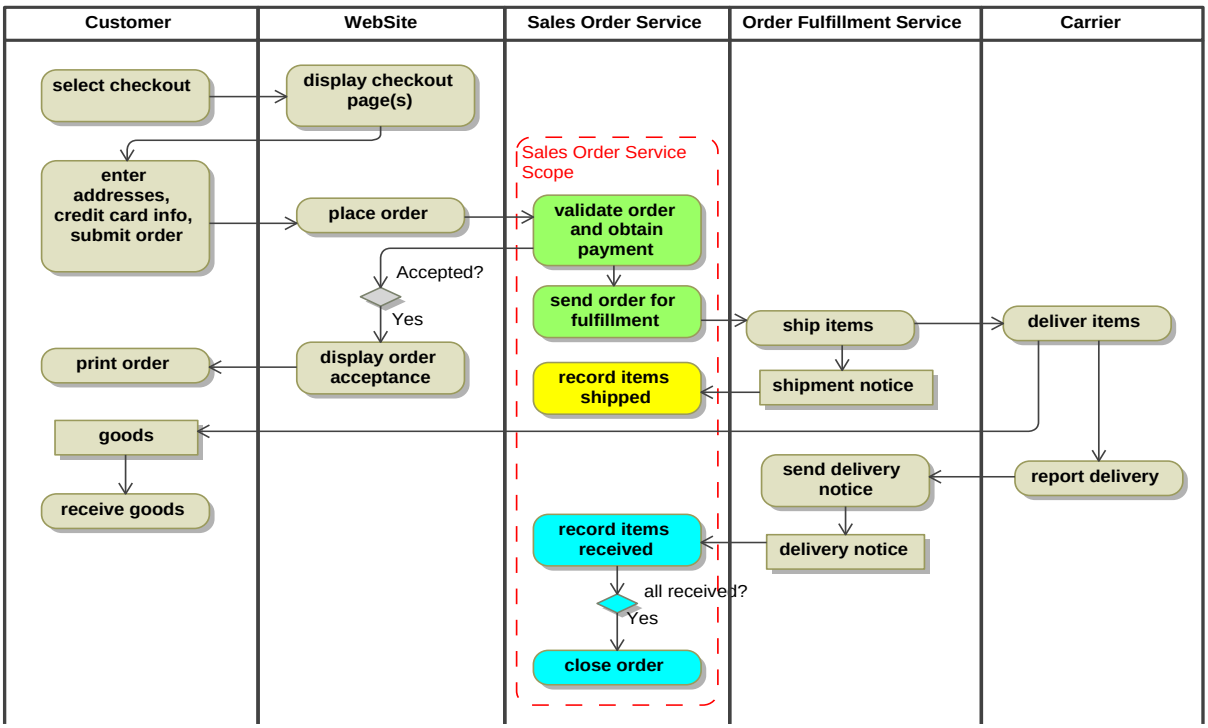


Figure 4: Order-to-Delivery Overview Scenario Showing Sales Order Service Participation

Representations at a finer level of granularity may be required to show details. Figure 5 shows the placeOrder() operation in context, indicating its observable actions and interactions with dependent services.

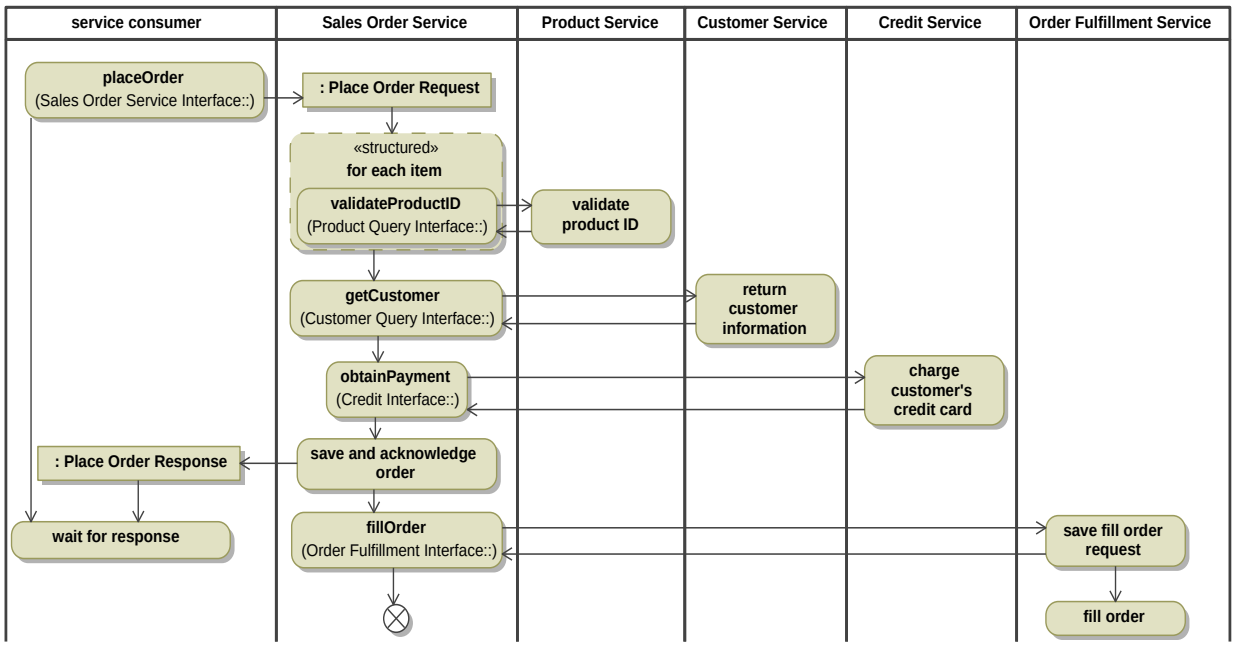


Figure 5: Scenario Showing placeOrder() Interaction Details

Protocol Semantics: There are a number of protocols commonly used for inter-component communication. These include REST, SOAP, XML over JMS, and file transfer, to name but a few. The communication interaction patterns associated with these and other protocols need to be clearly understood by the service user. The behavioral symptoms that occur under breakdown conditions need to be clearly understood as well.

Coordination: Service users (and service providers) need to understand the manner in which solution activity can be coordinated with service activity. Coordination can vary from fire-and-forget interactions up through distributed transactions. The coordination behavior under breakdown conditions (loss of a component, loss of a communication) needs to be understood as well.

At the solution level, the means by which responsibility is handed off from one component to another must be clearly understood. Part and parcel with this is the need to understand the ways in which component and communications beakdowns can affect the handoff and whether or not work can be lost due to those breakdowns.

Summary

Both service consumers and service providers need to clearly understand how a service is intended to fit into solutions. This understanding ranges from data structures through interfaces up to service integration into solutions. It requires understanding data at rest (i.e. state information) as well as data in motion. It requires understanding service interfaces and the behavioral semantics of service operations. It requires understanding how service activity can be coordinated with solution activity.

While it is possible to describe these things with human language, model-based approaches (done properly) provide a more concise and precise approach. It is believed that existing UML and SBVR models, if integrated, are sufficiently rich to represent many of the required structures and behavior, though extensions may be required to represent mappings involving computations.