

CHAPTER | 5

Business Models and Information Flow

In any business application, there is (or at least, there *should* be) a distinct understanding of what business problem the application is meant to solve. For example, the side effect of a customer billing application is the generation of bills to be sent to customers. Other side effects may include registering accounting details and the calculation of a monthly usage report, but overall the application is meant to create bills.

Unfortunately in practice, as applications are modified, merged, and expanded, the high-level understanding of the business problem gives way to dependence on implementation details and decisions that impose artificial constraints on the system. By virtue of the structured algorithm design in which most implementers are trained, we impose a control structure on the way that information flows through the processing. But this control structure does not always reflect the true dependencies inherent within the original application, because, for instance, we may have decided to break up a single subprocess into two stages that could truly have been executed in parallel, but an implementation decision may force one stage to precede another.

And as the system evolves, technical employee turnover leads to a loss of knowledge about how that application really models the original business problem. Interim decisions have imposed a strict flow of processing control, which may no longer reflect the true data dependence of the application. So when they attempt to dissect the way information is being used within the system, most analysts might throw up their hands in frustration.

As a case in point, I remember a particularly difficult analysis project meant to identify performance optimization opportunities within a processing environment in which 40,000 programs were being run. Our goal was to look for ways to streamline the processing by finding programs that could be

run in parallel. In theory it should have been simple to figure this out, because the environment in which these programs were run required manual scheduling. Yet the imposed execution schedule did not reflect the true control and data dependencies among the programs; instead, the schedule was based on the historical clock times at which the critical programs had finished! In the absence of a proper business model describing the control and data flow, this proved to be an exercise in futility.

In this chapter, we will describe what the savvy manager needs to know about how to coordinate a business model information flow, why the traditional assembly line of data processing can make this difficult, and how to organize participants to cooperate in mapping and documenting these models.

The Business Case

Consider this scenario: You have been tasked with building a data mart for the purpose of analyzing a customer value portfolio based on all customer interactions, ranging from telephone inquiries to purchases, returns, customer service calls, payment history, etc. On the one hand, you must determine what organizations are going to be supplying data, how and when the data sets are to be supplied, and how the data is to be organized and modified for integration into the data mart. In addition, you must be able to manage the quick integration of new data sets when it is determined that they are to be included in the data mart. Alternatively, you must be able to manage the provision of information services to the business analysts, each of which may be logically or physically situated in a different location.

It would be difficult, if not impossible, to build this system without having a clear understanding of where the data is coming from, how it needs to be manipulated before it enters a data warehouse, what data is to be propagated along the data mart, and what kinds of applications are using that data. More importantly, after the system is built it is critical to have a blueprint of the way that information flows into and out of the system to provide a tracking mechanism to back up any conclusions that are drawn through data analysis. To get a handle on how to manage this environment, it would be useful to have a high-level model of the processes associated with populating and using this data mart.

The Information Factory

Most systems can be viewed as a sequence of processing stages fed by directed information channels in a manner that conjures up the image of an Industrial Age factory. In fact, there are many practitioners who have taken to describing information systems in terms of an information factory (most notably Bill Inmon, Claudia Imhoff, and Ryan Sousa in *The Corporate Information Factory*). Although this view may actually constrain our perspective of the use and value of information, it is a useful paradigm for how a business process can be modeled.

The Value of Modeling Information Flow

At this point you may be wondering: “This is a book on business intelligence. Why should I care about nonanalytical processing?” The answer is that business intelligence (BI) is not limited to a company’s interaction with customers, but instead includes knowledge accumulated from any collection of data consumers, such as internal (i.e., within the organization) and external (i.e., cross-company) business applications. As an example, consider a supply-chain interaction between your company and a collection of product suppliers. There is embedded business knowledge that can be derived from examining all details of those interactions, including measuring vendor sensitivity to your requests, response time, methods of delivery, compliance with contractual agreements, and conformance to just-in-time delivery issues. To extract this intelligence we must understand how we have implemented our business applications and determine what data we need to collect and where that information needs to be collected. The information flow model will assist in this determination.

Design versus Implementation

Traditionally, implementers are trained in algorithm design to break down each application into a collection of discrete processing stages that can be essentially implemented in isolation. When all the stages are finished, they are combined to form the complete application. But this process of discretizing the construction of applications leads to an assembly line model of information processing in the way that data and partial results are forwarded from one processing stage to another. These processes take data (e.g., a transaction stream or extracted records from multiple data sets) as

input and provide some product as output. That can be a physical product (such as invoices to be sent out to customers), a side effect (such as the settlement of a sequence of transactions), or an information product (such as a BI report).

To remedy the eventual effects of this development process, an important part of the methodology of designing and implementing a business application is modeling the business process as a way of guiding the algorithmic implementation. In fact, building this model is the first step in the process of exploiting information. This business process modeling incorporates descriptions of the business objects that interact within the system as well as the interactions between users and those business objects. The same concept holds true for analytical and intelligence applications, where the eventual product is described in terms of analytical use and benefit.

Benefits of the Business Process Model

There are some major benefits for building this model. One is that understanding an information flow provides logical documentation for the business process. Another is that it exposes potential for adding value through the kinds of analytical processing we discuss in later chapters. A third benefit of this business modeling process is in communicating user requirements to the implementation team. When a formal framework is used to describe a process, not only does it ease the translation of user needs into system requirements, but it also provides the manager with a high-level view of how control migrates throughout the system and how information flows through the system, both of which in turn help guide the dissection of the problem into implementable components.

More generally, an information flow, as embodied as part of a business process model, provides the following benefits.

- **Development road map:** Identifying how information is used and diffused helps direct the development of interfacing between the discretized execution components as well as tracking development against the original requirements.
- **Operational road map:** When the application is in production, the model provides a description of how any analytical data sets are populated as well as a launch point for isolating problems in operation. It can also be used to track and isolate data quality problems, map workflow and control back to information use, and expose opportunities for optimization.

- **Management control:** This model provides a way to see how information propagates across the organization, to identify gaps in information use (or reuse), and to expose the processes involved in information integration.
- **Calculation of return on investment (ROI):** This allows the manager to track the use of information, the amount of value-adding processing required, and the amount of error prevention and correction required to add value and to relate the eventual business value back to the costs associated with generating that business value.

Information Processing and Information Flow

In this section we look at a number of different kinds of processing paradigms and how we can view a business application.

Transaction Processing

Operations in a transaction processing system are interactions between a user and a computer system where there is the perception of an immediate response from the system to the user's requests. A commonly encountered example of transaction processing is the use of an automated teller machine (ATM).

Although there is an appearance of a monolithic system that responds to user requests, behind the scenes each interaction may involve a large number of interdependent systems. The concept of a transaction actually incorporates this reality: A transaction is really a set of operations grouped together as a unit of work, where no individual operation takes its long-term effect unless all of the operations can take effect. So, using the ATM example, before the bank allows the ATM to disburse cash, the user's account balance must be queried to see if there are sufficient funds, the ATM must be checked to see if it has enough cash to satisfy the request, the user's account must then be debited, and the cash can be disbursed. Yet if the result of any of these subsidiary operations indicates that servicing the request is infeasible, all of the operations must be rolled back—you wouldn't want the bank to debit your account without giving you the cash, nor would the bank want the cash to be disbursed without debiting your account.

In this case the information flow follows the thread of control as it passes through the individual interaction associated with each transaction. A rough view of this information flow can be seen in Figure 5.1.

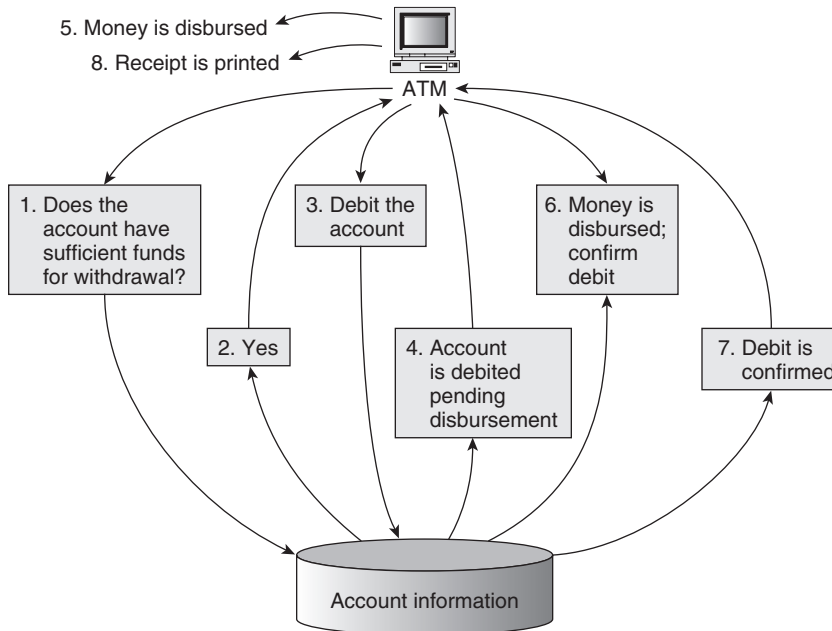


FIGURE 5.1 A transaction-based information flow.

Operational Processing

We will use the term *operational processing* to refer to a system that makes use of computers to control a process. An automated manufacturing line may have multiple machine components, each requiring system control instructions based on its internal operational requirements as well as depending on information inputs from other interconnected machine components within the entire system. For example, a potato chip manufacturing process contains a series of machines, such as a washer, a slicer, a fryer, a sorter, a flavor enhancer, and a packaging machine, each of which helps transform a potato into a collection of potato chips.

In this example, there is a lot of information required at multiple processing locations throughout the system to guarantee continuous, managed control of the system. Data about the individual interactions between sequential stages as well as systemic data need to be propagated to multiple controllers. To continue our potato chip factory example, each processing stage requires information about the flow of (unfinished) product from the previous stages. In addition, certain events will trigger auxiliary control

operations (e.g., the seasoning hopper volume falls below the required amount, triggering an alert and a pause in the assembly line). And global events can also trigger actions (e.g., the cooking temperature exceeds a safe limit, triggering a complete line shutdown).

Operational process information flows are likely to connect heavy sequential operational processing augmented by lightweight interconnections for exchanging control information.

Batch Processing

In contrast with transaction processing, batch processing takes collections of sequences of similar operations that are to be executed in *batches* (hence the name). Although both transaction processing and batch processing execute a series of operations, batch processing differs from transaction processing in terms of information flow in the granularity of application of each processing stage. A batch processing application is more likely to apply each processing stage to a set of data instances as a whole and then push the result to the next processing stage.

As an example, a company might accumulate transaction-based sales orders during the day but process those orders and prepare order fulfillment as a batch process at night. The fulfillment processing aggregates order line items by customer, determines packaging requirements, generates pick lists that instruct the warehouse workers what items are to be selected for each shipment, generates shipping labels with appropriate shipping vendor data, updates inventory totals, and generates orders to restock inventory, among other operations.

Batch processing information flows typically convey heavy data payloads between multiple processing stages, each of which performs a single component of the overall unit of work for the data collection. A rough view of a batch processing information flow can be seen in Figure 5.2.

Analytical Processing

Analytical processing involves the interaction between analysts and collections of aggregated data that may have been reformulated into alternate representational forms as a means for improved analytical performance. In this case, the information flow model most likely will take on two aspects: the flow of information into the analytical processing environment from its suppliers and the flow of information from the analytical processing system to its users. The first flow is likely to be more of an operational flow, in which data sets may

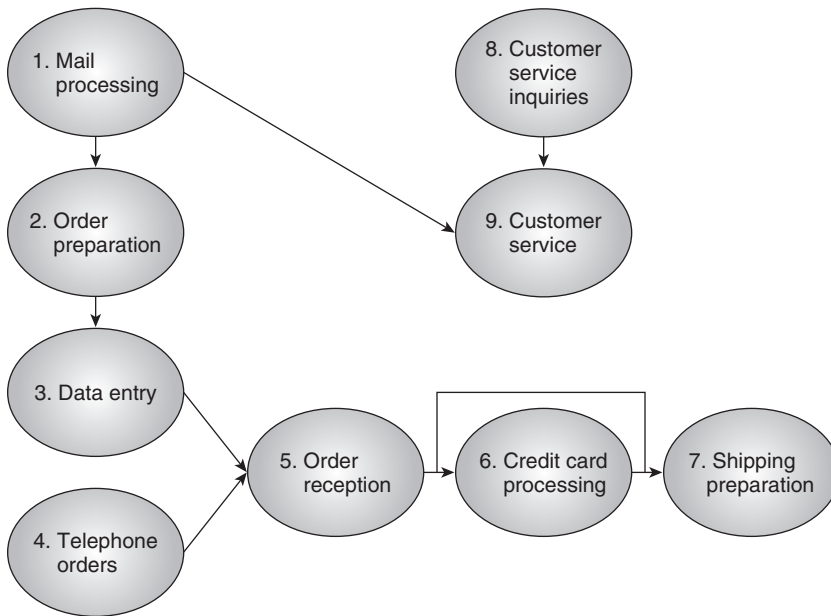


FIGURE 5.2 A batch processing information flow for order fulfillment.

be extracted and moved in large chunks to a staging area where those data sets move through different processing stages. And despite the BI aspect of the users' interactions, the information flow between the data mart clients may resemble a transactional information flow, with multiple analysts executing sequences of queries, although here there is less likely to be true transactions. A sketch of the information flow for analytical processing can be seen in Figure 5.3.

The Information Flow Model

The business process model reflects how a business process works; when we peer under the top layer of that model, what is exposed is the model for how both information and control are propagated through the business application. It is useful to have a formal way to describe the way data propagates through a system; in this section we will introduce a high-level information flow model.

An information flow model distinguishes the discrete processing stages within the process, describes how information flows through that system,

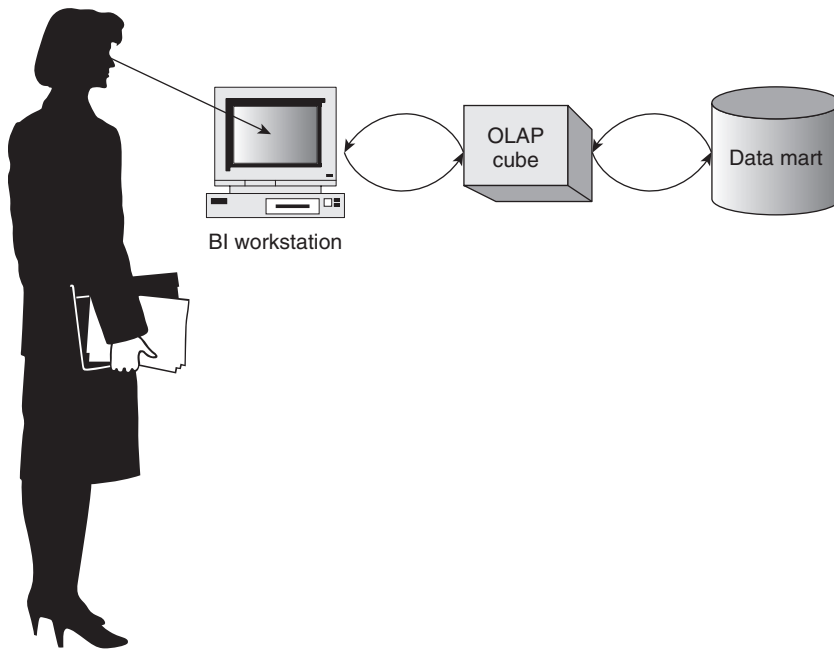


FIGURE 5.3 An analytical processing information flow.

and characterizes the kinds of data items that flow through the process. This model is valuable because it provides a basis for distinguishing between data dependencies, control dependencies, and artificially imposed implementation dependencies, which in turn can lead toward flow optimization, identification of bottlenecks, finding locations for insertion of data validation monitors, inserting data collection points for later analysis, and opportunities for increased business analysis points.

Information Flow: Processing Stages

In an information flow model, we distinguish discrete processing stages. Although the following list is by no means complete, we can characterize each processing stage as one of these classes.

1. **Supply**—the stage from which external suppliers provide data
2. **Acquire**—the internal stage where external data is acquired
3. **Transform**—a stage where information is modified to conform to another processing stage's input format

4. **Create**—an internal stage where new data instances are created
5. **Process**—any stage that accepts input and generates output (as well as generating side effects)
6. **Package**—any point at which information is collated, aggregated, and/or summarized
7. **Switch/route**—a stage that determines, based on some discrete set of rules, how to route data instances
8. **Decide**—a stage where some interactive (real or automated) agent's choice is captured
9. **Portal**—the delivery point for data that is meant to be consumed
10. **Consume**—the exit stage of the system

Information Flow: Directed Channels

Data moves between stages through *directed information channels*. A directed information channel is a pipeline indicating the flow of information from one processing stage to another, indicating the direction in which data flows. Our model is represented by the combination of the processing stages connected by directed information channels. Once we have constructed the flow model, we assign names to each of the stages and the channels.

Data Payload Characteristics

The last aspect of an information flow model is the description of the data items that are propagated between any pair of processing stages. The characteristics include the description of the information structure (i.e., columnar attribution), the size of the data instances, and the cardinality of the data set (i.e., the number of records communicated). More sophisticated models may be attributed with business rules governing aspects such as directional flow, validation, and enhancement as well as processing directives.

Usage in Practice

This section provides a few examples where value can be derived from modeling business process and information flow.

Information Valuation

In Chapter 2 we discussed the concept of the value of information; we can make use of a business process model to guide the determination of metrics

appropriate for measuring the value of data as well as identifying the locations for the insertion of monitors to collect those measurements. For example, if we want to measure how much a certain data set is used, we may want to tag the data at its insertion point into the information flow and to insert monitors at delivery points to check for the tagged item to tally usage statistics.

Root Cause Analysis

One example of the use of an information flow model is in identifying the source of a data quality problem. The effects of a data quality problem might manifest themselves at different stages within an information flow, perhaps at different data consumption stages. But what appear to be multiple problems may all be related to a single point of failure from earlier in the processing. By identifying a set of data quality expectations and creating validation rules that can be imposed at the entry and exit from each processing stage, we can trace back through the information flow model to the stage at which the data quality problem occurred. At that point we can follow forward through the information flow to find all processing stages that might be affected by this problem. Fixing the problem at the source will have a beneficial affect across the board, because all subsequent manifestations should be eliminated.

Operational Performance Improvement

Another use of an information flow model is to gauge both the strict control and data dependencies within the system, and the performance behavior for transferring data between processing stages as well as processing at each stage. An information flow model will show the true dependencies, which can then expose opportunities for exploiting task parallelism at the processing stage level. In other words, if there are two processing stages that are control independent (i.e., neither stage requires the completion of the other in order for it to begin) and data independent (i.e., neither stage's input is directly or indirectly derived from the other), then those two stages can be executed at the same time.

Large data set transfers also form bottlenecks, as do computationally intensive processing stages. If there are no data dependencies associated with the data flow or associated with each processing stage, then the business and information flow model can be used to explore opportunities for exploiting data parallelism. For example, if there is a large data transfer between two processing stages, it may be of value to break up the transferred data set into

chunks that can each be transferred over multiple physical input/output (I/O) channels.

Modeling Frameworks

Conceptual business process and information flow modeling can be actualized in different ways. In this section we'll explore some formal frameworks used for information and activity modeling, although these are just a sampling of the many frameworks available.

Use Case Analysis

Use case analysis, a process described by Ivar Jacobson in his book *Object-Oriented Software Engineering*, was designed to understand the nature of interaction between users of a system and the internal requirements of that system. A use case model specifies the function of a system and describes what the system should offer from the user's perspective, using three components.

- **Actors**, representing the roles that users play
- **Use cases**, representing what the users do with the system
- **Triggers**, representing events that initiate use cases

Whereas a use case describes a specific way of using the system by performing part of the function, a use case also represents a course of events that takes place when an actor interacts with the system. In the use case model, a trigger (which may occur as a result of an input data structure or an actor requesting an action, such as a report, but providing no input data, time, or some internal database or system event) is an event that initiates a use case. The collection of use cases constitutes a specification of the system. Embedded in the use case model is the conceptual business process model, although because the model is meant to help drive the requirements gathering for implementation, it may not be sufficient to represent the actual information flow model we discussed earlier.

Unified Modeling Language

As a successor to use cases, the Unified Modeling Language (UML) was developed as part of the Object Management Group's (OMG's) model-driven architecture. UML is a very rich descriptive framework that allows analysts to describe many aspects of system architecture. UML integrates some of the best notions of previous formalisms, including use cases.

UML can be used to develop business process and information flow models, especially because it has a facility for describing system behavior. In particular, state machines and activity graphs can be used to model process stage interaction. In a **state machine**, each state represents a situation where some condition is true. When a condition changes, the system is represented by another state; this behavior is modeled as a transition from one state to another within the state machine, which ultimately summarizes the entire system behavior as transitions between multiple states within the system.

An *activity graph* is a special kind of state machine that models a computational process in terms of how control flow and object flow affect transitions between states. It is the activity graph that could be best used for modeling information flow.

Integrated Definition Language

Integrated Definition language (IDEF) is a modeling language designed to describe functional processing and information flows. It comprises two descriptive standards: IDEF0, which is used to describe activity models, and IDEF1X for describing data models. Although IDEF is mostly used for system requirement analysis and workflow modeling, the IDEF language can be used for modeling information flow.

The basic IDEF0 activity modeling object in an IDEF model is referred to as an ICOM, an acronym for “input, control, output, and mechanism.” A general ICOM object can be seen in Figure 5.4. For modeling information flow, each ICOM object would represent a processing stage, with the input(s) and output(s) representing information channels and the control and mechanism describing the gating factors controlling the activity as well as the activity that takes place within the processing stage. The data that is being propagated along the inputs and outputs within the model is characterized using IDEF1X; each IDEF1X object describes an entity being modeled as well as relationships to other entities.

A complete system model, which embeds the information flow, is constructed as the outputs of one ICOM are connected to the inputs to other ICOMs. This sequential ordering of ICOMs exposes the operational dependencies inherent in the process and highlights the data dependencies. In addition, we can represent a processing stage at a high level and then decompose that stage into multiple ICOMs at a lower level, providing a hierarchical view through which the system may be drilled-down.

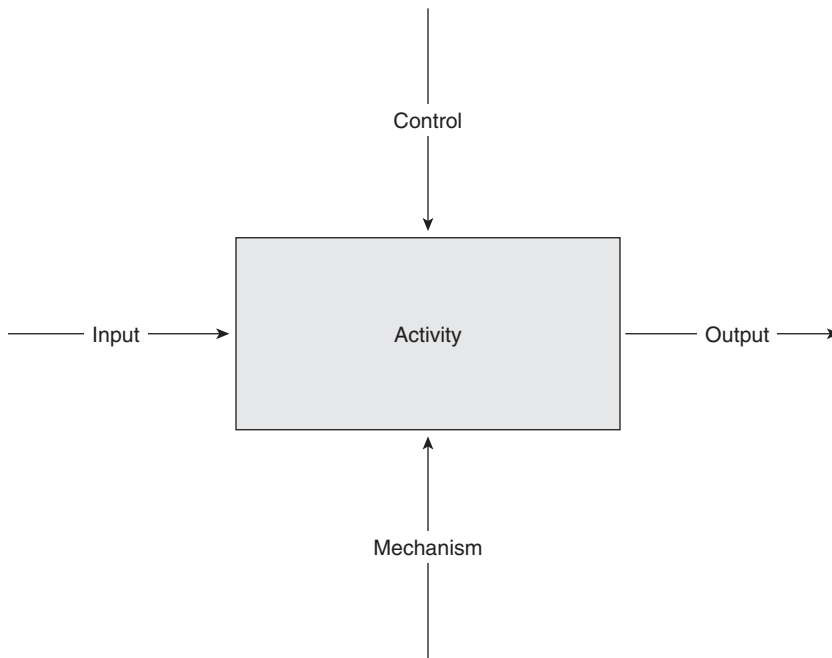


FIGURE 5.4 The ICOM object.

Management Issues

Probably the most critical management issue associated with business process and information flow modeling is the disconnect between the representation of the model and the actual implementation. There are a number of proposed standards for business process modeling, although the UML probably integrates the most popular methods.

The real issue is that despite the availability of CASE tools using UML or use cases, the output of these tools most likely will not be in a form that is easily integrated with legacy systems without a lot of external effort. In addition, even if these tools are used to generate code for actual use, there is little support for what could be referred to as the *roundtrip*, where generated code is modified or retooled for some reason but where that modification cannot be recaptured in the original modeling framework.

These issues imply that without a lot of investment in software infrastructure and management, the utility of a business process flow model is

limited to management purposes, and positive input must be made by the savvy manager to relate application development and operation back to the model. This means that there must be frequent synchronization between the documented model and the actual system itself.

Another issue is the sheer size of these modeling frameworks. Because of the attempt at converging on one modeling framework, attempts at standards include so much material that it is unlikely that you would ever be proficient in all aspects of the modeling language, which adds to the risk of using any particular choice. Also, the learning curve for these modeling languages may be quite steep, requiring an investment of time and money in training staff.

Learning More

The modeling of how information flows through the different implemented business processes in an organization can provide detailed information about how the business is run. In turn, this information can be captured as virtual metadata that can be fed into later analytical contexts. This information flow information can help in the determination of what data sets are to be incorporated into the enterprise data warehouse that will eventually feed the analytical applications. It also identifies the best locations within the information flow from which the data should be extracted.

The use of CASE tools to augment the documentation of information flow (and general workflow) is a good idea, although it is preferable to use tools that do not rely on proprietary formats and that can export their representations to other knowledge management tools.

For information on use case modeling, consider the following books.

- *Object-Oriented Software Engineering: A Use Case Driven Approach*, Ivar Jacobson et al., Reading, MA, Addison-Wesley, 1992.
- *Writing Effective Use Cases*, Alistair Cockburn, Reading, MA, Addison-Wesley, 2000.

For information on UML, see the following.

- *The Unified Modeling Language Reference Manual (UML)*, James Rumbaugh et al., Reading, MA, Addison-Wesley, 1998.
- www.omg.org/gettingstarted/what_is_uml.htm
- *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, ed 2, Martin Fowler and Kendall Scott, Reading, MA, Addison-Wesley, 1999.

