Chapter 1. Introduction to Document Engineering

Robert J. Glushko and Tim McGrath

1.0 Introduction

Twenty-four hundred years ago, a Middle Eastern farmer named Halfat paid his taxes by giving barley and wheat to King Artaxerces. The receipt that documented this transaction was recorded on a fragment of pottery.[1]

We don't use pottery, papyrus, and parchment anymore, and although paper hasn't gone away, electronic documents have replaced much of it. A corresponding evolution has taken place in the nature of document exchanges and the business processes they enable. Every major advance in transportation, communications, manufacturing, or financial technology has brought a need for different kinds of information flow. People have met these needs by developing specialized types of documents containing the required information. Letters of credit, bills of lading, paper currency, promissory notes, checks, and other new types of documents came into being in response to a business opportunity made possible by some advance in technology.

In the 19th century the telegraph and telephone made it possible to exchange information electronically and coordinate business activities at a scale vastly larger than before, leading to the rise of the modern corporation. The late 20th and early 21st centuries have witnessed the equally profound impact of the Internet (and related technologies such as the World Wide Web, electronic mail, and XML) on how businesses work. Now the web-based virtual enterprise can be open for business 24 hours a day, 7 days a week, with a global presence enabled by distributing people and resources wherever they are needed in either physical space or cyberspace.

Clearly, companies can't reliably achieve new business value by facing inward and focusing on efficiency. But they can't succeed in the dynamic 21st century global economy just by getting on the web either—they need to fundamentally rethink what they do and how they do it. Then they can begin to exploit their own strengths and start to rely on those of other organizations that may be halfway around the world but because of abundant bandwidth seem to be next door. When they face outward to create richer relationships with suppliers, customers, and business service providers and integrate their

internal business processes with those of their business partners, they create value they could not produce on their own.

And behind this flexible, adaptive business architecture remains the very simple and natural idea of document exchange. Documents organize business interactions and package the information needed to carry out transactions. The notion of documents as the inputs and outputs of business processes wherever they reside in the network is a technology-independent abstraction perfectly suited for the heterogeneous technology environment of the Internet.

We don't need to understand the technical nuts and bolts of XML and web services to appreciate the revolutionary power of this approach. Any business service that is invoked with an XML document and sends an XML document as its response can be a component in a service oriented business architecture. That business component can then be plugged into a new business model that may never have existed before.

But although the web services standards tell us how to package information into documents and where and how to route them, they don't tell us what any of the documents mean. We need Document Engineering to help us specify, design, and implement the documents that are the inputs and outputs of business services.

Document Engineering synthesizes complementary ideas from information and systems analysis, electronic publishing, business process analysis, and *business informatics.* Its unifying document-centric perspective helps us conceive and understand the new network-based business models made possible by the Internet and supporting technologies.

The essence of Document Engineering is the analysis and design methods that yield

- Precise specifications or models for the information that business processes require

- Rules by which related processes are coordinated, whether between different firms to create composite services or virtual enterprises or within a firm to streamline information flow between organizations.

Document Engineering provides the concepts and methods needed to align business strategy and information technology, to bridge the gap between what we want to do and how to do it. Describing business processes in terms of the more abstract notion of document exchanges makes it easier to understand the constraints imposed by legacy systems and technologies and to recognize the opportunities created by new ones if we focus on conceptual models of the exchanges rather than on how they are implemented. The expressiveness of XML for implementation models bridges the traditional gap between business strategy and its technology realization.

1.1 A Simple Business Transaction?

Imagine that you go into your local bookstore and notice a new book with an intriguing title, *Document Engineering* by Glushko & McGrath. You hand the clerk your credit card, and a few moments later you leave the store with your copy of the book. To describe what just took place, you might say that you purchased a book, using the single word *purchased* because the experience seemed like a single economic event or transaction taking place between you and the bookstore.

Now imagine that you are browsing the website at an Internet bookstore, GlushkoMcGrathBooks.com (from now on we'll just call it GMBooks.com). You navigate a few screens to select that new book with the intriguing title, *Document Engineering* by Glushko & McGrath. You add your credit card information and shipping address to the shopping cart form, and a few days later the book arrives by delivery service.

How would you describe what took place at GMBooks.com? At first glance the online experience seems equivalent to the bookstore experience, so you might describe it as "buying a book online." But if you analyze the online experience more closely, you can

see that it is composite service in which at least three separate transactions or exchanges of information occurred:

1. Your interaction with the GMBooks.com catalog to select the book you want to order.
2. A document exchange between GMBooks.com and a credit authority (a bank or authorization network like VISA or MasterCard) to verify your creditworthiness and charge your account.[2]
3. A document exchange between GMBooks.com and the delivery service with the instructions for picking up and delivering your book.

So what looked at first like a single event, "buying a book," turns out to be at least three separate events that have been combined in a particular sequence to create a composite business process (see Figure 1-1.)

This pattern is typical of many Internet retailers and completely invisible to you as the customer. But there may be even more involved here. The retailer taking your order doesn't have its own inventory of the books it offers in its catalog. Instead it maintains a virtual inventory, which consists of the books it can reliably obtain from distributors when a customer selects them from the online catalog. So other transactions that might take place are

4. An exchange between GMBooks.com and the distributor to confirm that the book you selected is available so that GMBooks.com can sell it to you and promise a delivery date.
5. The order sent by GMBooks.com to the distributor to obtain the book on your behalf.
6. The request for delivery (or forwarding instructions) sent from the distributor to the delivery service with the instructions for delivering your book. This document exchange takes place instead of Exchange 3 because the order taker never has your book!

This simple example, contrasting a "bricks and mortar" bookstore and an online bookstore, illustrates the disruptive force of the Internet on traditional business models. A virtual company created by using services provided by separate businesses can be created more rapidly, more flexibly, and at a lower cost than a traditional store can.[3]

The new business model of the virtual store changes both the business processes of the traditional model and the document exchanges required to carry them out. Redesigning and realigning these into a new business model requires the concepts and methods of Document Engineering.

A virtual company can be created rapidly, flexibly, and at low cost

1.2 The Extended or Virtual Enterprise

Before we go any further, we must point out that when we say things like *enterprise* or *business model*, we don't mean to rule out governments, educational institutions, or nonprofit institutions. *Business* is shorthand for "purposeful, systematized activity to create and exchange value" and can apply as well to government, educational, and nonprofit entities.

Unlike the physical bookstore, which might exist as a single entity in a fixed location, the online bookstore, GMBooks.com, functions as an extended or virtual enterprisemerges from the coordination of the activities of numerous independent businesses that collaborate to achieve their interlocking goals, as illustrated in Figure 1-1. This coordination takes place by the exchange of information between the retailer and book distributors, shippers, and credit authorities. The retailer doesn't need its own books and delivery trucks—it can replace inventory and equipment with information.

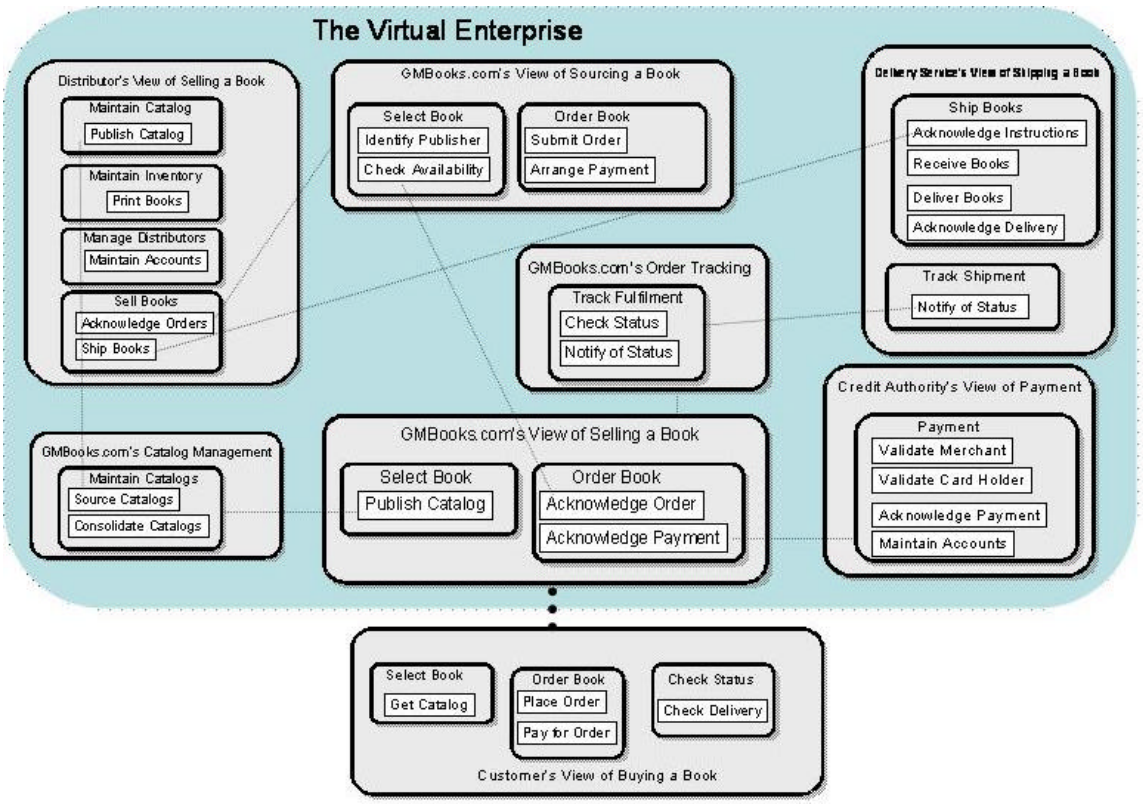

Figure 1-1. The GMBooks.com Virtual Enterprise

The business model used by GMBooks.com is a drop shipment pattern, where a retailer without inventory offers products from an aggregated catalog and routes customer orders to distributors or other firms who fulfill the orders from their own inventories.[4] While this coordination is usually invisible from the customer's perspective, it requires a complex and carefully managed series of document exchanges (often called a *document choreography)* over a period that may range from hours to weeks.

Independent business processes are coordinated by the exchange of information

Many dot-coms failed because their flashy websites could take orders from customers but did not implement the "back end" information exchanges with warehouses and shippers required to make reliable delivery promises to customers. Dissatisfied customers whose orders arrived late never ordered another product.

1.3 It's All About Exchanging Documents

The exchange of information between GMBooks.com and the other businesses that provide services to it takes place in the form of electronic messages or documents. In the book purchase scenario it is easy to identify the information that must be exchanged to carry out the desired business processes: an identifier for the book (perhaps an ISBN), a credit card number and purchase amount, and a customer's name and shipping address.

Some people might object to classifying these relatively small pieces of information as documents. They may want to distinguish between fine-grained, structured "data" and coarse-grained unstructured "documents" or use the latter term only where they can imagine something printed.

A chain of related documents will reuse common components

But more and more business processes involve both these ideas of "documents" and "data." A catalog might contain a mixture of description about products (text, graphics, photographs, and so on) and detailed data about their technical specifications. GMBooks.com depends on narrative documents like catalogs and book reviews as well as on transactional documents like orders, shipping notes, and payments to carry out its drop shipment business pattern. For information to flow efficiently from one type of document to another in this chain of related documents, there must be common content components that are reused. It isn't helpful to impose some arbitrary boundary between data and documents—what matters is the information components they convey.

1.3.1 The Document Type Spectrum

We view both documents and data on a continuum we call the Document Type Spectrum (see Figure 1-2) by analogy with the continuous rainbow formed by the visible light spectrum. It is easy to contrast highly narrative style documents from those that are highly transactionally oriented, just as it is easy to distinguish red from blue. But it can be difficult to distinguish different shades of a single color.

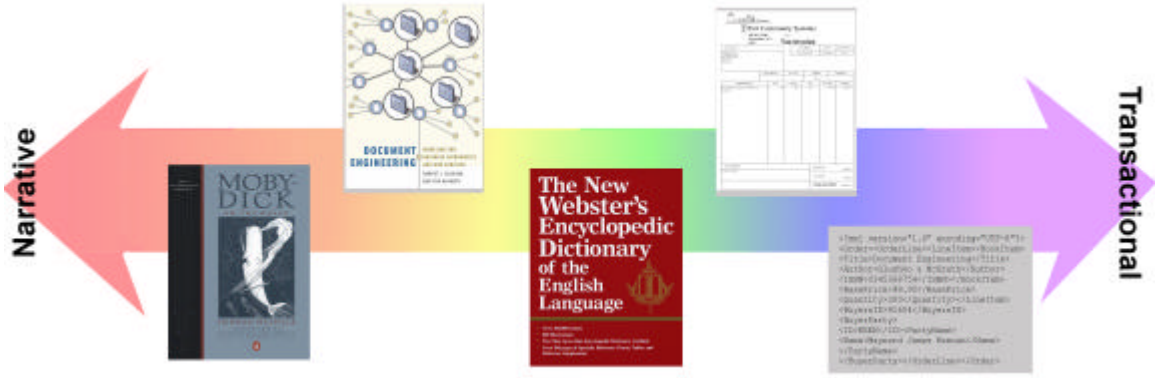

Figure 1-2. The Document Type Spectrum.

These difficult distinctions arise in the middle of the Document Type Spectrum where documents contain both narrative and transactional features. This is where we find hybrid documents like catalogs, encyclopedias, and requests for quotes.

The point is that this is a continuum. We don't see a magical point in the scale of information exchange from a short string of bits to complete purchase orders where data end and documents take over.

A document is a purposeful and self-contained collection of information

Nor do we see a sharp boundary between structured sets of data designed for use by computer applications and unstructured information designed for people. Defining *document* in a technology-neutral way as *a purposeful and self-contained collection of information* seems to cover both ends of the continuum.

1.3.2 Document Exchange as a Building Block in Business Model Patterns

Doing business by document exchange is natural and intuitive. Businesses use documents to organize their interactions with each other and to package the information needed to carry out a transaction or other meaningful unit of business. The seller may ask, "What do you want to order from my catalog?" and the buyer might ask, "Will you accept my purchase order?" The buyer and seller (or their lawyers) may negotiate a detailed contract with the precise terms and conditions of their business relationship. This contract often

specifies the content and timing of other documents that the parties are to exchange in the course of conducting their business with each other.

The exchanges of documents that take place to carry out business models follow common patterns. For example, the drop shipment pattern illustrated in Figure 1-1 is just one example of how a firm uses document exchanges with other firms to carry out its business model. Supply chains, business-to-business marketplaces, auctions, information brokers or aggregators, and content syndication networks are other examples of business processes that use document exchanges to combine or interconnect products or services from multiple businesses.

The document exchanges an organization uses to carry out its internal business processes also follow patterns. For example, the order management cycle can be described as ten steps that begin with order planning and end with post sales service. The complete cycle involves numerous documents that flow between sales, engineering, finance, logistics, customer service, and other divisions within the organization. The specific documents and divisions vary in different contexts, but the general pattern is ubiquitous.

There is no necessary relationship between these business process patterns and their organization within an business's management structure or their support in technology and systems. For example, the fact that a business conducts procurement processes to obtain goods and services does not imply that it has a procurement department or that it has an automated procurement system.

Of course, the business model may determine or at least constrain many decisions about how the enterprise is organized and what information technology it uses. The fit between what a business does and how it does it can be more easily assessed if the business model is defined in an abstract manner independently of its organizational and technological implementation. The foundation of the U.S. government's Federal Enterprise Architecture[5] effort is such a business reference model (BRM). Their goal is to support cross-agency collaboration, transformation, and government-wide improvement

by requiring that organizational structures and technology investments be aligned with the business model.

The alignment of business models and technology is easier to achieve when an organization systematically structures its business capabilities as self-contained resources or processes so they can efficiently interact and recombine to meet changing business requirements. Using standard documents as the interfaces for business processes is a natural outcome of organizing business functions as more discrete and flexible components.

GMBooks.com is a simple example of combining component business services to create a composite application. Real-world components like the Amazon.com catalog,[6] the UPS delivery and tracking functions, and Visa payment processing are all available as document-based web services for easy integration into other business systems.

In Chapter 4, "Describing What Businesses Do and How They Do It," we compare and contrast the organizational, business model, system architecture, business process, and information architecture perspectives from which one can analyze and describe models of document exchanges. This provides a repertoire of patterns of different granularity that can be reused to devise new business solutions.

1.3.3 Document Exchange as Loose Coupling

Document exchange as a mechanism for conducting business lets the participants focus on what they want to accomplish rather than how they must do it. The seller asks, "What do you want to order from my catalog?" and hopes not to have to ask, "What kind of software do you use to arrange and send electronic orders?" A relationship is called *loosely coupled* when the parties avoid dependencies, so that changes by one party have no impact on the other.

It is nonsensical to imagine a business relationship that depends on the color of the file cabinets in which documents are stored, the brand of accounting software used to calculate invoices, a database or directory structure, or anything else about the technology choices involving information or documents to be exchanged. A relationship with these

kinds of constraints would be too *tightly coupled* to cope with the ordinary evolution of business practices. But historically many approaches for integrating applications depend on screen layout, record or table structures, fine-grained application program interfaces (APIs), or other implementation details and are more tightly coupled than might be desirable.

Loose coupling is an old and familiar idea. Telephones and fax machines enable businesses anywhere in the world to exchange information with each other even if they have no existing relationship.

Loose coupling is an old and familiar idea

All they need to know is the other's phone or fax numbers, which they might find in a business directory. They don't need to know anything about the other's choice of telephone or fax equipment, and either business could buy a new phone system or fax machine and the other one wouldn't need to know and wouldn't care. Technical standards for how these devices connect to the phone network make them all look the same to the other side.

Of course, the problem with telephone messages and fax machines is that they don't make it easy for the recipient to extract the important information about the business activity in an automated way. That's why businesses (or different divisions within an organization) prefer to send computer-processable documents to each other—to improve accuracy and allow the information they exchange to grow in volume or frequency. And as with telephones and fax machines, standardization of these documents can be essential in maintaining a loosely coupled relationship.

What this all leads to is that two business organizations must agree on what the documents mean and on the business processes they expect each other to carry out with them, but they don't need to agree on or even know anything about the technology they use to create and process the documents.

Organizations must agree on what their documents mean

For example, suppose a customer sends a purchase order to GMBooks.com; if GMBooks.com can fulfill it, they respond with a purchase order acknowledgment, or perhaps with an invoice and a shipping note. As long as the customer and GMBooks.com understand each other's documents and can produce and respond with the documents appropriate for each other's business processes, they don't need to reveal how they produce the documents they send or how they process the documents they receive. The documents are the only visible interfaces to their respective business processes.

1.4 Understanding the Meaning of Documents

For most people understanding something they read in their native language is so immediate that they don't think much about it. The meaning seems to leap directly from the words on the page into their consciousness in a natural and automatic way. This is why we may feel surprised or confused if we later realize that other meanings or interpretations of the words were possible.

And we're not just talking about poetry or philosophy, where the author's intent or the inherent abstractness of the subject matter challenges readers to make sense of the words. Even in catalogs, forms, contracts, and other ordinary business documents, the relationship between words and meaning can be complex and subtle. For example, the same meaning can be described with different words (*Address* in one document might mean the same as *Location* in another one), or different concepts can be described with the same words (*Address* might mean the buyer's address in one document and the seller's address in another). The meaning of some words can change significantly in different business situations or contexts; consider that *Next Day Delivery* might mean delivery tomorrow but not if today is a weekend day or holiday because *Day* in *Next Day Delivery* means business day.

Situations like these can obviously cause misinterpretations, but people have a remarkable ability to refine or repair their understanding. However, computers and software have no such ability. We have all experienced system crashes and unexpected behavior when some bit of data was misinterpreted by application logic because it didn't

mean to the program what we thought it did (an infamous example occurred in 1999 when an interplanetary mission to Mars failed because one engineering team used metric units and another one didn't[7]).

So we need to be diligent and precise when we define the meaning (or semantics) of any information content produced and consumed by business applications. But this is easier said than done, and there is a great range in how diligent and precise we can be in doing so.

We need to be diligent and precise when we define semantics

At the very least we can try to define words to create a dictionary. At the other extreme our definitions are expressed in a formal language using a controlled set of terms and relationship types between them.

In the ideal world we end up with a complete view of how information is defined and used in different business contexts—what is often called the *information model*—a formal representation of the structure and semantics of information. Of course, people often aren't as careful or conscientious as they should be in creating information models. They may fail to recognize the seriousness of the semantic ambiguity problem, or they may have insufficient time, expertise, resources, or incentives to attack it. In either case, there can be substantial differences in the meaning and presentation of information within a single enterprise. And this is invariably reflected in any documents they create.

1.4.1 Incompatible Information Models

In large organizations it is easy to find numerous varieties of timesheets, expense reports, purchase orders, catalogs, calendars, and other types of documents. These are likely to contain incompatible information models that prevent time, expenses, and purchases from being aggregated or compared.

In Chapter 6, "When Models Don't Match: The Interoperability Challenge" we further describe the problem of multiple interpretations or formats for what is supposed to be the same information.

Even if each enterprise in a business relationship were disciplined in its own approach to modeling and describing the information it uses internally, that wouldn't be sufficient. There are at least two sides to every document exchange, and all parties need to ensure that they understand the documents in the same way.

One way to do this would be for every enterprise to adopt a common data model and use exactly the same definitions for the document components of their applications. But that's inconceivable; enterprises, applications, and people just don't stand still long enough to make it possible. It is more conceivable that two parties might each create conceptual information models to help them translate or transform the documents they receive so their applications can understand them. The Data and Information Reference Model being created as part of the U.S. government's Federal Enterprise Architecture is an exemplary and ambitious step in this direction.

Starting in Chapter 8, we'll present a case study about the development of a standard model of an event calendar to replace dozens of incompatible ones used at the University of California, Berkeley.

1.4.2 Standard Information Models

Both the common data model and conceptual information model approaches for ensuring that parties understand each other's documents are facilitated when the syntax, structure or semantics conform to common patterns or standards. Many of these have been developed for specific vertical industries by trade associations, industry consortia, or formal standards bodies.[8] Standards efforts are often the most successful where the stakes are the highest, and it isn't surprising that standards compliance is highest for business processes like payment initiation, reconciliation, funds transfers, and statutory reporting.

Standards for information components needed in all businesses are a more recent development. For example, descriptions of organizations and individuals, basic item details, measurements, date and time, location, country codes, currencies, business classification codes, and similar reusable patterns of information components are found in a wide variety of documents.

Standards for syntax, structure, and semantics facilitate document understanding

Standard reusable patterns are especially important when designing the set of documents needed to carry out a composite business process; because they encourage the assembly of documents from building blocks that are reused as information flows from one document into the next. In this regard, the Universal Business Language (UBL) effort, released in mid-2004 promises to be an extremely important standard.[9]

1.5 XML as an Enabling Technology

To exchange documents, computers or business applications require a precise and unambiguous language for describing information models. Since its emergence in the late 1990s, XML—the Extensible Markup Language—has rapidly become the preferred format for representing information in documents both on and off the Internet.

People who work in web publishing view XML as an improvement on HTML; that enables greater automation and consistency in formatting.

Programmers see XML as an Internet-friendly, easy-to-parse, and nonproprietary data format that they can use instead of ad hoc languages for application configuration and interprocess communication.

Electronic data interchange (EDI) developers see XML as a more expressive, maintainable, and therefore lower cost syntax for creating business messages.

XML's broad impact in publishing, programming, and EDI has made it a unifying technology for implementing applications that use Internet protocols, especially for those that span enterprise boundaries, such as web services.

XML has become the preferred language for representing information in documents

Expressing information content and processing logic in a computer-friendly XML vocabulary enables robust applications to be deployed efficiently and at a reasonable cost. XML content can be taken from documents, databases, and enterprise applications; combined and treated as a single source; and delivered to multiple users, devices, or applications.

In Chapter 2, "XML Foundations," we introduce XML from the perspective of modeling and document exchange. We emphasize the conceptual innovations in XML and don't dwell on XML syntax or schema languages. There are plenty of excellent books about the latter, and these aren't what is most important about XML anyway.

1.6 Using XML-Encoded Models to Design and Drive Applications

The expressiveness and flexibility with which XML encodes models makes it a powerful technology for improving software engineering. Applications that are built using models to bridge the traditional gap between design and implementation are often called *model-based applications* or model-driven applications.[10] They share information and can be integrated with others more readily. And they are vastly easier to deploy and maintain than those developed without explicit models or for which the models were left behind when coding began.

Some developers still ignore the well-known benefits of a disciplined software development methodology with controlled iteration of analysis, design, implementation, and user feedback and still employ labor intensive and ad hoc techniques that do not predictably yield quality software. The reasons (or excuses) for not following a software engineering approach are well-known: unrelenting user demands for new or revised functionality, competitive pressure for rapid application deployment and modification, or simply the difficulty of obtaining correct requirements without coding something and testing it. Unfortunately, the results are also well-known: little reuse of information or processes across applications, applications that are coupled in unpredictable ways by shared data, and business rules and workflow specifications embedded into application logic.

It doesn't have to be this way. Data dictionaries, programming language classes, database schemas, UML models, spreadsheet templates, and XML schemas can represent the rules and semantics for the documents and processes needed by software applications. These different ways of expressing models are designed for different purposes, but what

is important is that each of these representations can be used in a rigorous and formal way to define models that can then be used as specifications for generating code or configuring applications.

In this book we emphasize the use of XML-encoded implementation models to design and drive applications. But, there is certainly nothing about models of documents or processes that requires them to be represented in XML.

Nevertheless, using XML to encode implementation models yields an overall rigor, reusability, and programmability unmatched by other representations. Furthermore, XML's facility for document encoding is an excellent match for the document exchange architecture of the Internet. For those who prefer other representations of data models, programming paradigms are emerging in which XML schemas, programming language objects, database schemas, and UML models can be treated as equivalent because XML schemas can be used to generate any of the other formats if required.[11]

In Chapter 15, "Implementing Models in Applications," we discuss the use of XML for document and process implementation models as explicit representations of application requirements.

1.6.1 Document Models as Interfaces

The simplest case of model-based applications is also the most common. For countless information-based activities that have moved to the web, the application is little more than a document displayed in a browser that users interact with in ways that are determined by the document's model.

On the narrative end of the Document Type Spectrum are E-books or other structured publications in which user interface features like tables of contents, hypertext links, and navigation aids are generated from the names or attributes of the information components in the document. On the transactional end of the Document Type Spectrum are E-forms in which applications collect the information specified in the document's model to automate processes that previously have relied on printed forms. We can readily imagine applications where information moves within and between companies—filling out

purchase orders, submitting a budget or timesheet, seeking reimbursement for expenses, applying for a grant or job, registering for classes or events, filing income taxes, making insurance claims—the list is endless.

In our online bookstore example, a customer's order from an online catalog might be captured using a web-based E-form.[12] Some pieces of this information, such as the customer's name and address or the title of the book being purchased may be required in other applications, such as those dealing with supply, delivery, or billing, as illustrated in Figure 1-3.

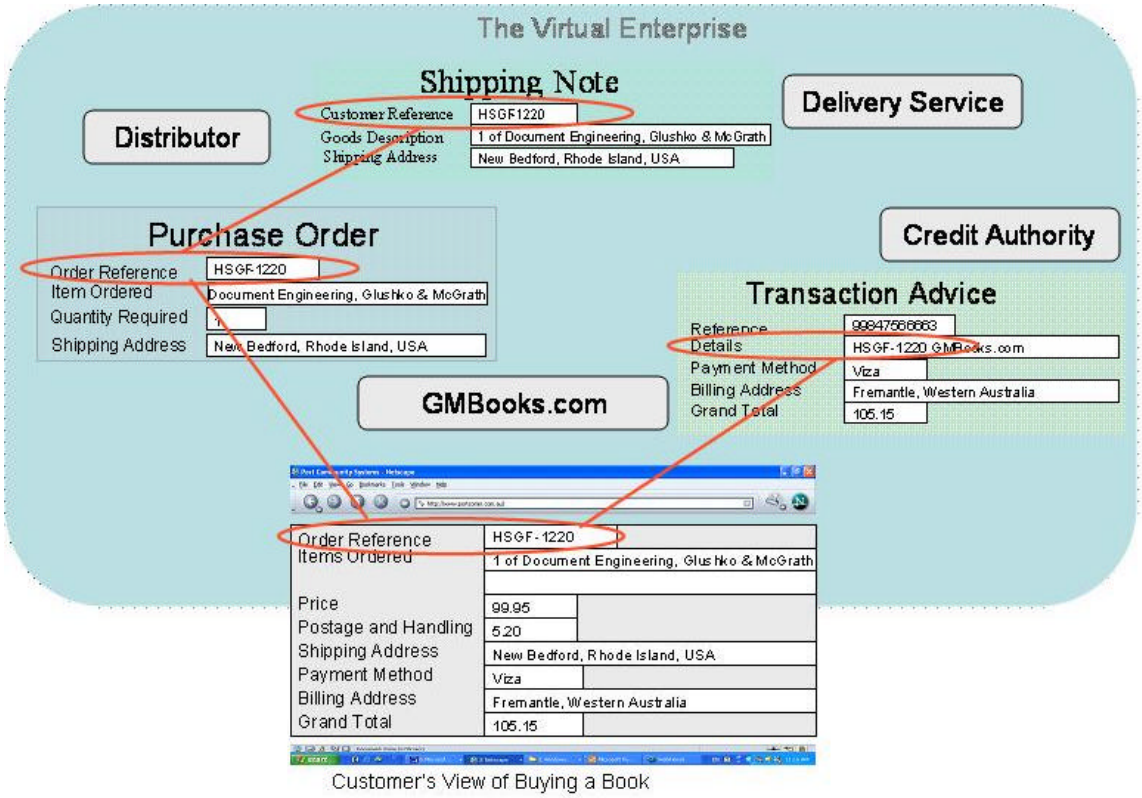

Figure 1-3. Overlapping Content Models in the Virtual Enterprise

These shared pieces of information form the glue that binds the different services to create the virtual enterprise that the customer sees as GMBooks.com. The benefits of a common model for these shared pieces of information are obvious: no data reentry and no omissions or misinterpretations. And this is a trivial example. Consider the benefits of

reuse in the average cross-border trading process, which can involve up to 40 documents and 200 data elements, 30 of which are occur in at least 30 documents.[13]

While many applications begin as user interactions with a form, the business processes that follow might be carried out by computer programs with no human involvement. But it makes sense for us to look at both kinds of model based interactions in the same way, generalizing the idea of documents as interfaces for people to the idea that documents are interfaces to business services or business processes.

Documents describe the interfaces to business processes

In both cases the document conveys or captures information in an exchange with another party or process without necessarily revealing anything about how the information is consumed or created by each participant in the exchange.

1.6.2 Models of Business Processes

Figure 1-3 illustrates the reuse of information between numerous documents that together carry out a business process. But the mere reuse of information from one document to another isn't sufficient to make the business process work. It is also essential for all parties in a document exchange to agree on the purpose or context of a document, which means understanding the business process in which the document exchange is taking place.

Suppose GMBooks.com sends a list of books to another business. This same list of books might appear in an order, an order response, an order change, a price and inventory check, a shipment notice, a bill of lading, an invoice, and so on.

All parties in a document exchange must agree on the context of use

But nothing in the list of books itself communicates anything about the purpose, intent, or business context with which the list should be interpreted by the organization that receives it.

If GMBooks.com sends a purchase order to a book distributor and expects a purchase order acknowledgment in return, what happens if the book distributor's normal business practice (when it can fulfill an order) is to only send an invoice and a shipping notice?

If GMBooks.com's applications are incapable of handling such an electronic response from the book distributor, the process breaks down.

In such situations, it is unlikely to be trivial for either GMBooks.com to modify its order management system to dispense with its acknowledgment and accept the document that the book distributor sends. Nor could we expect the book distributor to modify its systems to produce what GMBooks.com wants. But one or the other must do this to enable an automated business exchange.

Clearly, the rules of the business process that control the pattern of document exchanges or, more generally, defines the agreement or mutual understanding of the parties to the exchange, should be expressed in an explicit model.

For example, GMBooks.com could accompany the order with a *business process model* that defines both the documents it expects to exchange when it sends orders and the sequence of their document exchanges. This model can be defined as another document that is used to configure the software on the distributor's side, or it might even be executable and used at run time to ensure that the appropriate documents are exchanged and processed in the specified sequence.

We may not be quite there yet but many initiatives are working diligently to get us closer by developing standards for specifying processes, their composition, and their coordination. Furthermore, many software vendors are developing middleware for using the specifications to control or verify document exchanges.

How these problems of business process integration are resolved depends on what causes them. Technology mismatches are a significant factor. Also significant are the existence of industry standards or reference models, the relative power in the business relationship, the technological and process maturity of each firm, and the extent to which the firms have complementary long-term business strategies. In Chapter 16,

"Management and Strategy in Document Engineering" we look at broader factors that determine the success or failure of efforts to exchange information within and between enterprises.

1.6.3 Web Services and Service Architectures

Using documents as interfaces and thereby hiding implementation details underlies the idea of service -oriented architectures (SOA) as a way to create new applications or systems such as *web services* by integrating or combining components of other ones.

A technical definition of a web service is "an interface that describes a collection of operations that are network accessible through standardized XML messaging."[14] This means that any self-contained application functionality or information resource is turned into a service by packaging it so that it exposes only input and output XML documents. Typically, these are transported over the Internet.

But this definition, though entirely comprehensive, doesn't explain why there is so much hype about web services and SOAs. One of the most common senses of *service* contrasts it with *products* to mean some kind of activity performed by a person; so if you aren't a programmer you might not realize from the definition that almost anything can be turned into a service. Because of the abstraction level introduced by document exchange, a service can be

- Anything that can send or receive a document
- Anything that can accept a document, process it and return a result
- Anything that can accept a document and allow the user to act on it
- Anything that can accept a document and forward it to some other application or destination
- Anything that can generate a document as a result of user interaction, processing a received document, or some other event.[15]

So a service can be anything and do anything, as long as the information needed to request it and the work or results that it produces can be effectively described using XML. Note also that the way we've defined services allows them to be provided by

people as well as by software or other automated means, and the document interface by itself provides no hints.[16]

Anything that takes requests and describes its results using XML can be a web service

No small set of examples can convey the range of possible services, but here are some anyway: stock quotes, tax rates, inventory levels, order tracking, payment processing, restaurant reservations, traffic conditions, sports statistics, credit ratings, algebraic expression evaluation, and language translation.[17]

While a service can carry out some useful business activities on its own, if its document interfaces are described in standard ways, it can combine with other services to create a composite application that provides additional value because of the combination and is more efficient in invocation. For example, consider how a travel information service could be created by combining separate services that provide personalized information about local news, weather, cultural events, traffic conditions, hotels, restaurants, and so on. You could request all of this interrelated information with little more than the name of the destination city, and it would be assembled invisibly by the composite service.

Adopting a flexible SOA benefits a firm in many ways. The core idea that applications should be built by assembling service components rather than repeatedly coding them, promises lower cost and a more general approach for integrating or reusing separate systems or resources within an enterprise. Duplicated functions can be consolidated; for example, in a large enterprise a single service for processing payments might replace dozens of existing applications.

SOAs also enable web services to expose inward-facing legacy systems and data repositories to external businesses or customers and thereby add value to business relationships. For example, a web service that looks up customer details in a customer database can be combined with one that knows about orders in a ERP system, creating a composite service that locates the current orders when a customer calls in.

Because web services are loosely coupled and hide implementations, document interfaces allow firms to maintain a clean and stable relationship to partners and customers. Even if an organization subsequently migrates its internal processes and data from legacy systems, users of the web service shouldn't notice.

Document interfaces maintain clean and stable relationships between business partners

By using independent components, web services also make it easier and cheaper to adopt new technologies incrementally without affecting any existing business functionality. Implementation transparency supports more objective "build vs. buy" decisions about business services and permits comparisons among alternative providers. Because document interfaces can be implemented in any technology, platform compatibility concerns are lessened. And since businesses can have nonessential processes performed by another firm without being locked into a relationship with that provider, traditional arguments for running business applications internally might now lose to those for outsourcing. But as we shall see in Chapter 4 "Describing What Businesses Do and How They Do It", such decisions may not be that simple.

Finally, as more third-party service providers adopt document interfaces for hosted services, enterprises can more quickly react to changes in business conditions and customer demand by treating software resources like utilities, using only as much as needed. The flexibility, extensibility, and responsiveness made possible by web services and service oriented business architectures are becoming central to the marketing and branding of platform vendors, consultants, and professional services firms.[18]

However, this is *not* a book about web services, and while we will discuss them briefly in Section 4.4, "Views of Business Architecture," we will not go into any more detail about web services specifications or technologies. Once again, there are many excellent sources that do that.[19] Instead we focus on how to understand the business context for web services and on the conceptual tasks of analyzing and designing the documents and processes that might ultimately be implemented in service oriented architectures.

1.7 Document Specifications and Document Engineering

Where do the specifications for the documents needed by the new business models come from? We propose that they should come from Document Engineering—a new discipline for specifying, designing, and implementing the documents that serve as the interfaces to business processes.

We do not mean to imply that every document or process model needs to be created from scratch—far from it. Just as with every other engineering discipline, Document Engineering emphasizes the reuse of existing specifications, standards, or patterns that work, reducing costs and risks while increasing reliability and interoperability. Useful patterns for Document Engineering include those encoded at the implementation level in the form of XML schema libraries or EDI message standards. Others are at more conceptual levels, in the form of industry reference models for common business processes, or even in more abstract patterns for the organization of activities between businesses such as supply chains, marketplaces, or straight-through processing.

Of course, no existing pattern is likely to be totally suited to the required context of use. So a business must follow an engineering approach that develops models that meet these requirements. And even then it is essential to design and implement the models in a manner that enables its subsequent modification and reuse. In Chapter 7, “The Document Engineering Approach,” we introduce Document Engineering as an artifact-focused view of the classical requirements-analyze-design-refine-implement methodology.

1.8 Document Engineering—a New and Synthetic Discipline

The analysis and design methods of Document Engineering have their roots in other fields, primarily information and systems analysis, electronic publishing, business process analysis and business informatics, and user-centered design. Each of these disciplines looks at documents and processes differently, and while each of them is highly effective in some areas, they all have blind spots where their methods and techniques do not work well.

Many people have contrasted narrative types of documents that mostly contain text with transactional types that mostly contain data, but they typically conclude that *documents* and *data* cannot be understood with the same terminology, techniques, and tools. For example, with narrative documents, such as those that are traditionally called *publications* and intended for use by people, analysis and modeling techniques are usually described as *document analysis*.

In contrast, transactional documents are optimized for use by business applications and differ in other substantial ways from traditional user-oriented publications. The analysis and design methods used for transactional documents are often described as *data analysis* or *object analysis*.

Task analysis and related techniques for user-centered design overlap with document and data analysis to identify the intent and information requirements for the tasks people perform.

Finally, while *business process analysis* can be conducted in domains that involve either or both narrative and transactional document types to set the context for document or data analysis, analyzing the content of the documents required is not its primary goal.

Document Engineering synthesizes the complementary ideas from these separate fields, emphasizing what they have in common and applying it with a unified focus to a broad range of documents and processes.

Document Engineering synthesizes complementary ideas from separate disciplines

This synthesis is essential because narrative and transactional documents are often closely related, either by structural transformation or by business processes. Consider, for example, the close relationship between tax forms and the instructions for filling them out, or between product brochures and purchase orders.

1.8.1 Business Process Analysis

When an organization wants to improve its effectiveness and efficiency, it often conducts a business process analysis (or reengineering) to acquire a better understanding of what it

does and how it does it. Often the goal is to assess business capabilities or competencies and identify processes that enable strategic opportunities or pose strategic risks for the organization.

Business analysis is also required when two organizations consider joint ventures, partnerships, or other strategic relationships that involve doing strategic business with each other. This analysis can determine the compatibility of business processes, customer and supplier relationships, accounting practices, and the day-to-day processes that define the corporate culture of each organization.

Whether within an organization or between them, business analysis usually begins with an abstract or broadly defined perspective on business activities and works from the top down through a hierarchy of business reference models, business processes, collaborations, and transactions. Because the usual goal is increased understanding of how things work from a business perspective, the process analysis often stops at the transactional level where document exchanges are visible. Often no one pays any attention to the design of the documents, their implementation, or the technical capabilities they require to design, develop, and deploy. The analysts may assume that the technology exists or will be created to implement the business decisions that emerge from the high-level process analysis. In any case, it's someone else's problem.

1.8.2 Task Analysis

Task analysis (or user analysis) is the observation of users performing the tasks or *use cases* when the application or system must support people and not just other applications. Task analysis identifies the specific steps and information that users need to carry out a task. Task analysis is especially important when few documents or information sources exist because user problems or errors can suggest that important information is missing.

1.8.3 Document Analysis

In contrast to the top-down approach of business and task analysis, document analysis is inherently a more bottom-up activity. This is especially true when the motivation for analyzing documents is the narrow goal of transforming existing printed documents or

business forms into electronic versions, a process known as document automation. Indeed, when the business driver is a mandate to automate the exchange of documents with a dominant business partner, as Wal-Mart has done with its major suppliers,[20] the paramount goal may be to take an existing manual process and encode it in documents according to process specifications imposed by the partner. Any process or task analysis in this one-sided situation can be viewed as needing little attention or, in the extreme case, as being irrelevant.

A more typical business motivation for document analysis is an enterprise's desire to become more efficient and effective at managing and distributing its documents. A common goal is single-source publishing in which content is managed as reusable information components and assembled as needed in different types of documents or output formats. For example, the same content might appear in product catalogs, printed installation and repair manuals, a CD-ROM E-book, and web pages. Process analysis is important in this situation but still secondary to the need to analyze the existing and potential documents very carefully.

Document analysis emphasizes the study of narrative style documents as artifacts because of the complex ways in which they merge presentation with structural and content components. Making sense of this complexity requires a wide range of document analysis techniques developed by publishing, text processing, information architecture, and graphic design experts. Document analysis is typically carried out with the goal of separating a specification of a document's content and structure from its presentational characteristics such as fonts, type sizes, and formatting used to represent or highlight various structural or content distinctions.

Once this separation is accomplished, a model of the document is created, usually called a schema. The optimal prescriptive schema for a set of documents is one that best satisfies the requirements of current and prospective users for carrying out specific tasks with new instances.

Finally, one or more stylesheets can be used to assign formatting or rendering characteristics in a consistent manner to any document that conforms to this schema.

1.8.4 Data Analysis

Data analysis has its roots in philosophy and linguistics, but in its current incarnation is a set of techniques used for designing database systems. It is primarily devoted to understanding and describing the properties and relationships between information components or objects.[21] The typical goal of the data analyst is to define conceptual models that organize these components efficiently to support a broad range of contexts or applications. Because their information is often stored as large structured sets of data in databases, data analysis is a key step in database design.

Data analysis methods, like those of document analysis, are bottom-up in the sense that they are applied to existing artifacts. But in contrast to the heterogeneous narrative artifacts for which document analysis techniques are best suited, the more transactional artifacts to which data analysis methods apply best are homogeneous. Transactional documents usually exist as a limitless number of almost identical instances, often produced mechanically to represent some state of an activity or business process. Such documents are extremely regular in their structures and have strongly defined content components, but provide minimal or arbitrary presentation features.

The regularity of transactional information has enabled the development of more formal techniques for modeling its use in information systems. These techniques progressively refine and abstract information models by identifying repeating or recurring structures, removing redundancies and technology constraints, and otherwise creating a more concise and reusable representation of the information components.

1.8.5 Unification in Document Engineering

We acknowledge that document analysis, data analysis, task analysis, and business analysis come from different intellectual traditions. In addition, the practitioners of these approaches often come from different educational backgrounds, may have little professional communication with each other, and can fail to recognize the overlap in their

goals and methods. We cannot, however, just shrug our shoulders and treat documents, data, processes, and user interfaces as separate universes.

For example, the services in a service -oriented architecture involve both documents and processes, and their information invariably flows between narrative documents and transactional ones. To make these services work, the businesses or business units involved must implicitly or explicitly reach a common understanding about how their processes should be designed, how they can be deconstructed into service components, the documents and information they exchange, the timing of the exchanges, and the people, organizations, or roles involved. This common understanding must be represented in models of the required documents and processes that are comparable in abstraction and satisfy the requirements for their context of use. This can happen only if document, data, task, and business process analyses can be brought together in a unified approach.

To achieve this common understanding Document Engineering proposes a document-centric reformulation of traditional data analysis. But we recast its formal and specialized methods to apply equally to narrative style documents. At the same time it takes the best practices of document analysis and applies them to understanding information components. Finally it adapts task and business process analysis techniques to identify the requirements and business rules of their context of use.

This synthesis achieves the composite goal of all four analysis methods

This synthesis achieves the composite goal of all four analysis methods—creating formal specifications of information components and classes of documents that contain them, satisfying both the business processes in which they participate and the people who create and use them.

Viewing narrative and transactional types of documents as different points on a continuous Document Type Spectrum (see Figure 1-2) is a fundamental part of the new Document Engineering approach. Likewise, it is essential to make the top-down activities

of business process and task analysis meet in the middle with the bottom-up efforts of document and data analysis.

1.9 The Document Engineering Approach

Commercial firms, governments, universities, and other types of organizations have different goals and conduct different kinds of Document Engineering projects. But before they undertake any project, each must make a business case that identifies its objectives and the likely return on investment. These management and strategy decisions shape the project's goals and permeate most of its activities, and we could properly view them as the first phase of Document Engineering. But it is difficult to discuss the overarching perspectives of *what* to do and *whether* to do it before explaining *how* to do it, so we'll defer these concerns until Chapter 16, "Management and Strategy in Document Engineering" and not treat making these decisions as a separate phase.

Document Engineering organizes its modeling approach into eight phases as shown in Figure 1-4.

We briefly outline the approach here and will explain it in detail starting in Chapter 7.

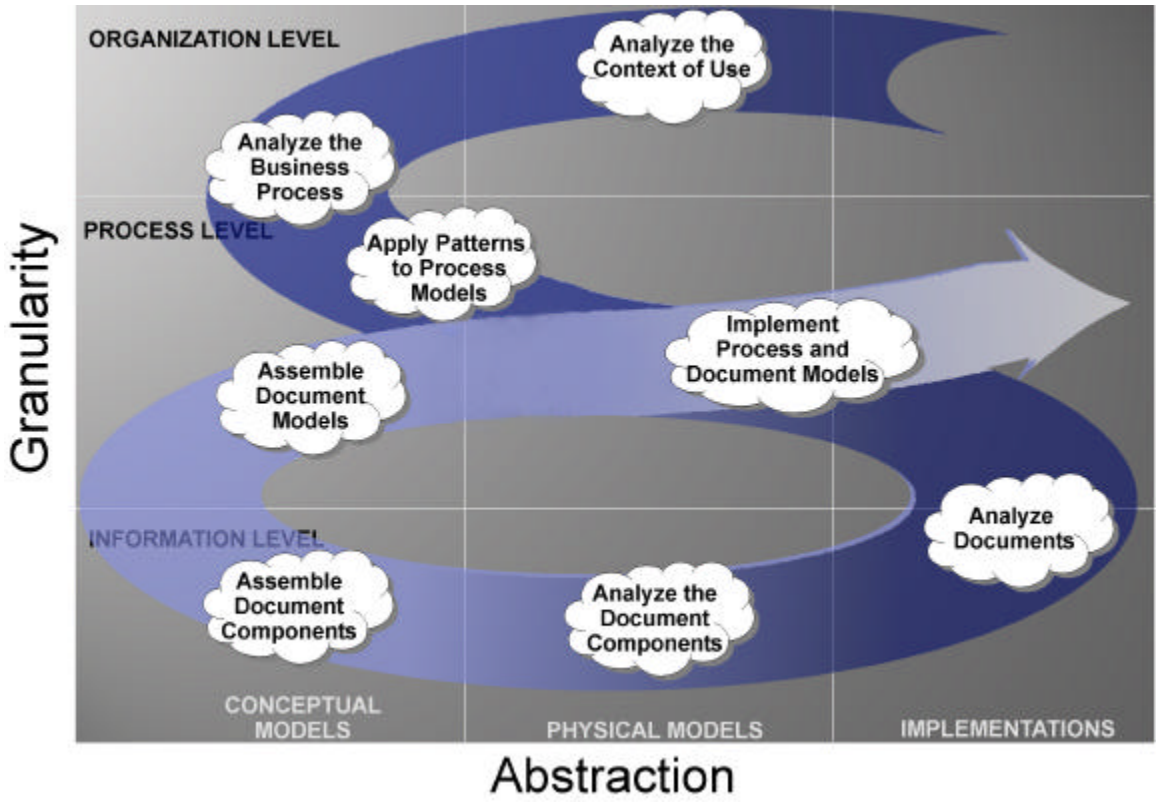

Figure 1-4. The Document Engineering Approach

In the first phase, **Analyzing the Context of Use**, business and task analysis techniques establish the context for a Document Engineering effort by identifying the requirements and rules that must be satisfied to provide an acceptable solution. Chapter 8, "Analyzing the Context of Use," explains this phase in detail.

In the next two phases—**Analyzing Business Processes and Apply Patterns**— we apply business process analysis to identify the requirements for the document exchange patterns needed to carry out the desired processes, collaborations, and transactions in the context of use. These patterns identify documents that are needed, but only generally as the payload of the transactions. The complete requirements for the documents can't be determined without analyzing existing documents and other information sources. Chapters 9, "Analyzing Business Processes" and Chapter 10, "Designing Business Processes with Patterns" describe this phase.

The next phase—**Document Analysis**—involves identifying a representative set of documents or information sources (including people) and analyze them to harvest all the meaningful information components and business rules. Chapter 11, “Analyzing Documents,” and Chapter 12, “Analyzing Document Components,” present the activities in this phase.

In the **Component Assembly** phase we develop a document component model that represents structures and their associations and content that define the common rules for the possible contexts of use. Chapter 13, “Assembling Document Components” presents the steps of this phase.

We then move from analysis tasks to designing new document models. In the **Document Assembly** phase, we use the document component model to create document assembly models for each type of document required. If possible we reuse common or standard patterns to make the documents more general and robust. Chapter 14, “Assembling Document Models” presents this phase.

The new conceptual models we have created for processes and documents can be viewed as specifications for interfaces, for generating code, or configuring an application that creates or exchanges new documents. These models represent substantial investments in understanding a context and capturing its requirements in a rigorous way. Using these models to implement a solution in an automated or semiautomated manner exploits those investments to bridge the gap between knowing what to do and actually doing it.

In the **Implementation** phase these conceptual models are first encoded using a suitable language to support their physical implementation. This is most likely to be XML, but because of the technology-neutrality of our approach, the models can be implemented in languages such as UN/EDIFACT or ASN.1 if required.

Chapter 15, “ Implementing Models in Applications,” begins with a discussion of encoding document and process models and then reviews the issues that arise when applications are based on these models.

SIDEBAR: Why We Call It Document Engineering

Document engineering may seem a novel formulation, but we couldn't think of a more appropriate combination of terms to describe what this book is about. We want to highlight the creation of tangible end products with economic or social value (that is, documents), and we believe that process is more strongly implied by *engineering* than any other word.

The closest existing discipline to what we are defining is probably business informatics, which seeks to "combine the modern theory, methods and techniques of business (i.e. organization science) and informatics (i.e. information and computing science) into one integrative programme."[22] This definition certainly covers many of our goals, but it doesn't emphasize the need for conceptual modeling of the documents and processes at a granularity that is implementable, which we believe is fundamental.

In addition, while business informatics seems to have a foothold in Europe and Australia, the phrase is almost invisible in the United States (somewhat surprising given the relative familiarity there of "bioinformatics" and "medical informatics" as names for disciplines and academic departments). An exception is the Business Informatics organization at IBM's Watson Research Center headed by Dr. David Cohn, who wrote the foreword to this book.

A lawyer might say, "I'm a document engineer. I create the documents that govern business relationships, ensuring that the document handles my client's needs while getting agreement from the other side so that there are no surprises later."

A technical writer or information architect might say, "I'm a document engineer. I design documents so that they contain the information my intended audience needs. I follow company and industry practices for content, structure, and presentation to convey the information in an optimal way."

A programmer might say "I'm a software engineer, and in addition to designing programs I design the data structures, objects, or messages that convey or exchange information from one process or program to another. So I'm a document engineer, too."

Our definition of *document engineering* is mostly consistent with those of the lawyer and the technical writer, with some modest differences. Unlike the documents they create, the documents we want to engineer are likely to be used more often by an application or web service than by a person. And until relatively recently, most programmers were vastly more familiar with fine-grained and tightly coupled application program interfaces than with the coarse-grained, loosely-coupled document exchanges.

But all of us share the goals of conveying the right information in a mutually intelligible and standardized fashion, and we follow a disciplined approach to ensure that the documents are useful and reliable.

Nevertheless, the combination of "*document*" and "*engineering*" remains surprisingly novel. If you google the separate words "*software*" and "*document*", they have roughly the same number of results, but the results when you search for "*software engineering*" are orders of magnitude higher than those for "*document engineering*". That's good. We have a nearly blank slate on which to write, and we're confident that over time our new phrase will start to catch up.

1.10 Key Points in Chapter 1

- Doing business by document exchange is natural and intuitive.
- We define document in a technology-neutral way as a purposeful and self-contained collection of information.
- We generalize the idea of documents as interfaces for people, to the idea that documents are interfaces to business processes.
- A virtual company created by using services provided by separate businesses can be created more rapidly, more flexibly, and at a lower cost than a traditional one can.
- When businesses exchange documents, they must agree on what the documents mean and on the business processes they expect each other to carry out with them, but they don't need to agree on the technology they use.

- We need to be diligent and precise when we define the meaning of any information produced and consumed by business applications.
- We emphasize XML because it has become the preferred format for representing information in documents but many other representations can be used to define models.
- Document Engineering synthesizes complementary ideas from the separate fields of business process analysis, task analysis, document analysis and data analysis.
- The essence of Document Engineering is analysis and design methods that yield precise specifications for the information and rules that business processes require.

1.11 Notes

1. Israel Eph'al and Joseph Naveh, *Aramaic Ostraca of the Fourth Century B.C.* (Magnes, 1996). An ostracon is a fragment of pottery. Halfat's tax receipt is Ostracon #13 in Eph'al & Naveh's book. Such ancient tax receipts are commonly found in Egypt and elsewhere.
2. This validation might have taken place in the “bricks and mortar” bookstore, too, but it is essential in the online store because more stringent regulations apply whenever the merchant doesn’t see the actual card. But an even more important point here is that what looks like a single information exchange from either bookstore's perspective is in fact several transactions between the merchant, the merchant’s bank, the authorization network, and the bank that issued the customer’s credit card.
3. Robert Glushko, Jay Tenenbaum, and Bart Meltzer, “An XML framework for agent-based e-commerce,” *Communications of the ACM*, 42 (1999): 106-115. This paper introduced the idea that XML documents could serve as interfaces to business services that could be combined to form virtual companies. It significantly predates the "web services" announcements.
4. The website at http://www.the-drop-ship-guide.com is a useful guide, written from the perspective that you want to set up an Internet store that uses drop shipment (last visited 18 January 2004).
5. The Federal Enterprise Architecture Program Management Office (FEAPMO) publishes all of its documents at http://www.feapmo.gov (last visited 18 January 2004) .

6. Nick Wingfield, "In latest strategy shift Amazon is offering a home to retailers," *Wall Street Journal*, 24 September 2003. See also Amazon Web Services at http://www.amazon.com/gp/browse.html/104-5602037-8299135?node=3435361 (last visited 19 January 2004).
7. NASA, *Mars Climate Orbiter Mishap Investigation Board Phase I Report* (10 November 1999): 16. "The MCO Investigation Board has determined that the root cause for the loss of the MCO spacecraft was the failure to use metric units…Specifically, thruster performance data in English units instead of metric units was used in the software application code." ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf (last visited 3 September 2004).
8. The most comprehensive listing of relevant standards is the XML Cover Pages list of "XML Applications and Initiatives" at http://xml.coverpages.org/xmlApplications.html (last visited 19 January 2004).
9. The home page for the UBL initiative is at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl (last visited 19 January 2004).
10. A wide range of application development approaches are called *model-based* or *model-driven*, but the latter term is strongly associated with specific technologies and methods proposed by in the Object Management Group's MDA initiative (http://www.omg.org/mda/), so we prefer to use the former term in a more generic sense.
11. We are certainly aware that the Object Management Group's MDA considers UML models the normative representations, playing the same role as XML schemas do for us. For a UML-centric approach, see David Carlson, *Modeling XML Applications with UML* (Addison-Wesley, 2001) or Anneke Kleppe, Jos Warmer, and Wim Bast, *MDA Explained* (Addison-Wesley, 2003).
12. We use *web form* generically here to include HTML forms, which have long been the primary user interface for fill-in-the-form services, as well as those based on the W3C XFORMS recommendation or implemented using Adobe Reader, Microsoft InfoPath, or similar software for rich client applications.
13. Australian Department of Foreign Affairs and Trade. *Paperless Trading: Benefits to APEC*. (2001). http://www.dfat.gov.au/publications/paperless/paperless_trading.pdf (last visited 28 December 2004). The most common paper documents required for international trade are: Insurance Certificate, Certificate of Origin, Letter of Credit, Bill of Lading, Waybill, Manifest, Declarations, Sanitary and Phytosanitary Certificates, Payment Order, Remittance Advice, Debit Advice, Customs Clearance, Purchase Order, Invoice, Forwarding Instruction, Stowage Plan/Bay Plan, and Arrival Notice Advice.
14. Heather Kreger, "Web services conceptual architecture," IBM, http://www-4.ibm.com/software/solutions/webservices/pdf/WSCA.pdf (last visited 23 September 2004).

15. Bart Meltzer, "XML and the network economy," keynote address at CommerceNet Japan XML Conference, Tokyo, June 1998.
16. Although you might conclude from its limited availability, slow response latency, or other similar quality of service measures that a service was being provided by a person rather than by a computer.
17. There are numerous directories of web services. See, for example, the Microsoft UDDI service registry at http://uddi.microsoft.com/ (last visited 3 September 2004) or http://www.xmethods.com/ (last visited 3 September 2004).
18. To IBM, this is *business on demand;* to HP, it is *adaptive;* to Microsoft, it is *agile. Real-time* or *event-driven* are related terms that aren't as clearly tied to particular companies.
19. The SOAP and WSDL Web Services specifications are maintained by the W3C at http://www.w3.org/TR/soap/ and http://www.w3.org/TR/wsdl; the UDDI specification is at http://www.uddi.org/; and others maintained by the Web Services Interoperability Organization are at http://www.ws-i.org/ (all last visited 3 September 2004). O'Reilly publishes a wide range of books on web services, which are listed at http://webservices.oreilly.com/ (last visited 3 September 2004).
20. Alorie Gilbert "Wal-Mart project boon for software makers," C/NET News.com, 14 August 2003. http://news.com.com/2100-1017_3-5064075.html (last visited 10 June 2004).
21. Data analysis is related to object analysis. Object analysis techniques focus on the methods that will operate on or manipulate the information embodied in the object.
22. Utrecht University, Master's Programme in Business Informatics, http://businessinformatics.nl/index.php?id=1&subid=0