Document Engineering

Robert J. Glushko

This short paper was prepared for the Symposium on “The Coevolution of Technology-Business Innovations” held at IBM Almaden Research Center on September 24-25, 2003. This paper is a lightly edited excerpt from a book titled “Document Engineering” by Robert J. Glushko and Tim McGrath, to be published in 2004 by MIT Press.

Document Exchange in Business Architecture

Online stores, supply chains, business to business marketplaces, auctions, information brokerages and news syndication networks are all examples of business practices or applications that use information exchanges to combine or interconnect products or services from multiple businesses. Every one of them relies on the exchange of documents that describe the products or services being offered, their buyers and sellers, their origins and destinations, the amount and proof of payment, and so on. Each of these ways of doing business implies distinct patterns in inter- and intra-company coordination carried out by document exchanges.

The nature and repertoire of these patterns evolves whenever new information technologies emerge. In the 19th century the teletype and telephone made it possible to coordinate business activities at a scale vastly larger than before, leading to the rise of the modern corporation. In the late 20th and early 21st centuries we have been witnessing the equally profound impact of the Web, electronic mail, and XML on how businesses work. Put simply, innovation in communications and information technology has enabled more frequent information exchanges, more granular information exchanges, and information exchanges with more partners.

The global scope and speed of these information exchanges possible on the Internet has transformed how business gets done in almost every industry. But in one essential way things have not changed – the information being exchanged is packaged or presented as documents.

Putting document exchanges at the foundation of every business transaction is not a new idea. This basic need to record business transactions and related administrative information was a primary motivation for the invention of writing in Mesopotamia and Egypt between four and five thousand years ago.

An important and unchanging characteristic of document exchange between businesses is technology independence. The meaning of a Mesopotamian document didn’t depend on the fact that it was expressed as Babylonian cuneiform markings in...
clay. Likewise, when two modern 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 should not need to agree on or even know anything about the technology used to implement the processes.

**Understanding What Documents Mean**

For most people the experience of 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 consciousness in a natural and automatic way, as if the words were “self-describing” and directly conveyed their meaning. This is precisely why we can 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 here, 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 to understand. 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 might be described with the same words (“Address” might mean “Buyer’s Address” in one document and “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 “tomorrow” but not if today is Friday or Saturday because “Day” in “Next Day Delivery” means “Business Day.”

So we need to be diligent and precise when we define the words and information content produced and consumed by business applications. To do this we create data dictionaries, document schemas, programming language classes, database schemas, UML class diagrams, spreadsheet templates, system documentation and other representations or models of the structure and semantics for the concepts and objects needed in our applications.

In the ideal world we end up with a complete end-to-end view of how information and data are defined and used throughout our enterprise – what is often called the enterprise data model or enterprise information architecture. But too often people don’t do this as completely or as conscientiously as they should, either by failing to recognize the seriousness of the problem or by rationalizing that have insufficient time, expertise, resources, or incentives. As a result, there can be substantial differences in the meaning and format of information in the variety of applications within a single enterprise.

And even if each enterprise in a business relationship were disciplined in its own approach to modeling and describing the information it uses, 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. Of course, one way to do this would be for every enterprise to use exactly the same definitions for the information components of their documents. But it is more practical that they would use each other’s
document definitions to help them translate or transform the information they receive from so that they can understand it.

It is also essential for both parties in a document exchange to agree on the purpose of a document, a kind of meaning that is often best expressed in terms of the business transactions required to complete the exchange.

**Documents as Interfaces**

With so many information-based activities moving to the Web the browser has become ubiquitous as a user interface, and countless applications and services use a “fill-in-the-web-form” paradigm to automate processes that previously relied on printed forms. We can readily imagine workflow applications where form-based information moves within and between companies -- submitting a timesheet, seeking reimbursement for expenses, applying for a position, registering for classes or events, and so on.

In an online bookstore, some pieces of information like the customer’s name and address or the title of the book being purchased flow through many documents, applications, and formats. The customer’s order from an online catalog might be captured using a Web form in HTML. Information extracted from this form ends up in other documents, like those relating to availability, shipment, or billing. These documents are likely to be generated by legacy or enterprise applications and might be produced in a proprietary format or as EDI messages. These legacy documents are increasingly being transformed into XML when they are conveyed across enterprise boundaries because of XML’s flexibility in precisely encoding the information model produced at one end while making it easy for a receiving application to extract the information needed on its end and to transform it if necessary. XML has the added benefit that it can easily be rendered in a browser using a style transform if there is a need for a person to view it.

The more complex and heterogeneous the information technology environment in which business takes place, the more it makes sense for us to look at business relationships from the perspective of the technology-independent abstraction of document exchange. This perspective generalizes the idea of documents as interfaces for users to the idea that documents are interfaces to business services or business processes. In both cases the document conveys information in and out without necessarily revealing anything about how the information is consumed on the way in or created on the way out.

**Web Services**

Business people are becoming as comfortable as technologists with the idea that the Internet is a standards-based platform for document-centric computing. Using documents as interfaces and thereby hiding implementation details underlies the idea of web services as a way to create new applications or systems by combining components of other ones.

Turning some application functionality or information resource into a web service puts it in a document exchange “wrapper” with defined input and output documents and
makes it accessible over the Internet using standard addressing methods and message protocols. The standards for describing the document interfaces of web services enable them to be combined to create a composite application like our online bookstore or to coordinate the movement of information through a network of business relationships like a supply chain.

As more software providers adopt document interfaces for hosted services, enterprises can treat software resources as if they were a utility, giving them more flexibility and responsiveness as business conditions and customer demand change. To IBM, this is “business on demand;” to Microsoft, it is the “agile enterprise;” to HP, it is the “adaptive” one. Many other platform vendors, consultants, and professional services firms are touting the similar ideas about web services and service-oriented business architectures.

Obviously, we are as enthusiastic as anyone about the promise of web services to reduce integration costs for otherwise incompatible systems and make businesses more responsive and flexible. However, it can be easier to talk about using documents as interfaces than it is to modify an application to work that way. While some vendors of “web services enablement” tools claim that a Java application can be set up as a web service quickly, the ad hoc or proprietary data structures and formats for legacy systems require much more effort to expose them as web services. We’re not quite there yet, but many people are working hard to get us closer.

Model-Driven Applications
But the proliferation of vendor offerings about web services and their associated marketing campaigns shouldn’t obscure the two simple ideas about document exchange at their core that we’ve discussed so far. The first is that the specifications for the document content are encoded in XML content models or schemas. The second is that the business logic that controls the document exchanges is represented in some workflow or choreography or business process specifications. These two ideas are broadly applicable to both software engineering and application development, not just to web services.

XML has become a unifying technology for implementing applications that use Internet protocols and web service standards, especially for those that span enterprise boundaries. Expressing information content and document exchange logic in the computer-friendly XML format enables applications to be deployed efficiently and robustly at reduced cost. XML from documents, databases, and enterprise applications can be combined and treated as a single source from which content can be transformed for multiple users, devices, or applications. XML-centric programming paradigms are emerging in which XML schemas, programming language objects, database schemas, and UML models can be treated as equivalent.

Generating application code or configuring an application platform from XML specifications bridges the traditional gap between design and implementation. Such model-driven applications are vastly easier to maintain. Rapid application deployment and modification is essential as businesses experiment with business models that take advantage of the Internet.
Document Models Come From “Document Engineering”

But where do these models for document content and business process come from? We propose that they should come from Document Engineering.

Document Engineering is evolving as a new discipline for specifying, designing, and implementing the models of documents that use Internet information exchange mechanisms to request or return the results of business processes. The essence of Document Engineering is the analysis and design methods that yield precise models that describe the information these processes require and the rules by which related processes are coordinated and combined.

We don’t mean to imply that every information or process model needs to be created from scratch – far from it. Just as every other engineering discipline does, document engineering emphasizes the reuse of existing models because they describe successful examples or embody best practices, reducing costs and risks while increasing reliability and interoperability. Useful models for document engineering include those encoded at the implementation level in the form of XML schema libraries or EDI message standards. Other useful models are at more conceptual levels in terms of industry reference models for common business processes, or even in more abstract patterns for the organization of activities between businesses using supply chains, marketplaces or hubs.

Sometimes, of course, no existing model is suitable, and then it is necessary to follow an engineering approach that yields a model that meets your requirements. But even then it is essential to design and implement the model in a manner that enables its subsequent modification and reuse.

The analysis and design methods of Document Engineering have their roots in other disciplines, primarily information and systems analysis, business process modeling, electronic publishing, and distributed computing.

But we believe that the synthesis of the complementary ideas from these separate fields is a worthy discipline on its own.