Document Engineering: Analyzing and Designing the Semantics of Business Service Networks

Dr. Robert J. Glushko
University of California Berkeley
glushko@sims.berkeley.edu

Tim McGrath
Universal Business Language
tmcgrath@portcomm.com.au

Abstract

Behind the idea of Business Services Networks lies the very simple and natural idea of document exchange. But when they are implemented without disciplined semantics, the input and output documents of business processes often partition their information in incompatible ways, severely constraining the loosely coupled, "plug and play" interoperability that is the defining vision of business service networks.

This paper introduces the new discipline of Document Engineering, a set of analysis and design techniques that yield meaningful and reusable models of the information exchanges within and between enterprises.

Document Engineering relies on the skills and tools of business process, document, data, and task analysts. One of the innovations of Document Engineering is to exploit these different techniques for reaching the same goal

As a result, with business services developed using Document Engineering techniques the semantics are precise, based on patterns to make them reusable, and support a complete and unambiguous relationship between the model and the schemas and software that implements it.

1. Introduction

Behind the idea of Business Services Networks lies the very simple and natural idea of document exchange.

Whether they were encoded on papyrus, parchment, paper, or in digital forms, documents have always served as the packages of information needed to carry out business transactions. The seller may ask, "What do you want to order from my catalog?" and the buyer might ask, "Will you accept my purchase order?" They would never first ask about each other's system interfaces. The notion of documents as the inputs and outputs of business processes wherever they reside on a network is a technology-independent abstraction perfectly suited to the heterogeneous technology environment of the Internet.

Using documents as loosely coupled interfaces and thereby hiding implementation details underlies the idea of service-oriented architectures as a way to create new applications as services by integrating or combining components of others. A Web 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. And because Web Services are loosely coupled, their document interfaces allow firms to maintain a clean and stable relationship to partners and customers.

But while the Web Services standards dictate how to reveal the interfaces and message definitions for the XML documents that they send and receive, they say nothing about the conceptual design of those services and their enabling documents. They tell us how to package information into documents and where to put them, but they don't tell us what any of it means. When they are implemented without disciplined semantics, the input and output documents of Web Services often partition their information components in incompatible ways, severely constraining the "plug and play" interoperability that is the defining vision of business service networks.

We have developed Document Engineering [1] as a set of analysis and design techniques that yield meaningful and reusable models of business processes and their documents. Document Engineering is a novel combination of words that highlights 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.

2. Document Engineering in a Nutshell

At its essence Document Engineering is a document-centric adaptation of the classical three-level modeling framework. This approach distinguishes between external representations that describe specific things, artifacts, or instances in the world, physical (or internal) views that present different models of instances in some technology, and conceptual views or models that abstract those descriptions from any particular implementation.

Document Engineering isn't only concerned with the semantic components in the documents being exchanged. It is also concerned with the information exchanges within and between enterprises and the techniques for contextualizing them for particular industries or domains.

Describing business processes and their documents in terms of the more conceptual notion of information exchanges makes it easier to understand the constraints imposed by legacy systems and technologies. We are also able to recognize the opportunities created by new processes if we focus on conceptual models of the information exchanges rather than on how they are implemented.

As a result, in business services developed using Document Engineering techniques the semantics are precise, based on patterns to make them reusable, and support a complete and unambiguous relationship between the model and the schemas and software that implements it.

2.1. The Model Matrix

Document Engineering distinguishes three levels of abstraction. The least abstract models describe specific instances of business documents, processes, or other artifacts. Physical models are more general because they describe a set or class of instances, but they still capture the technology in which the instances were implemented. Conceptual models remove the implementation technology to emphasize the semantic relationships that define some class of instances.

We can also distinguish what businesses do according to the depth or granularity with which we describe the business relationships in each model, creating a hierarchy that helps us analyze and design business services and their networks. From the organizational or business-to-business perspective, models are coarse with just the most important roles and relationships visible. At the process level, more details of the relationship are visible, and we begin to see the documents that are exchanged to carry out each process. The information level is the most granular perspective, and we can see specific information components within the document models.

These two dimensions of model abstraction and model granularity let us define a model matrix (Figure 1) that shows in a single diagram the relationships among business model, business process, and business information models at both conceptual and physical levels. This gives us a framework for discussing the most important and reusable models and for explaining how the most granular models for business information and business processes are composed and choreographed to create more complex models of greater scope.

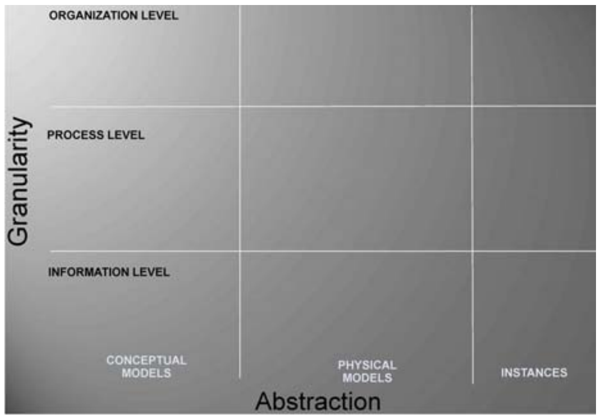

Figure 1. The Model Matrix

The models we organize in the upper left corner of the matrix are broad in scope and abstract in perspective. It is useful to take this broad perspective on what businesses do and the relationships between them because models at higher levels of abstraction and granularity establish the requirements and rules for more granular ones in which documents are specified.

We call these sets of requirements, the context of use. When these rules are expressed in models we can use them to design and drive the business services using them. We can then share the models with other organizations and enterprises and promote interoperability by ensuring that we understand each other's contexts.

As we move to the right and down in the matrix, models become narrower in scope and more concretely tied to technology and implementation.

3. The Yin and Yang of Document Engineering

An important idea embodied in the model matrix is the essential and inescapable relationship between models of processes and models of documents. At the center of the model matrix, where processes are described as transactions, processes and documents are two perspectives on the same thing. Are process models just combinations of document exchanges, or are documents just the payload in processes? The answer is yes to both questions. Business process and documents are the yin and the yang of Document Engineering.

These central concepts of Chinese philosophy might seem out of place here, but they express perfectly the complementary and opposing relationships between business processes and documents. Processes produce and consume documents, which are a static snapshot or the tangible result of the process activity. Process descriptions emphasize business concerns and determine whether ways of doing business are compatible. Document descriptions emphasize technology concerns and determine whether business systems are compatible. We can separate processes and documents in our analysis, discussion, and models, but in the end they are always interconnected because both business and technical compatibility are necessary.

In practical terms this means that models for processes and documents need to be developed with the same care and to compatible levels of detail. It explains why we need a Document Engineering approach that exploits complementary modeling approaches.

4. A Unified Approach

There is no single correct way to create models of business processes and their documents. Document Engineering relies on the skills and tools of business process, document, data, and task analysts. One of the innovations of Document Engineering is to exploit these different techniques for reaching the same goal.

Figure 2 graphically depicts this common goal as reaching the middle of the model matrix from different starting points.

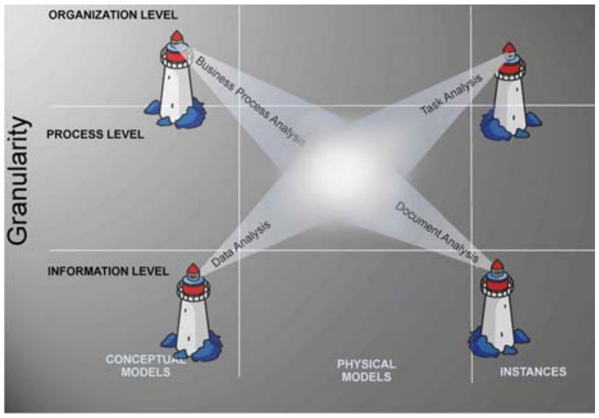

Figure 2. The Unified Approach

Business process analysis typically starts with abstract views of business models and processes. These are organized in the upper left corner. This high-level analysis establishes the context for understanding the semantics of the information in the other sections of the matrix.

Task analysis (or user analysis) is the observation of people performing the tasks or use cases when the application or system must support human interfaces and not just other applications. Task analysis identifies the specific steps and information that people need to carry out a task, so it is based on actual artifacts and activities, which are represented on the right side of the matrix. Task analysis and document analysis are closely related; document analysis reveals candidate information components and task analysis reveals rules about their intent and usage. Task analysis is especially important when few documents or information sources exist, because human problems or errors can suggest that important information is missing.

Document analysis tends to start from analysis of document instances. We show this on the lower right side. These techniques extract or disentangle the presentational, structural, and content components of documents or other information sources.

Data analysis (or object analysis) techniques often start from a conceptual perspective about a domain and yield an abstract view of the information components revealed by document analysis. So this approach is represented as starting from the lower left corner of the model matrix.

5. The Document Engineering Approach

Figure 3 depicts the Document Engineering approach as a path through the model matrix to carry out a set of analysis, assembly, and implementation tasks.

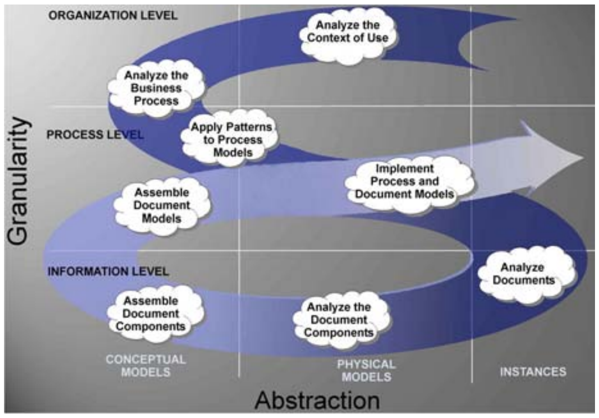
ORGANIZATION LEVEL
Analyze the Context of Use
Analyze the Business Process
PROCESS LEVEL
Apply Patterns to Process Models
Implement Process and Document Models
Assemble Document Models
INFORMATION LEVEL
Analyze Documents
Assemble Document Components
Analyze the Document Components
CONCEPTUAL MODELS
PHYSICAL MODELS
INSTANCES
Granularity
Abstraction

Figure 3. The Document Engineering Approach

We show this path as being equally wide as it winds its way through the phases of Document Engineering, but in practice different phases may get more or less emphasis.

Top-down or strategic efforts to align business organization and technology cut a broad swath through the top of the model matrix. These efforts create models that are very abstract or very generic, partitioning activity into large, goal-oriented chunks to provide a big picture view of the context of use.

In contrast, bottom-up and more document-driven projects emphasize the path through the lower half of the model matrix. These efforts may yield a large number of models for transactional processes, often refined by the specific types of document they produce or consume.

But high-level goal-oriented models lack the detail needed for implementing and integrating the applications built to achieve them, and low-level models of documents and information components by themselves don't provide much help in aligning high-level business goals with technology choices and implementation decisions. That's why it is worthwhile to follow the entire path through the matrix. Developing a variety of models of varying emphasis and granularity ensures that any new models we create for business processes and documents are complete, consistent, robust, and deployable in applications that meet actual business requirements.

Following the complete path also helps to overcome the fundamental modeling challenge of achieving a consistent level of abstraction so that patterns and models from different perspectives can fit together. There is also a large granularity gap between business models and document models. Our path through the model matrix yields successively more granular models that bridge the gap.

5.1. Understanding the Context of Use

What we call context is actually the collective sum of the requirements for our area of interest. So the first phase of Document Engineering, analyzing the context, involves identifying these requirements and the rules they must satisfy.

On the top row we begin with high-level, organizational analysis to understand the main business activities and the people and organizations that participate in them. This strategic perspective is an essential foundation for developing infrastructures needed by service-oriented architectures or in carrying out corporate mergers and acquisitions. Models at this level describe the broad context of use for the documents and processes we will define at more granular levels.

5.2. Patterns for Business Processes

Patterns are models that are sufficiently general, adaptable, and worthy of imitation that we can reuse them. A pattern must be general enough to apply to a meaningfully large set of possible instances or contexts. It must be adaptable because the instances or contexts to which it might apply will differ in details. And it must be worthy, that is the instances or contexts to which the pattern might apply should benefit from following it.

Indeed, sometimes a pattern is so carefully constructed or consistently adopted it becomes an official or de facto standard. So just like every other engineering discipline, Document Engineering emphasizes the reuse of existing specifications or standards. Doing so reduces costs and risks while increasing the reliability and interoperability of the deployed solution.

Using Business process patterns resolves the tension within an organization between meeting internal requirements and interacting with others. This is why much of what businesses do can be described using a small repertoire of business patterns because they share common requirements for their context of use [2].

So we believe effective business process design should focus on the analysis, reuse, and creation of patterns.

Choosing and instantiating appropriate patterns for business processes entails adopting a predefined context of use. Using a business process pattern also suggests the relevant users and other stakeholders from whom we can obtain or confirm requirements. For example, we might describe Dell Inc. as a computer manufacturer because we want to highlight its Make-to-Order manufacturing pattern and focus on its relationships with organizations in its supply chain. But if we wanted to emphasize Dell's direct sales model, we would apply a pattern that brought with it the set of requirements and business rules associated with direct distribution.

5.3. Analyzing Documents and their Components

Describing the actual documents needed by a business model starts to take place during the document analysis phase.

The selected process model (or pattern) will identify the roles that documents play. But it is when we undertake document analysis that we expose the business rules that govern the content, structure, presentation, syntax, and semantics of the information contained in the documents.

We analyze existing document models (such as XML schemas) as well as any document guidelines and standards, sample document instances, web pages, and other information sources to harvest all potentially meaningful information components and the constraints that govern their values, arrangement, and use. Of course we can’t ignore the people who create, review, approve, query, or do other things with these documents. In particular, in models for new business processes where few documents exist, what we can learn from people is critical because we can derive information and document requirements from their goals and tasks. In many situations existing documents are extremely valuable proxies for, or confirmations of, what people tell us.

The component analysis phase starts with the harvesting task. This identifies the individual semantic components contained in each of the selected documents or information sources.

In component assembly we assemble sets of these information components into meaningful structures to create a coherent conceptual view we call the document component model. We advocate doing this by using data analysis techniques that normalize the components into structures based on their functional dependency.

5.4. Designing Document Models

We then turn from analysis to design as we start to create models for new types of documents based on the components, structures, and associations that satisfy the requirements for our context of use. We call these new models document assembly models. In the document assembly task we apply the rules for assembling each new document type from the information components described in our document component model.

As with process models, effective design of document assembly models also requires us to recognize when patterns (or standards) can be reused, when a new pattern should be created, and what distinguishes one pattern from another.

5.5. Implementing Models

The conceptual models we have described so far represent substantial investments in understanding sets of business rules and capturing contextual requirements. In the implementation tasks, we start to create modeling artifacts that will actually define or drive applications and their interfaces. We want to use these in an explicit way to implement a solution in an automated or semi-automated manner. This is what we mean by a model-based application.

For many applications in the domain of Document Engineering, all or much of the model-based functionality involves business service networks. For example, a well-designed Web Service exposes its interface as a document model and interacts with other services according to a process model. These models specify the information components produced and used by the service, and any code that is needed to receive the document and extract its information components can be automatically and reliably generated from them in most cases. There should be a complete and unambiguous relationship between components of the model and components in the code that will process them.

Model-based applications can then be implemented using software whose generic functionality is made context-specific by configuring or extending it to use the context-dependent information and behavior specified in the model.

The first step in achieving this is to realize the conceptual models in physical models.

For documents we call these physical models, the document implementation model. Document implementation models realized in markup languages are more commonly known as schemas. For example, when XML is used to encode document implementation models, many aspects of the integrity of a document’s information components, as well as the business rules applied to the data, can be derived from the XML schemas.

For physical models of business processes, realization means adopting a suitable metamodel (such as the ebXML BPSS [3]) to encode the specific rules and the requirements for our given context of use. This means that the modeling artifact itself is encoded as a document. We call this realized artifact, the business process implementation model.

Using this approach, we would say that the RosettaNet PIPs [4] are examples of business process implementation models encoded in XML using their Implementation Framework metamodel [5].

Web services and service-oriented architectures can be implemented in this model-based way when the document and process models they use are designed to separate generic and context-dependent functionality.

5.6. Encoding Models in XML

We can encode implementation models in any of several different XML schema languages. Each offers different tradeoffs in simplicity, expressive power, and maintainability.

Choosing an XML schema language includes the potential to reuse patterns from existing XML vocabularies. These are usually published as physical models using one schema language as their authoritative format. For example, the UBL vocabulary provides XML Schema definitions for common components such as Item, Party, Tax, Address, Amount, and Location [6].

Of course, interoperability or legacy constraints at the edges of business service networks may mandate other encoded representations such as UN/EDIFACT (ISO 9735) [7], ANSI ASC X12 [8], and industry-specific legacy syntaxes [9]. We may even have to encode our document implementation models in several syntaxes. For example, the UN/CEFACT group [10] has chosen both UN/EDIFACT and XML syntaxes for their document implementation models.

Choice of representation language alone is not sufficient to realize a document implementation model. Regardless of the syntax chosen, it is also necessary to develop or adopt rules that govern the techniques for encoding that language.

With the increased emphasis on XML in service-oriented architectures and web services, many books on XML encoding have emerged, but there still seems to be more emphasis on the nuances of schema syntax (there are too many) rather than on the best way to encode the semantic content [11].

A promising exception comes from the UBL initiative, which has produced a comprehensive set of naming and design rules for encoding document assembly models (based on ebXML Core Components [12]) into XML Schemas.

6. Summary

The services in a Business Service Network involve both documents and processes. To make these services work, the organizations involved must implicitly or explicitly reach a common understanding about how their processes should be designed, how they can be deconstructed into document-based service components, the information they exchange with the documents, 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, formal rigor, and traceability to the requirements and artifacts from which they were developed. This can happen only if document, data, task, and business process analyses can be brought together in a unified approach like that in Document Engineering.

7. References

[1] Glushko, Robert J. and Tim McGrath, Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services. (MIT Press, 2005). This paper is a modestly adapted extract from that book.

[2] Collections of process patterns can be found in Thomas Malone, Kevin Crowston, and George Herman (Eds.), Organizing Business Knowledge: The MIT Process Handbook (MIT Press, 2003) (http://ccs.mit.edu/ph/) or in the International Benchmarking Clearinghouse managed by the American Productivity and Quality Center (APQC) (http://www.apqc.org)

[3] The ebXML BPSS is a business process metamodel described as an XML schema (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ebxml-bp)

[4] The RosettaNet PIP Directory, (http://www.rosettanet.org/pipdirectory)

[5] The RosettaNet Implementation Framework, (http://www.rosettanet.org/rnif)

[6] The Organization for the Advancement of Structured Information Standards (OASIS) Universal Business Language. (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl)

[7] UN/EDIFACT is the international EDI standard maintained by the United Nations Centre for Trade

Facilitation and Electronic Business (http://www.unece.org/cefact).

[8] ANSI ASC X12 is the US EDI standard maintained by the Associated Standards Committee (http://www.x12.org)

[9] An example of this is BISAC for the book publishing industry (http://www.bisg.org).

[10] The United Nations Centre for Trade Facilitation and Electronic Business (http://www.unece.org/cefact/)

[11] For example, James Bean, XML for Data Architects (Morgan Kaufmann, 2003), or Berthold Daum, Modeling Business Objects with XML Schema (Morgan Kaufmann, 2003)

[12] ebXML Core Components (http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf)