Modeling Methods and Artifacts for Crossing the Data/Document Divide

Robert **Glushko**

Abstract

Many people have contrasted narrative types of documents that mostly contain text with transactional types that mostly contain data, and they typically conclude that documents and data require different terminology, techniques, and tools in XML vocabulary development. But narrative and transactional documents are often closely related, either by structural transformation or by business processes. The emerging discipline of Document Engineering proposes a document-centric reformulation of traditional data analysis, and recasts its formal and specialized methods like normalization 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 identified in transactional contexts.

Table of Contents

1. Introduction

The design, implementation, and successful deployment of a document-centric application or service involve many interrelated activities, most of which involve models of documents and the processes that use them. The first step is usually a requirements gathering and scoping phase that defines the context for the project. For strategic projects, this context is often defined by coarse goal-oriented process statements or models. For tactical projects, the context is often defined in terms of specific types of documents that must be automated or integrated. The bulk of the work involves the creation of conceptual models of the documents and processes at compatible levels of abstraction to ensure that the documents can be used by the processes. One of the last steps of the overall effort is using the document and process models in application or user interfaces, for generating code, or for configuring an application.

Our focus in this paper is on the central tasks of creating the document models. We propose an approach that resolves an apparent conflict between models of narrative documents and models of transactional ones.

Many people have contrasted narrative types of documents that mostly contain text with transactional types that mostly contain data, and they typically conclude that documents and data cannot be understood and handled 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."

1.1. Overview of Document Analysis

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.

Eve Maler and Jeanne El Andaloussi's Developing SGML DTDs: From Text to Model to Markup [Maler1995] is the definitive treatise on document analysis with SGML for technical publications and other published types of documents. Published in 1995, this book was the first to systematize the evolving best practices for using SGML as an encoding syntax for models of document types. The book's subtitle elegantly summarizes the goals of classical document analysis: to analyze a set of texts, create a conceptual model of their information components, and then encode the model in a markup syntax like SGML or XML.

More specifically, 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 document schema or document type. 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.

1.2. Overview of Data Analysis

Data analysis has its roots in philosophy and linguistics, but it is most currently understood as 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. 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 these information components are often stored as large structured sets of data in databases, data analysis is a key step in database design. The conceptual model created via data analysis methods is also typically called a schema, but this is generally meant to be a "database schema" rather than a "document schema," and it would never be called a "document type."

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, both of which facilitate the development of a conceptual model that describes them. Furthermore, unlike narrative documents, transactional documents typically contain minimal or arbitrary presentation features.

The stronger constraints in contexts that include transactional documents can be formally specified and analyzed, and the principles and methods for doing so were first developed as part of Codd's Relational Theory [Codd1970] for the design of databases. In particular, the principle known as normalization involves a set of techniques for modeling components and structures that minimizes redundancy and supports integrity.

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.

1.3. The Document Type Spectrum

Undeniably, documents and data differ in important ways, and these influence the techniques by which they are analyzed and by which models of them are developed. But it is essential to view narrative and transactional information on a continuum. Glushko and McGrath call this the Document Type Spectrum by analogy with the continuous rainbow formed by the visible light spectrum ([DocEng2002]). At the endpoints, 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.

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

The point is that this is a continuum. Defining document in a technology-neutral way as a purposeful and self-contained collection of information seems to cover both ends of the continuum and lets us avoid trying to impose a distinction about where some information ceases to be a document and becomes a set of data.

1.4. Unification in Document Engineering

Once we acknowledge the Document Type Spectrum, we are motivated to emphasize what document analysis and data analysis have in common instead of focusing on their differences. This unification or synthesis is embodied in the emerging discipline of Document Engineering ([DocEng2005]). The methods and artifacts used in Document Engineering build on those aspects of traditional document analysis that are appropriate for transactional documents while merging them with techniques from data modeling and object-oriented design. Combining these two approaches exploits the rigor of data modeling and normalization to make document analysis more systematic, while exploiting the heuristics and flexibility of the latter to make the former more pragmatic.

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.

2. Setting the Stage to Compare and Synthesize Document Analysis and Data Analysis

The review and synthesis of the methods used to analyze and model narrative and transactional documents in this paper is organized in terms of the three central modeling activities of Document Engineering. These are harvesting components, consolidating components, assembling components, and assembling document models. We will briefly introduce these four activities and then treat each of them in depth to compare, contrast, but ultimately to converge how we think about them for narrative and transactional documents.

We analyze existing document models as well as any implementation 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 domains or new business models 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.

After we have harvested and disaggregated candidate components from our sampled inventory of documents and information sources, we need to ensure that every component is semantically distinct. That is, we want to have only one name for each component. This means we must merge synonyms (candidate components with different names but the same meaning) and rename homonyms (candidate components with the same names but different meanings). Our resulting modeling artifact will be a consolidated table of content components.

These candidate components are a set of meaningful building blocks that can use be used to assemble semantically richer structures and models of documents. Next, 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.

We then use this component model to assemble the components into one or more document assembly models. The basic model of a document consists of two types of components, the content components that contain discrete information values and the structural components that are aggregations of the content ones. On this basis, a document model can be described as a top-level structural component that assembles the set of components needed to carry out a self-contained exchange of information.

Defining this document model is a two-stage process. First, we must assemble our components into structural building blocks composed of dependent components. These structural components also associate with other structural components in various roles. This creates a generalized view of the domain or context of use sometimes known as a Domain Model but which we prefer to call a document component model.

Each document assembly model takes a different view of the document component model by following the relationships between components that enforce the interpretation required for its more specialized context of use. The document assembly model can then be encoded as an XML schema.

3. Harvesting Components

Our ability to understand the common semantics embodied in the inventory can be constrained by differences in how the documents or information sources are presented. To identify the concepts and meanings for our components, we need to see past these differences. Extracting the underlying semantic components from their physical implementations is called harvesting the inventory.

Documents from all parts of the Document Type Spectrum contain components but those we find on the narrative end tend to be presentational and at the transactional end more content based. In the center we find structural components

that may be required for presentational or semantic requirements. Put another way, it is the mix of the three varieties of components that determines where a document fits along the spectrum.

The easiest components to find are those in transactional documents. Here candidate components typically appear as labeled data entry fields in forms or are explicitly marked up or delimited in some stream of data. It is also useful to study the source code of any relevant applications that process documents. These may contain both business rules and hidden components disguised as data variables. Application or markup code often describes a component more precisely than the labels that appear in a user interface.

In contrast to transactional documents, documents on the narrative side of the Document Type Spectrum are likely to have fewer, harder-to-identify candidate components. These documents generally have fewer processing requirements and therefore fewer rules about specific components. And with fewer rules, we need fewer components because we don't need to distinguish them.

This is true, almost by definition, for documents that are entirely narrative because they tell a story whose themes, characters, and plot develops gradually. While it is possible to label sections or chapters of the text with titles or assign index terms to them, it just isn't useful to treat those parts as specialized types of content on that basis.

Narrative documents can hide or obscure candidate components in paragraphs or other blocks of text. Document analysts refer to these as "Mixed Content" components because they are mixed into surrounding text that may be more generic. A common form of mixed content is an otherwise unstructured text paragraph that contains emphasized words, glossary terms, references to tables or figures, citations to supporting documents, or links to footnotes or endnotes (these are often called "Inline" components).

As we move from the narrative end of the document type spectrum toward regions with hybrid document types such as reference books, product documentation and operating or assembly instructions, components are more readily identified. Sometimes the components are explicitly labeled, such as Note, Warning, or Instructions, but most of the time they aren't. But we can often recognize components such as Question, Answer, Code Example, Illustration, Caption, Map, and Portrait. There are more presentational rules about these components so they occur more predictably and have a more consistent appearance when they do.

All types of documents contain structures that group their components. These can be either presentational or semantic, but we are most concerned with the semantic ones.

Because the rules governing narrative documents tend to be weak and nonspecific, there are limits to the rigor with which components can be grouped into structures. So the set of candidate components that emerges from analysis of narrative documents is typically a combination of content and presentational structural ones (such as lists and tables). The relative proportions of each type of component in this mix are hard to predict and heavily influenced by the skill and biases of the document analyst.

Semantic structures are more evident in transactional documents. Because of their formal and precise definitions, we generally find semantic structures implemented as containers for components. In more narrative types of documents, cross-references, footnotes, and hypertext links and anchors are the most typical mechanisms for implementing semantic structures.

4. Consolidating Components

We can begin consolidation with the candidate components from any of the information sources, but we recommend using the most authoritative source or the one that yielded the most components. This is usually a transactional document type.

Sometimes it will be difficult to decide whether a candidate component duplicates one already in the consolidated table. To improve semantic understanding, we should confirm any business rules that apply to harvested components,

especially those that constrain its possible values or dependencies on other components. These constraints are more numerous for components harvested from transactional documents but they can exist for narrative components.

A good way to distinguish homonyms is to add a context qualifier to create more precise names. We might distinguish among the three types of Item Identifiers by naming them Specific Item Identifier, Catalog Product Identifier, and Line Item Identifier.

Having harvested and consolidated a list of components, we can take our consolidated table of candidate components and construct a conceptual model of the components by assembling them into a document component model.

5. Assembling Component Models

The ultimate objective of information analysis in Document Engineering is to create a generalized, conceptual model capable of expressing the business rules for all types of documents required within the context of use. We call this artifact a document component model.

A document component model should define all the necessary components that maximize reuse and minimize redundancy when designing new document models.

How rigorously you can define a document component model depends on the number and precision of the business rules we need to satisfy. But there are some simple principles.

A small set of loose rules indicate a context of use that can be satisfied by a simple document component model, while more precise rules demand more sophistication in the model.

Business rules and the components that emerge from transactional documents tend to be more content oriented. This means the components for these contexts lend themselves to precise definition in terms of data types, possible values, and occurrence restrictions.

In contrast, the rules emerging from contexts dominated by narrative documents are more qualitative and less precise. So the components that emerge from their analysis tend to be larger, have a more generalized meaning, and are less suited for or subject to absolute instance or structural rules. For example, in contexts with more narrative types of documents such as technical publications, reports, policies, procedures, and reference books, the content components tend to be in coarser blocks of text without much regular internal structure. Even in these components the rules we discover may concern the relationships among components and reflect principles or best practices in document design. These are often specified in style guides, rules, or templates that guide authors when they create these types of documents.

Indeed, it is often because of the obvious need to leave decisions about content up to the author in some contexts that narrative documents are employed rather than transactional ones. Instances of narrative documents like repair procedures, policies, or textbooks might conform to structural rules, but their content is inherently heterogeneous and the semantic relationships among their content components can be only weakly specified. We can't easily reduce the process of creating them to "filling out a form" in which every instance has exactly the same components in precise and fixed relationships.

The differences between transactional and narrative style documents and the nature of the business rules that apply to them influence the approach to creating document component models. In particular, they may determine the extent to which we can employ formal and rigorous techniques to decide which candidate components go together to create aggregate or composite structural components.

The weak semantic constraints in narrative components often don't provide unambiguous justification for deciding what components go together. But this doesn't mean we have no way of advancing the goals of increased regularity and minimal redundancy in our component models. We can use our judgment as designers to enforce stronger and clearer constraints in models. We can also eliminate choices that could lead to inconsistencies or interoperability problems.

Of course, in contexts where there are few strong semantic constraints, you can't fully exploit the power of formal techniques for assembling document components. We may resort to an approach that assembles structures iteratively through a kind of reverse engineering of the documents required or suggested by the context. This approach is called "Core plus contextualization" in Document Engineering, and in effect bypasses the formal analysis of component assemblies in favor of direct assembly of document models (see [Bloodworth2004] for a modeling case study that contrasts normalization and "core plus contextualization.")

Nevertheless, while the processes of normalization are formally defined, ultimately it is the heuristic interpretation of business requirements and rules that determines how we apply these formalisms and therefore determine the quality of the final component model.

Practical experience tells us that we may need to reverse or relax our interpretation of dependency for the sake of other requirements such as simplicity, interoperability, or efficiency. This may result in a smaller set of somewhat larger components than complete normalization would yield.

But even in these cases, having dependent component structures as a reference allows us to make conscious rather than ad hoc modeling refinements. In other words, while it is not essential to have a perfectly normalized document component model, we should at least understand if and why it is not.

One common refinement is to try to reduce the number of associations in the model. For example, if we determine that actually having multiple instances of a role in an association is rare, even though not impossible, it may be simpler to merge the two structures and accept some potential duplication and therefore redundancy. In effect, we denormalize parts of the model.

6. Assembling Document Models

A document component model might be the final modeling artifact if we were designing databases, but we have more work to do if we want to design documents.

The reason we're not done yet is because a document is a self-contained set of information for a specific purpose. So a document that describes a book, tax receipt, customer, purchase order, or flight reservation will organize the information it contains from the perspective of a single transaction or event. A book is published, taxes paid, a customer signed up, a purchase order issued, a flight reservation made. But a document component model is a description of the network of all possible interpretations of the components and their associations. If we want to exchange documents with a specific interpretation we need another kind of model. What we call a document assembly model is such a model.

By document assembly we mean defining a top-level structure and nesting the subsidiary components within a hierarchy to form an inverted tree of components. The challenge with document assembly is to design models that satisfy the requirements and optimize the reuse of common components.

When we are dealing with document exchanges, we don't want flexibility, we want unambiguous semantic interpretation. For example, in the model of a specific type of document we do not want to allow any alternative roles and associations, only those required for the context of that document. Therefore a document assembly model defines one document-specific view of the more complex document component model. For this reason documents are based on hierarchical models.

Each document assembly model imposes an unambiguous definition of a document structure. The hierarchy expresses rules about the use of the components. In effect, when we present a hierarchical document assembly model we are saying; "for this document, interpret the information this way." This ability of hierarchical structures to convey semantics makes them natural for documents and the document assembly models that define them.

Some types of documents, particularly those on the narrative end of the Document Type Spectrum, have such a common assembly model that it is immediately recognizable as a pattern. For example, many books assemble a foreword, preface,

introduction, chapters, appendices, and an index in that order. As they assemble their document models, authors and publishers employ these patterns because they recognize that those processing the documents will be familiar with them.

The assembly patterns for narrative documents might seem different from those for transactional documents such as orders, flight bookings, or calendar event submissions. However, they all follow the same principle that assembly is based on the contextual requirements for a given document.

Each type of document usually requires its own document assembly model. To start creating this model we must choose the structural component that will form the root of the document tree. We can think of this as the entry point into the document component model. Once we choose an entry point, we need to make decisions about the inclusion of other structures and their components. These decisions are based on the business rules the roles other structures have in their associations with the entry point structure. These rules are stronger in transactional contexts. First, the choice of associations available is influenced by the cardinality of the role. If the role is mandatory, the associated structure must be assembled into the model. Optional associations are assembled into the model only if their roles are required by structural or semantic business rules. For all structures in the assembly model we must also decide which content components are required. Again, we must include any mandatory content components. And once again, the use of optional components is based on the rules for our context of use.

The document model assembly follows associations through the component model and makes choices about which components to include until all the requirements of the context of use are satisfied.

7. Summary

Document analysis and data 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 and data as separate universes. Too many applications and services involve a mixture of narrative and transactional document types, and information can efficiently flow from one to the next only if it is modeled in a way that doesn't assume or favor one class of document type over another.

To produce document models like these, Document Engineering proposes a document-centric reformulation of traditional data analysis. But it recasts 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.

This synthesis achieves the composite goal of both 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.

Bibliography

[Bloodworth2004] Bloodworth, Allison and Glushko, Robert J., *Model-driven Application Design for a Campus Calendar Network*, IdeAlliance XML 2004 Conference, 2004.

[Codd1970] Codd, Edgar F., *A relational model of data for large shared data banks*, Communications of the ACM, 1970 (377-387).

[DocEng2002] Glushko, Robert J. and McGrath, Tim, *Patterns and Reuse in Document Engineering*, IdeAlliance XML 2002 Conference, 2002.

[DocEng2005] Glushko, Robert J. and McGrath, Tim, *Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services*, MIT Press, 2005.

[Maler1995] Maler, Eve L. and El Andaloussi, Jeanne, *Developing SGML DTDs: From Text to Model to Markup*, Prentice Hall, 1995.

Biography

Robert **Glushko**
University of California, Berkeley
Center for Document Engineering, School of Information Management and Systems [http://cde.berkeley.edu/]
Berkeley
California
United States of America

Bob Glushko is an Adjunct Professor at the University of California at Berkeley in the School of Information Management and Systems and is the Director of the Center for Document Engineering. He has twenty-five years of R&D, consulting, and entrepreneurial experience in information management, electronic publishing, Internet commerce, and human factors in computing systems. He founded or co-founded three companies, the last of which was Veo Systems in 1997, which pioneered the use of XML for electronic commerce before its 1999 acquisition by Commerce One. From 1999-2002 he headed Commerce One's XML architecture and technical standards activities and was named an "Engineering Fellow" in 2000. He has an undergraduate degree from Stanford, an MS (Software Engineering) from the Wang Institute, and a PhD (cognitive psychology) from UC San Diego.