8

Analyzing the Context of Use

8.0 INTRODUCTION 244
8.1 UNDERSTANDING DOCUMENT AND PROCESS REQUIREMENTS 247
8.2 CONTEXT AND REQUIREMENTS 254
8.3 EXPRESSING REQUIREMENTS AS RULES 259
8.4 RULE TYPES AND CONTEXT DIMENSIONS 268
8.5 KEY POINTS IN CHAPTER EIGHT 271
8.0 INTRODUCTION

During the first phase of a Document Engineering effort, we identify the context of the business problem and the requirements that must be satisfied by the documents and business processes in its solution. Understanding the problem in terms of a pattern like those discussed in Chapter 4 is desirable because it often helps us reuse all or part of an existing solution. In particular, a pattern can suggest which types of documents we’ll need to find or design, in which business processes we are likely to deploy them, and the relevant users and other stakeholders from whom we can obtain requirements and with whom we can test our proposed solution. The chosen pattern brings with it a set of requirements, rules, and constraints that we can verify and extend in subsequent analysis and design.

As we saw in the Model Matrix (Figure 3-7), patterns we might reuse range from abstract or generic to very specific. Conceptual business patterns like supply chain or document automation are widely applicable but don’t convey many specific requirements, whereas more specific patterns like collaborative, planning, forecasting, and replenishment (CPFR) or straight-through processing (STP) embody rich patterns of requirements that relate to specific business processes, industries, types of products, and so on. Adding requirements to an abstract pattern customizes it to suit our context of use.

Before we go any further, we must make it clear that when we talk about requirements in Document Engineering, we are focusing on the requirements that must be satisfied by models of documents and business processes and by their computer-processable implementations (most often using XML). Of course not all the requirements will emerge at the beginning, and a more complete understanding of the context of use will develop as we go through our analysis. In addition, there will be requirements for the software applications that will use or enforce the document and process models, but these are outside of the scope of Document Engineering. We don’t minimize the importance of identifying and satisfying these software requirements, but we don’t have anything special to say about those activities.
The requirements for the documents and processes must be precise and verifiable to be useful. This means we need to express them as rules (or constraints). We also find it helpful to distinguish different types of rules. Some types of rules define the meaning of information components, their possible values, and their presentation. Other types of rules govern the combination and assembly of individual information components into reusable aggregates and documents. Still other types determine the processing of information components and documents, including the roles and policies that control what processes or people can access or change information.

When these rules are expressed in models we can use them to define and drive the business services using them. We can then share models with other organizations and enterprises and promote interoperability by ensuring that we understand each other’s contexts.

Beginning with this chapter, we will detail the typical phases and activities of the Document Engineering approach, using as a case study the Event Calendar Network Project at the University of California, Berkeley.

---

**Introducing the Event Calendar Network Project**

In Chapters 8 through 14, we’ll illuminate many of the concepts used in the Document Engineering approach, using a project known as the Berkeley Event Calendar Network. This project makes a good case study because it deals with a familiar domain, illustrates many common business processes, and is relatively small and self-contained. It also involves a variety of different documents that span the Document Type Spectrum from highly designed graphical publications to transactional data content.
The Event Calendar Network project was initiated at UC Berkeley to improve the creation and reuse of information about events taking place on campus. At Berkeley, academic units and organizations use a variety of calendars, schedules, and lists of events, and there are scores of different calendars on the Berkeley.edu domain. Each calendar has its own way of describing events, uses different forms for submitting them, and follows different rules governing content, structure, and presentation. Because some events and calendars are of interest to overlapping audiences, many of the events are appropriate for multiple calendars. However, incompatible information models prevent the automated exchange of event information between calendars, so each event must be manually submitted to each calendar.
Despite its university setting, this situation is typical of problems that occur in every large organization with time sheets, expense reimbursement, registration, and other administrative documents. It also represents many business-to-business interactions with incompatible catalogs, product descriptions, and trading terms and conditions.

Just before this book went to press, a group called the Calendar and Scheduling Consortium announced plans to encourage vendors to adopt standard calendar and schedule models so that their applications could share information.²

8.1 UNDERSTANDING DOCUMENT AND PROCESS REQUIREMENTS

Document and process requirements are constraints that must be satisfied for a Document Engineering solution to be considered acceptable. Identifying the correct requirements early in a project makes it less likely that we will have to throw away or rework the analysis, design, or implementation activities. But how do we identify good requirements? There are three basic guidelines:

- Requirements are most often functional: descriptions of what the solution must do or must not do or prevent or enable someone to do. Good requirement rules are expressed using verbs and conditions such as “may,” “must,” and “must not.”

- While requirements can be quantitative or qualitative, they should always be verifiable. For example, it is not much good to define a requirement like “the document model must be robust,” or “the process model must follow relevant standards,” if we have no way of measuring document robustness or determining what process standards are relevant.

- Finally, requirements should not dictate how the solution is to be achieved. That is the responsibility of design.

Requirements should not dictate how the solution is to be achieved
8.1.1 STRATEGIC AND TACTICAL REQUIREMENTS

The Document Engineering approach is suitable for a diverse range of projects. Some projects are tactical or narrow in scope, like deploying a simple web service, automating an existing workflow process involving a business or administrative form, or aggregating a few information sources of the same type. Others are strategic and broad in scope, like the design of an end-to-end information supply chain or the development of service oriented architectures.

Whether a project is more tactical or strategic defines what is or is not possible, how much work it will take, the scale or scope of technology needed to implement a solution, and the likelihood that the project will succeed. It also affects the scope of the requirements gathering process.

A tactical effort, like exposing some existing business functionality as a web service or straight-through processing for a printed form, may appear to have a relatively small number of requirements to consider. It is tempting in these cases just to “get on with it” by creating an XML schema directly from an API or the labeled data entry fields and begin working on the software that will handle document instances, especially if the system is chartered as a proof-of-concept or prototype. Rigorous requirements, conceptual models that capture them, and other intermediate artifacts called for in a Document Engineering approach can seem superfluous.

Even in a tactical project it is important to identify requirements in a disciplined way.

Yet even in an apparently tactical project it is important to make the effort to identify document and process requirements in a disciplined way. In particular, it is critical to determine whether what starts off as a tactical effort is likely to stay that way or may be incrementally burdened with additional strategic requirements. Making it clear what is out of scope at the onset of a tactical project can prevent creeping featurism, which undermines whatever elegance and maintainability the initial design might have.
For example, as businesses increasingly turn to the Internet, a challenging situation often arises when an application meant for internal use only is suddenly expected to operate across organizational or enterprise boundaries or to comply with external standards. Too narrow a view of the initial requirements might make it difficult to function in such broader contexts. We saw this in one of the interoperability scenarios in Chapter 6, with orders from Japan arriving at a bookstore application designed only for U.S. postal addresses. Meeting new requirements will be impossible if the internal system began as a prototype that was gradually hacked into the “real” system, because its informal requirements will be deeply embedded in its programming code.

In contrast, an application designed to be based on an explicit model would be much more able to replace a system that meets internal requirements with one that meets external ones. The extra effort in understanding current and potential requirements and implementing them in a more formal way can prevent cost and schedule overruns, customer support and retention problems, and operational inefficiencies.

The stakes are too high in strategic projects to proceed without a requirements phase

The stakes are simply too high in strategic projects to proceed without a requirements phase, because a failure can severely hamper the company’s ability to carry out its business strategy, cause substantial financial losses, and even cause it to go out of business. In strategic projects like the introduction of ERP or integration with numerous supply chain partners, no one would proceed without a careful effort to identify the document and process requirements. In fact, large companies or organizations often identify and validate their requirements by conducting a contract definition phase or issuing a request for information (RFI) document in which they engage multiple companies or consultants to define the required context of use and come up with some preliminary solution or design concepts.

Nevertheless, because the number of requirements to consider may be almost impossibly large in strategic projects, a different kind of scoping challenge arises. Because the problem being defined might be too large to take on all at once, it is necessary to conduct it in phases. But we can understand how to safely break up a complex problem into simpler parts only if we understand how the parts fit together. So it is essential to understand the dependencies between requirements to identify phases with minimal overlap.
At UC Berkeley, calendar events are similar in many ways to announcements, policies, procedures, and other types of content that are distributed on a regular basis to the campus community. It might have been possible to scope the Event Calendar Network project more broadly to address this general publishing or syndication problem. However, the IT culture at Berkeley is very decentralized, almost by conscious analogy to the autonomy of academic departments, and relatively few enterprise projects are attempted. So the Event Calendar Network team explicitly ruled out designing a general-purpose content distribution system and focused narrowly on the requirements of the calendar domain.

8.1.2 SOURCES OF REQUIREMENTS

The initial motivation for a Document Engineering project can come from almost anywhere—a management task force, an assessment conducted by an external consultant, or a suggestion by a clerk or machine operator to improve a process they struggle with every day. But we can identify some common sources of requirements.

There is no sharp line dividing requirements analysis and document analysis.

It is tautological for us to say that many of the requirements in document-intensive projects are contained in existing documents, and there is no sharp line dividing “requirements analysis,” in which we get requirements from people, and “document analysis,” in which we obtain them from documents. But it is easier to explain the latter if we begin by discussing the former, treating them as separate activities, so we defer the special issues and steps for getting requirements and rules from specific documents to Chapters 11 and 12.

Users, operators, customers, clients, and experts in the domains within the scope of our project will all have useful things to say about what processes an application should or should not carry out and the information they require. Marketing and sales people, because of their relationships with customers and competitors, can also provide essential requirements. Product managers are especially good people to interview about requirements because their job is to assemble and balance the often-con-
flicting perspectives from customers, marketing, and engineering. The executive project sponsor or whoever is funding the effort might not provide technical requirements but will certainly have requirements for an acceptable solution.

Of course, we cannot assume that these people know each other, talk to each other, or even agree with each other. Deciding in advance whose requirements take priority helps resolve conflicts when they arise and helps maintain consistency in designs.

In some start-up situations the problem calls for a truly new system and there will not be any current users from whom to obtain requirements. Nevertheless, we can and should identify intended users and make specific hypotheses about their characteristics, preferences, and capabilities that would determine requirements. Inventing a few of these concrete personas or roles can help prevent assumptions about a single typical user who doesn’t actually exist.4

There are also some less reliable sources of requirements. Software designers and developers often think they are user surrogates, but this is almost never true even if they were once part of the user population. Managers of the users are not good user surrogates, either. People who manage users often think they can speak for them. Who else understands the big picture that the mere users may not grasp? But managers might not know the real problems users face, especially if there are disincentives for revealing those problems. A worker is not likely to tell his boss that he makes errors, lacks essential information, or simply can’t understand the forms he is asked to fill out, even if it is not his fault.

Event calendars at UC Berkeley are used by students, staff, faculty, alumni, and the general public. But because the viability of a calendar network depends on a critical mass of calendars sharing events through a central repository, the calendar administrators were treated as the most important sources of requirements. The Event Calendar Network team set out to define requirements that would meet or exceed the current needs of existing calendars to make it worthwhile for calendar administrators to adopt the new system.
8.1.3 GENERIC REQUIREMENTS

Some requirements apply to almost every Document Engineering situation and might seem obvious. But requirements that are so fundamental that everyone assumes them are precisely those that should be made explicit, because of their importance.

Requirements that are so fundamental that everyone assumes them are precisely those that should be made explicit.

We’ve compiled a list of some of the generic requirements for Document Engineering efforts:

- **Automated information capture.** Eliminate manual entry (or reentry) of information when documents are created, reusing as much as possible from other documents or sources.

- **Straight-through processing.** Minimize the need for any human intervention as a document flows through some specified processes.

- **Timeliness.** Make information available to those who need it when it is needed and when promised, and update it promptly when it changes.

- **Accuracy.** Ensure that every piece of information in a document is correct.

- **Completeness.** Ensure that a document contains all the information it should or that its recipient (person or application) expects.

- **Automated validation.** Provide a schema or specification that enables information to be validated.

- **Interoperability.** Enable information to be used “as is” or via automated transformation by systems or applications other than the one that created it.
• **Standards compliance.** Conform to regulations or standards for information accessibility, availability, security, and privacy.

• **Customizability.** Facilitate the internationalization, localization, extension, and restriction of information.

• **Usability.** Present information in a format or medium that is easy to use and understand by its intended users.

• **Identifiability.** Ensure that the design or appearance of a document signals that it comes from our organization or company; also called branding of the information.

Not all of these requirements apply to every project, and they can be somewhat incompatible with each other. For example, efforts to ensure that a document is accurate and complete can undermine its timeliness, just as emphasizing timely publication might jeopardize accuracy and completeness. The generic requirements that focus on document processing by applications or machines can sometimes conflict with those that emphasize document use by people.

Requirements can be incompatible with each other

Because of these inevitable tradeoffs, it is essential that we understand the perspectives and priorities of the different stakeholders in our project, because they can disagree substantially. Marketing, technical writing, and web design personnel can sometimes be adamant about identifiability and the importance of high production values in documents and user interfaces, but once basic usability is achieved, customers usually care much more about information accuracy and timeliness.

We once saw an engineer almost start a fistfight with a graphic designer when the former learned that the latter had rounded some data values to create a more aesthetic layout with columns of equal width. The engineer said something like “the company has spent millions of dollars getting the product to meet a competitive benchmark and your stupid desire to line up the decimal points just threw it away.”
8.2 CONTEXT AND REQUIREMENTS

The generic requirements listed in the previous section are a convenient checklist when starting a Document Engineering project. But we don’t want a generic solution; we want one that fits the specific situation we face. On the other hand, while the specific situation may be a new one for our organization or company, much of what businesses do can be described using a small set of reusable patterns. We have described organizational, architectural, process, and information patterns. But an even more general way to think of them is as predictable combinations or clusters of requirements. That is, as patterns of context.

Organizational, architectural, process, and information patterns are clusters of requirements

So we could say that the context of a Document Engineering effort is composed of a set of requirements made up of two parts: a part that is specified by contextual patterns, and a part that reflects specific rules that customize or refine these patterns. This situation is just another instance of the Pareto 80/20 principle; in this case it means that we can obtain most of our requirements fairly easily if we can identify one or more appropriate patterns, but identifying the remaining small proportion of our requirements will take most of our effort.

Most requirements can be identified by using patterns, but identifying the rest will take more effort

Figure 8-2 illustrates this idea about patterns as reusable combinations of requirements. The context of use for Context C is covered by a common context pattern defined by Requirement 4, a shared contextual pattern with Context B as defined by Requirement 5 and its own specific Requirement 6.
To put this into practice, let’s consider a generic procurement context that involves a buyer party, a seller party and perhaps a carrier party (to ship the goods). Figure 8-3 depicts this procurement process pattern as a use case diagram.
This generic pattern includes the business processes and roles for the participating businesses. The complete pattern would also include a set of documents, but these are not shown.

Adding requirements to an abstract pattern customizes it to suit our context of use.

In an actual procurement implementation the requirements provided by this pattern will need to be augmented. For example, the processes used to order fuel from Japan will need to follow additional rules. These may differ slightly from the processes of dealing with ordering fuel from Norway, and differ significantly from the processes needed to order tickets for events on the Berkeley Events Calendar. Yet these are all variations of the procurement pattern.

We can apply the notion of context dimensions to organize and analyze these different environments so that we can more easily reuse their sets of requirements and rules that will drive our designs.

The notion of context dimensions helps us more easily reuse sets of requirements.

The most serious attempt to describe contexts for this purpose was that of the ebXML project, which proposed eight dimensions suitable for describing business-to-business global trade. These were Business Process, Product Classification, Industry Classification, Geopolitical, Official Constraints, Business Process Role, Supporting Role and System Capabilities. Using these dimensions to describe a specific context or project domain requires some way to uniquely identify points on each one, sometimes called context drivers (see SIDEBAR).

Not all of the ebXML context dimensions are appropriate in other application domains, but numerous other taxonomies are potentially useful for defining context drivers. For example, when we discussed patterns in business in Section 3.3.1, we mentioned two classification schemas for products, the North American Industry Classification System (NAICS) and the UN/SPSC product and services coding system. To distinguish geographical or regional contexts we might use the ISO 3166 country codes and for more localized contexts (at least in the United States) we could use the FIPS codes or Standard Metropolitan Statistical Areas from the U.S. Census Bureau.
There are also many classification schemes for business models and business organization contexts, such as those proposed by Afuah and Tucci, Timmers, and others.\textsuperscript{7}

\section*{Context in ebXML}

The ebXML project proposed eight context dimensions. These can be envisioned as defining a multidimensional “8-space” or coordinate classification system in which millions of different contexts would be distinguished by their values on each dimension.

We can best explain this idea with an example. Consider an export broker buying aircraft fuel in Japan for shipment to Korea. The documents required in this situation would need information appropriate for contexts such as:\textsuperscript{3}

- Business Process = Procurement
- Product Classification = Aircraft Fuel
- Industry Classification = Petrochemicals
- Geopolitical = Japan
- Official Constraints = Export
- Business Process Role = Buyer
- Business Supporting Role = Intermediary

Each of these values would have associated with it sets of rules that satisfy the requirements for that context dimension, and taken together they would form the set of requirements for the unique situation defined by all eight dimensions. The ebXML architecture further envisions a repository in which these sets of rules are stored and from which they can be retrieved to assemble the document definition needed for any context.\textsuperscript{9}

While the ebXML context dimensions are helpful in describing typical requirement patterns, we aren’t convinced that many contexts in the real world can be distinguished this neatly. For example, contexts are sometimes dependent on each other in complex combinations. Using the example in the sidebar, we may discover that some petrochemical industry requirements or constraints are not applicable in Japan. What if the buyer is also an exporter? Which context takes precedence? Maybe there are requirements that apply only when exporting to Korea, or only to petrochemical exports to Korea.
So while it is important to have some way of organizing requirements for documents and processes that helps us understand their constraints and dependencies, the exact scheme we use isn’t critical. We prefer to take a more heuristic and informal approach to show how context dimensions can aid in understanding patterns of requirements without requiring their rigorous and perhaps overly simplistic, formal classification.

Informal context dimensions can aid in understanding patterns of requirements

Suppose we refine the generic procurement pattern in Figure 8-3 to locate the seller party in another country. We have now created an imported goods procurement context. The use case diagram for this more specific context is shown in Figure 8-4.

![Figure 8-4. The Imported Goods Procurement Pattern](image)

Comparing these two models reveals some additional requirements for the imported goods context. Two new roles of broker and customs are involved because they are dependent on the seller party. In other words, if the seller party is not in the same country as the buyer party, we need additional roles in our process.
In addition to these two new roles, there are also additional transactions and related documents that are dependent on this new context. We now need to arrange customs clearance and have the goods inspected. All these new roles, transactions, and documents are dependent on the primary roles, transactions, and documents. The context of use establishes the dependencies of the business process and exposes the underlying reusable patterns.

It seems easy to describe the Berkeley Event Calendar Network project on some of the ebXML context dimensions, such as:

- Industry Classification = Public University
- Geopolitical = California USA
- Official Constraints = Mandates of open disclosure and free access

These dimensions and values suggest many of the important requirements for the solution and help to control the scope. But the dimension of Product Classification doesn’t offer us any insights about our domain of Calendar Events.

It is also not obvious how to categorize this business process, because both supply chain and content syndication patterns might be appropriate. In Chapter 10 we’ll consider the implications of using either of these two process patterns for our case study.

8.3 EXPRESSING REQUIREMENTS AS RULES

Whether they come from people, documents, or patterns, we must carefully record the requirements we identify as we analyze our context. Some analysts advocate using formal logic languages to enable precise expression of requirements, while others advocate using more natural languages. Because we will ultimately encode most of the requirements in conceptual models of documents and processes and then in XML, we don’t feel a need to be prescriptive about their format in this initial phase. We will use a semiformal style of recording requirements as rules.
Requirements are more useful when expressed as rules

However, we do propose some rigor in categorizing the requirements we identify because it helps ensure that we have a comprehensive set of them. In the following sections we will discuss seven types of requirements that can apply to documents and processes:

- **Usage** requirements define the policies or privileges that govern user access to information or applications.

- **Structural** requirements define co-occurrence or aggregation relationships between components.

- **Semantic** requirements define the meanings of components by specifying properties, dependencies, or roles as generalizations or specializations of other components.

- **Process** requirements define actions to be applied whenever a given condition or set of information is encountered.

- **Syntactic** requirements concern the form in which documents or processes are encoded in a physical or implementation model.

- **Presentational** requirements govern the appearance or rendering of information components.

- **Instance** requirements establish rules or constraints about the values of information components.

These classifications are also compatible with the framework embodied in the Model Matrix. The first three apply at the conceptual end, the next three at the physical column, while instance rules apply to the implementations themselves. This is shown in Figure 8-5.
We should not look for or expect to find requirements in exactly these categories, but we can use them as convenient bins to hold the rules for the requirements that may arise from an interview, document, or some other information source.

8.3.1 Usage Requirements

Usage requirements define the policies or privileges that govern user access to information or applications. Access policy is often governed by roles within an organization, with each category of users having different privileges for viewing or changing information, sometimes at the granularity of a single component.

Role-based access control requirements are usage rules that require organizational roles or responsibilities to be represented in a model that drives an application.\(^\text{12}\)
Externalizing these rules is essential to enable an organization to change policies if needed without reimplementing the applications to which they apply. Examples of role-based access control requirements would be “An employee’s salary can be viewed by a Manager but can only be changed by the Human Resources Department.”

Usage requirements in the Berkeley Event Calendar Network project include controls on who can submit information to calendars. For example, there is a rule that states “Events can be submitted to the central calendar only by a campus-recognized department, unit, organization, or registered student group.”

### 8.3.2 Structural Requirements

Structural requirements define co-occurrence or aggregation relationships between components. Their rules determine the assembly of components into documents.

Structural rules determine the assembly of components into documents

For example, in the context of a repair manual one requirement might be, “it is good practice to include a warning along with any procedure that might be dangerous to the person doing it.” Likewise, following the adage that a picture is worth a thousand words, there might be a rule to “include one or more illustration or diagram that portrays the arrangement or assembly of parts in the device or machine to which the dangerous procedure applies.” Yet another structural rule might state that “a caption is required for each illustration.”

In a procurement context, we might have a rule such as “every order must have an order number, a buyer, and an issue date.” This defines the required aggregation of components. Other structural rules, such as “an order must have at least one order line,” establish both required roles and number of occurrences in the associations between structures.
Structural rules can identify sets of components that are naturally related to each other. A common example would be those that apply to both an order and an order change document. There may be a rule that the structure of these two documents must be identical.

Structural integrity is a requirement for consistent assembly of structures in physical or implementation models, such as requiring identical page boundaries for the electronic and printed versions of documents. Structural integrity is a common requirement when a document exists concurrently in multiple formats, especially when content revisions are highly localized (as in loose-leaf publications with placeholder pages that say “this page intentionally left blank”).

With Event Calendars, some of the requirements were expressed in rules such as:

- “An Event may or may not have a Location.”
- “A Location is of interest only if there is an Event there.”
- “An Event may have a Start Date and either a Duration Period or an End Date.”

We will say more about structural rules when we look at disaggregating structures in Chapter 12 and assembling new structures in Chapter 13.

### 8.3.3 SEMANTIC REQUIREMENTS

As we saw in Chapter 6, the most important requirement in achieving interoperability in a document-centric application is agreement on the meaning of components. Semantic requirements define these meanings.

The most important requirement in achieving interoperability is semantic agreement.

Rules for meanings can be discovered in dictionaries of terms, controlled vocabularies, thesauri, or formal ontologies and are the foundation of any document component model because they must be satisfied by every document instance. For exam-
example, “an order number is an identifier that is unique to the buyer” establishes properties for the component referred to as Order Number. Another type of semantic rule establishes a pattern or generalization for a component. For example, “a buyer is a specific type of party.”

Semantic rules may also expose dependencies, such as “the price of an item can vary depending on the buyer.”

In the Berkeley Event Calendar Network, some of the semantic requirements were given in rules such as:

- “An Event is something that takes place.”
- “An Event is identified by its Title.”
- “Only approved Events can be in the University Calendar.”

We will say more about semantic rules when we look at building a document component model in Chapter 13.

### 8.3.4 PRESENTATION REQUIREMENTS

Presentation requirements govern the appearance or rendering of an information component. Their rules are more common in documents from the publication or narrative end of the Document Type Spectrum because they sometimes have usability or aesthetic implications, which are important to people. For example, “an item description must be on the same page as the product’s image,” might be a presentation rule for a product catalog.

Presentation rules are more common in publication-type documents because their rules are important to people.

Gathering these requirements for technical publications, reports, policies, procedures, reference books, and other nontransactional documents can be difficult. The associations between these components reflect only general principles or best prac-
tices in document markup. Sometimes the rules are specified in style guides, rules, or templates that guide authors when they create instances of these documents.

Presentation integrity is a particularly stringent requirement, often mandated by legislation or contracts, to reproduce a document exactly as it appeared in its original presentation. We see this with International Letters of Credit and Bills of Lading, where we can readily imagine a bank or customs inspector carefully comparing computer-generated and original printed documents. So it is not uncommon in international trading contexts to have rules such as “a Bill of Lading must conform to the UN Layout Key.”

An important presentation requirement that emerged in the Berkeley Event Calendar Network project was the need to generate customized calendars from events in a central repository that reproduced the appearance of existing calendars. Many departments and other campus organizations strive to maintain a distinct visual “brand” for their websites, and an integrated calendar driven by the central repository must be stylistically consistent with its host site.

### 8.3.5 SYNTACTIC REQUIREMENTS

Syntactic requirements concern the language in which documents or processes are encoded for implementation. We saw in Chapter 6 that even documents that conform to the same conceptual model can differ substantially if they don’t use the same encoding syntax or follow the same encoding rules.

In most current Document Engineering efforts, the preferred syntactic requirement is to use XML for implementation. An additional syntactic requirement to use a particular XML vocabulary might also inherit some semantic, content, and structural requirements.
8.3.6 **PROCESS REQUIREMENTS**

Process requirements define actions to be applied whenever a given condition or set of information is encountered. The rules for process requirements are sometimes called behavioral or procedural rules. So when an ordered item is out of stock, for example, a process requirement could be that a customer back order request is created. An example of a process requirement that constrains the scope of the transactions involved is “the procurement process covers all the transactions involved from the requesting of goods until the charging for those goods.” The process rule that defines an explicit transaction would be something like “the seller may respond to an order with an order response.”

Process rules provide the bridge between documents identified in a business transaction and the components needed by those documents.

Process rules can also provide the bridge between documents identified in business transactions and the components needed by those documents. For example, the rule “details of goods that are receipted must be acknowledged” implies that any acknowledgment document must contain details of the goods involved.

In the Berkeley Event Calendar Network project one of the business processes involves parties submitting information about an event to the Public Affairs Department for publication in the central calendar. One of the requirements of this process was given by the rule “an Acknowledgment Receipt is given to the Event Owner for every Event submitted.”

8.3.7 **INSTANCE REQUIREMENTS**

Instance requirements establish rules or constraints about the values of information components. In contrast with the other types of requirements, instance requirements apply to the document instances rather than to the models. A document model might
follow all semantic, presentation, structure, syntax, and processing rules, but any
given instance of the document might fail to satisfy the instance rules.

A document model might follow all semantic, presentation,
structure, syntax and processing rules but the document
itself might fail to satisfy instance rules

Two examples of instance rules that constrain value include “the total value of a sin-
gle order cannot exceed US$1 million,” and “descriptive text must not exceed 80
characters.” Another example, where the value of a component is constrained to a
fixed set of values, is “the currency code should be expressed using ISO 4217 codes.”

Rules for the values of one component can also be dependent on the value of others,
such as “the issue date must be earlier than the delivery date.” There are also com-
plex combinations of instance rules, such as “extended price is quantity multiplied
by price.” This rule not only makes the value of “extended price” dependent on two
other components but also describes the semantics of the component itself. So this is
an instance rule that also contains a semantic one.

Referential integrity is a special instance requirement to keep dependent information
synchronized throughout a document. For example, “all cross references must refer
to valid section identifiers,” or “the number of items on an order must agree with the
stated total.”

Content integrity is another special requirement that seeks to preserve the content
(but not necessarily the presentation) of the original document. This requirement is
especially important in transactional documents that are data-intensive, where it is
essential that values in documents need to be precise and fixed so that every docu-
ment that uses them can use them in exactly the same way. Content integrity is usu-
ally a default requirement, but we include it here because we want to contrast it with
the other kinds of integrity requirements that are not usually imposed by default.

In the Event Calendar Network project, one of the content requirements
was given by the rule “The End Date of an Event must be the same or
later than the Start Date.”
8.4
RULE TYPES AND CONTEXT DIMENSIONS

It is often difficult to decide when the activity of identifying requirements is finished. It can be hard to defend limits on a project’s scope and there is always another person or document to interrogate for potentially important requirements. One method we have used to determine a stopping point for this process is to arrange the requirements we’ve identified according to the rule types and the context dimensions to assess how much of the universe we’ve covered.

We can arrange the requirements according to their types and context to assess how much of the universe we’ve covered

We don’t believe strongly enough in the ebXML context dimensions to suggest that we cannot stop before we find 56 categories of requirements (7 rule types in each of 8 context dimensions) or anything nearly that prescriptive. But it is enlightening to apply whatever model of the context we use to organize and analyze the requirements and rules that will drive our design of processes and their documents.

So let us return to the example from earlier in this chapter of an export broker buying aircraft fuel in Japan for shipment to Korea. Some of the business rules in this situation could be applied to the ebXML context dimensions as shown in Table 8-1.
<table>
<thead>
<tr>
<th>Context Dimension</th>
<th>Type of Rule</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Process = Procurement</td>
<td>Semantic</td>
<td>Every offer must have a unique identification. Every offer must have an acceptance transaction.</td>
</tr>
<tr>
<td>Product Classification = Aircraft Fuel</td>
<td>Structural</td>
<td>An item is identified by both quality and a batch number.</td>
</tr>
<tr>
<td>Industry Classification = Petrochemical</td>
<td>Structural</td>
<td>Hazardous regulations may apply that involve supplementary information components.</td>
</tr>
<tr>
<td>Official Constraints</td>
<td>Process</td>
<td>Certificate of Origin must be supplied on delivery. Bill of Lading must conform to the UN Layout Key.</td>
</tr>
<tr>
<td>Business Process Role = Buyer</td>
<td>Process</td>
<td>Specify party to organize delivery.</td>
</tr>
<tr>
<td>Business Supporting Role = Intermiadiary</td>
<td>Process</td>
<td>Separate Offer and Acceptance transactions are reflected from the ultimate Buyer to the ultimate Seller.</td>
</tr>
</tbody>
</table>

Table 8-1. Examples of Rules Expressing Requirements of Context
The set of rules shown in Table 8-1 is obviously incomplete and perhaps even a little simplistic. But we hope it demonstrates the value of a systematic approach for collecting and organizing the initial requirements for a Document Engineering project.

As we progress through our Document Engineering approach, our analysis will expose more requirements and rules. We will exploit these when we come to design documents to better satisfy our context of use.
8.5 KEY POINTS IN CHAPTER EIGHT

- There is no sharp line dividing requirements analysis and document analysis.
- The stakes are too high in strategic projects to proceed without a requirements phase.
- Even in a tactical project it is important to identify requirements in a disciplined way.
- Requirements that are so fundamental that everyone assumes them are precisely those that should be made explicit.
- Requirements are more useful when expressed as rules.
- Requirements can be incompatible with each other.
- Organizational, architectural, process, and information patterns are clusters of requirements.
- Most requirements can be identified by using patterns, but identifying the rest will take more effort.
- Adding requirements to an abstract pattern customizes it to suit our context of use.
- The most important requirement in achieving interoperability is semantic agreement.
- Structural rules determine the assembly of components into documents.
• Presentational rules are more common in narrative or publication documents because their rules are important to people.

• Process rules provide the bridge between documents identified in a business transaction and the components needed by those documents.

• Even if a document model follows all semantic, presentation, structure, syntax, and processing rules, the document itself might fail to satisfy instance rules.

• We can arrange the requirements according to their types and context to assess how much of the universe we’ve covered.