The XML usage spectrum

Introductory Discussion

- Real-world concepts
- Documents vs. data
- Machine-oriented messaging (MOM)
- People-oriented publishing (POP)

Chapter 3

Like the Jets and the Sharks, the factions never mixed.[1] On the one hand there were the document-heads armed with word processors and formatters. On the other, the data-heads from the relational database world. It's time for a truce – no, an alliance. XML finally makes clear an internal truth: documents and data are the same thing. To be precise, documents are the interchangeable form of data!

3.1 | Is XML for documents or for data?

What is a document?

The dictionary says:

> "Something written, inscribed, engraved, etc., which provides evidence or information or serves as a record".

1. Depending on your cultural proclivities, these are either athletic teams or the rival gangs in *West Side Story*.

Documents come in all shapes and sizes and media, as you can see in Figure 3-1. Here are some you may have encountered:

- Long documents: books, manuals, product specifications
- Broadsides: catalog sheets, posters, notices
- Forms: registration, application, etc.
- Letters: email, memos
- Records: “Acme Co., Part# 732, reverse widget, $32.50, 5323 in stock”
- Messages: “job complete”, “update accepted”

An e-commerce transaction, such as a purchase, might involve several of these. A buyer could start by sending several documents to a vendor:

- Covering note: a letter
- Purchase order: a form
- Attached product specification: a long document

The vendor might respond with several more documents:

- Formal acknowledgment: a message
- Thank you note: a letter
- Invoice: a form

The beauty of XML is that the same software can process all of this diversity. Whatever you can do with one kind of document you can do with all the others. The only time you need additional tools is when you want to do different kinds of things – not when you want to work with different kinds of documents.

And there are lots of things that you can do.

3.2 | A wide spectrum of application opportunities

Sorry about that, we’ve been reading too many marketing brochures. But it’s true, nevertheless.

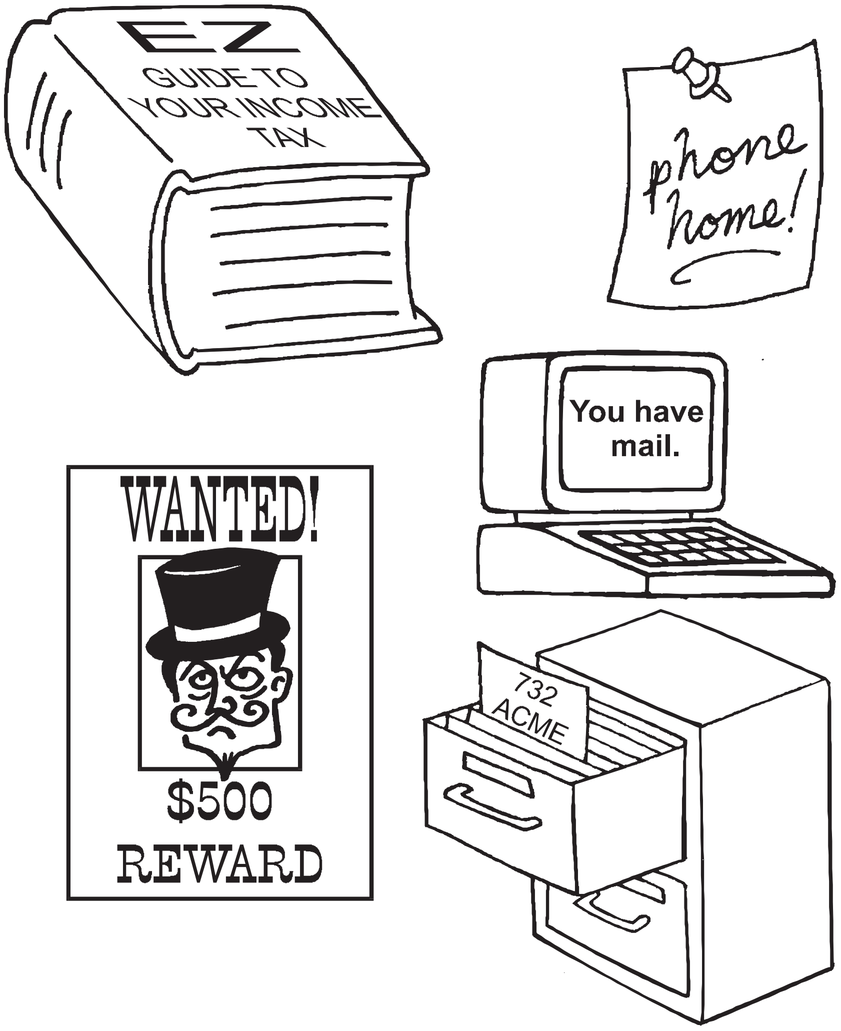

Figure 3-1 Documents come in all shapes and sizes.

At one end of the spectrum we have the grand old man of generalized markup, *POP* – People-Oriented Publishing. You can see him in Figure 3-2.

At the other end of the spectrum is that darling of the data processors, *MOM* – Machine-Oriented Messaging. She smiles radiantly from Figure 3-3.

Let's take a closer look at both of them.

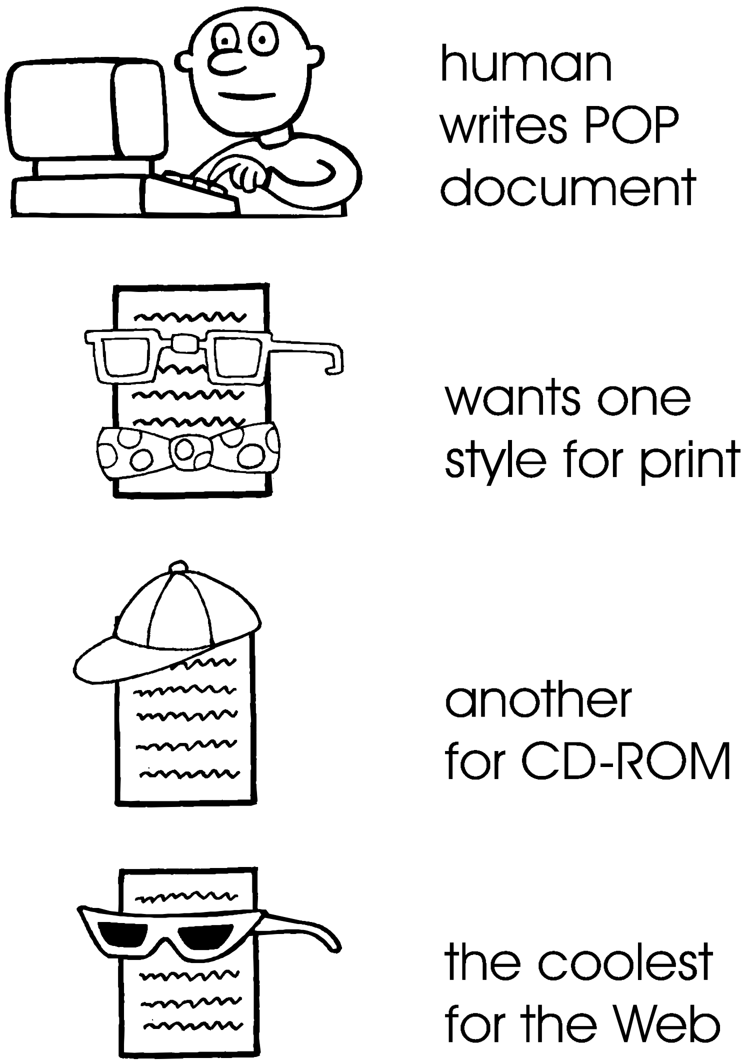

Figure 3-2 POP application.

3.2.1 *People-oriented publishing*

POP was the original killer app for SGML, XML's parent, because it saves so much money for enterprises with Web-sized document collections.

POP documents are chiefly written by humans for other humans to read.

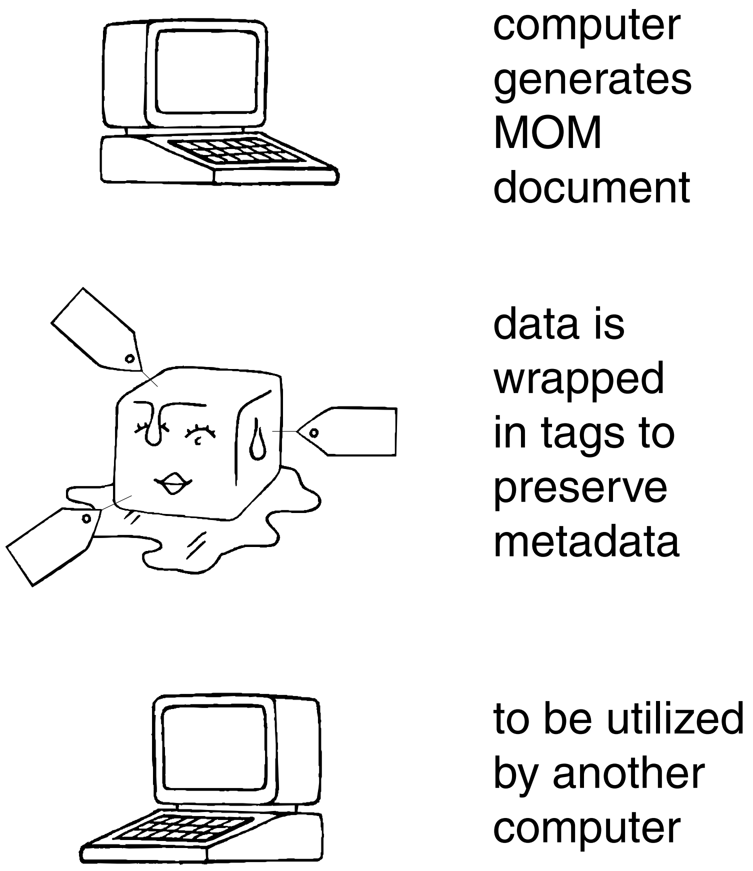
computer generates MOM document
data is wrapped in tags to preserve metadata
to be utilized by another computer

Figure 3-3 MOM application.

Instead of creating formatted renditions, as in word processors or desktop publishing programs, XML POP users create unformatted abstractions. That means the document file captures what is *in* the document, but not how it is supposed to look.

To get the desired look, the POP user creates a stylesheet, a set of commands that tell a program how to format (and/or otherwise process) the document. The power of XML in this regard is that you don't need to choose just one look – you can have a separate stylesheet for every purpose.[2] At a minimum, you might want one for print, one for CD-ROM, and another for a website.

POP documents tend to be (but needn't be) long-lived, large, and with complex structures. When delivered in electronic media, they may be interactive. How they will be rendered is of great importance, but, because XML is used, the rendition information can be – and is – kept distinct from the abstract data.

3.2.2 *Machine-oriented messaging*

MOM is the killer app – actually, a technology that drives lots of killer apps – for XML on the Web.

The software that processes the messages is called *middleware*.[3] As you might suspect from the name, it is software that comes between two other programs. It acts like your interpreter/guide might if you were to visit someplace where you couldn't speak the language and had no idea of the local customs. It talks in the native tongue, using the native customs, and translates the native replies – the messages – into your language.

MOM documents are chiefly generated by programs for other programs to read.

Instead of writing specialized programs (clients) to access particular databases or other data sources (servers), XML MOM users break the old two-tier client/server model. They introduce a third tier, the "middle tier", that acts as a data integrator. The middle-tier server does all the talking to the data sources and sends their messages to the client as XML documents.

That means the client can read data from anywhere, but only has to understand data that is in XML documents. The XML markup provides the *metadata* – information about the data – that was in the original data source schema, like the database table name and field names (also called "cell" or "column" names).

The MOM user typically doesn't care much about rendition. He *does* care, though, about extracting the original data accurately and making

2. We know that all office suites offer some degree of stylesheet support today, but XML (well, GML) did it first, and still is the only way to do it cleanly.
3. In fact the MOM acronym has long been used in the industry – and in previous editions of this book – to mean *message-oriented middleware*. We prefer the new meaning because it puts the emphasis on the data, rather than the software, some of which still hasn't caught on to using XML.

some use of the metadata. His client software, instead of having a specialized module for each data source, has a single "XML parser" module. The parser is the program that separates the markup from the data, just as it does in POP applications.

And just like POP applications, there can be a stylesheet – a set of commands that tell a program how to process the document. It may not look much like a POP stylesheet – it might look more like a script or program – but it performs the same function. And, as with POP stylesheets, there can be different MOM stylesheets for different document types, or to do different things with message documents of a single document type.

There is an extra benefit to XML three-tier MOM applications in a networked environment. For many applications, the middle-tier server can collect all of the relevant data at once and send it in a single document to the client. Further querying, sorting, and other processing can then take place solely on the client system. That not only cuts down Web traffic and overhead, but it vastly improves the end-user's perceived performance and his satisfaction with the experience.

MOM documents tend to be (but needn't be) short-lived, non-interactive, small, and with simple structures.

3.3 | Opposites are attracted

To XML, that is!

How is it that XML can be optimal for two such apparently extreme opposites as MOM and POP? The answer is, the two are not really different where it counts.

In both cases, we start with abstract information. For POP, it comes from a human author's head. For MOM, it comes from a database. But either way, the abstract data is marked up with tags and becomes a document.

Here is a terminally cute mnemonic for this very important relationship:

Data + Markup = DocuMent

Aren't you sorry you read it? Now you'll never forget it.

But XML "DocuMents" are special. An application can use three different processing techniques with one:

- *Parse it*, in order to extract the original data. This can be done without information loss because XML represents both metadata and data, and it lets you keep the abstractions distinct from rendition information. Once extracted, the data can be manipulated as needed by the application.
- *Render it*, so it can be presented in a physical medium that a human can perceive. It can be rendered in many different ways, for delivery in multiple media such as screen displays, print, Braille, spoken word, and so on.
- *Hack it*, meaning "process it as plain text without parsing". Hacking might involve cutting and pasting into other XML documents, or scanning the text to get some information from it without doing a real parse.

The important revelation here is that data and documents aren't opposites. Far from it – they are actually two states of the same information.

The real difference between the two is this:

- When data is in a database, the metadata about its structure and meaning (the schema) is stored according to the proprietary architecture of the database.
- When data is in a document, the metadata is stored as markup.

A mixture of markup and data must be governed by the rules of some *notation*. XML and SGML are notations, as are RTF and Word file format. The rules of the notation determine how a parser will interpret the document text to separate the data from the markup.

Notations are not just for complete documents. There are also *data object notations*, such as GIF, TIFF, and EPS, that are used to represent such things as graphics, video (e.g., MPEG), and audio (e.g., MP3). Document notations usually allow their documents to contain data objects, such as pictures, that are in the objects' own data object notations.

Data object notations are usually (not always) in *binary*; that is, they are built-up from low-level ones and zeros. Document notations, however, are frequently *character-based*. XML is character-based, which is why it can be hacked.[4]

Since databases and documents are really the same, and MOM and POP applications both use XML documents, there are lots of opportunities for synergy.

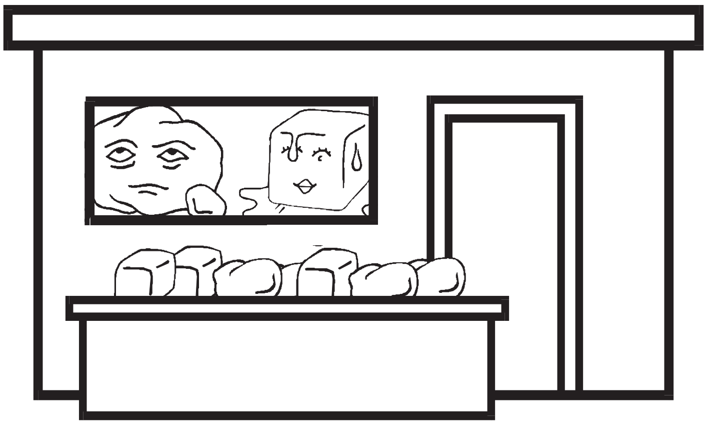

Figure 3-4 Dynamic servers: The MOM and POP store.

3.4 | MOM and POP – They're so great together!

Classically, MOM and POP were radically different kinds of applications, each doing things its own way with different technologies and mental models. But POP applications frequently need to include database data in their document content – think of an automotive maintenance manual that has to get the accurate part numbers from a database.

Similarly, MOM applications need to include human-written components. When the dealer asks for price and availability of the automotive parts you need, the display might include a description as well.

---

4. In fact, a design objective was to support the *desperate Perl hacker* – someone in a hurry who writes a script to scan XML without parsing it. Note that the term "hacker" had none of the "cracker" stigma implied by the popular press. The only security compromised by the desperate Perl hacker is his job security, for leaving things to the last minute!

With the advent of generalized markup, the barriers to doing MOM-like things in POP applications began to disappear. Some of the POP-like applications you'll read about later in the book appear to have invented the middle tier on their own. And now, with the advent of XML, MOM applications can easily incorporate POP functionality as well.

What is now emerging is a new generation of composite systems, dynamically serving both persistent POP information and dynamic MOM data. They use databases to store information components so they can be controlled, managed, and assembled into end-products in the same way as components of automobiles, aircraft, or other complex devices. Think of them as the MOM and POP store (Figure 3-4).[5]

In fact, we'd go so far as to say there is no longer a difference in kind between the two, only a difference in degree. There really is "an endless spectrum of application opportunities". It is a multi-dimensional spectrum where applications need not be implemented differently just because they process different document types. The real differentiators are other document characteristics, like persistence, size, interactivity, structural complexity, percentage of human-written content, and the importance of eventual presentation to humans.[6]

At the extremes, some applications may call for specialized (or optimized) techniques, but the broad central universe of applications can all be implemented similarly. Much of the knowledge that POP application developers have acquired over the years is now applicable to MOM applications, and vice versa. Keep that in mind as you read the application descriptions and case studies.

That cross-fertilization is true of products and their underlying technologies as well. All of the tool discussions in this book should be of interest, whether you think of your applications as chiefly being MOM or being POP. It is the differences in functionality and design that should cause you to choose one product over another, not their marketing thrust or apparent orientation. As you read the tool category discussions, you'll find remarkable similarities among tools that were developed for entirely different purposes.

5. Generations ago the Mom and Pop store (grocery, convenience, etc.) was the achievement of the entrepreneurial couple who'd lifted themselves out of the working class. Today they'd have an e-commerce website!
6. The relationship between documents and data is explored further in 7.11, "Documents and data", on page 152.

3.5 | Conclusion

We've covered the key concepts of XML itself in previous chapters and the spectrum of its uses in this one. Along the way we've discovered that:

- Documents are the interchangeable form of data
- MOM data is (usually!) made by machines
- POP data is (usually!) made by people
- MOM apps deliver MOM (and/or POP!) data to machines
- POP apps deliver POP (and/or MOM!) data to people

In the next two chapters, we'll look at the key application concepts at both ends of the spectrum.