XML Documents in the Enterprise — How to Make the Switch

Abstract

Authoring documents in XML can have significant benefits to an organisation. This talk examines those benefits and what they could mean to an organisation. It also discusses a difficult problem many people face when they decide to switch to XML - what to do with legacy documents. A novel technique for dealing with this problem will be discussed.

Introduction

There is no doubt that XML has paved the way for a number of exciting applications that may dramatically change the way business is conducted. It is also true, however, that some existing business processes may benefit from the application of XML technology.

This talk focuses on one such process - the production and storage of documents within an organisation. In some ways, this is returning to the "roots" of XML, since SGML and similar languages were conceived with the idea of making document creation and processing more efficient. Though the ideas have been around for a long time, the technique is rapidly becoming more cost-effective. This is largely due to the interest in XML and the resulting increase in the number of tools available for authoring and storage of XML documents.

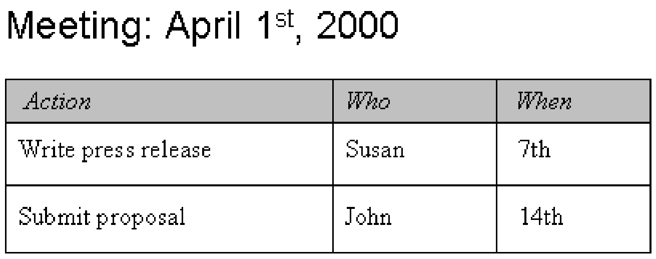
Obviously the decision to change the way documents are created cannot be taken lightly, and various issues will need to be considered. This talk will attempt to address some of the more obvious questions that may be asked:

- what are the benefits I can expect from using XML to create documents?
- which documents are the best candidates for being created in XML?
- how do I cope with my existing documents which are not in XML?

An Example

To demonstrate various ways of creating documents, we'll consider a hypothetical example. Let's say that our organisation has a policy that at every meeting a document containing the minutes must be produced. This document must be sent to all attendees, and saved somewhere for future reference.

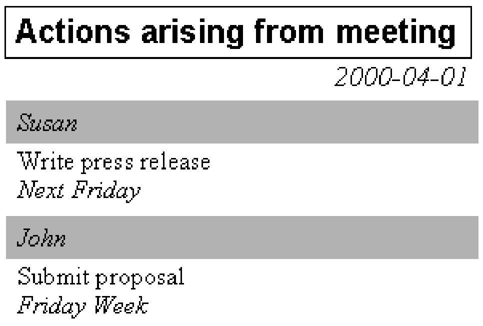
Given this scenario, we can assume that an author might use a word processor to produce a document such as the following:



Meeting: April 1st, 2000

Action	*Who*	*When*
Write press release	Susan	7th
Submit proposal	John	14th

(greatly simplified, of course). Another author might produce a document like this:



Actions arising from meeting

2000-04-01

Susan

Write press release
Next Friday

John

Submit proposal
Friday Week

Although the two documents look very different, they both fulfill their intended function well. In both cases it is possible to quickly determine that they record the results of a meeting, and the individual action points are clearly identified by the formatting.

Now let's further assume that we have collected a large number of these documents describing meetings over time, and that we wish to construct a database from the action points. Since there are many documents, it is desirable to set up some automatic process to collect the data. This presents us with a number of problems:

- There is probably no naming convention that allows us to easily locate the documents. Trying to identify the documents by content is difficult (what do the above two documents have in common?).
- Having located the documents, it is difficult to extract the action points. We would need to try and enumerate all of the different presentation styles used by authors.
- Extracting some of the data needs a global understanding of the content. Note that the same date appears in the above documents as "14th" and "Friday Week".

If the above document were stored in XML, it might look like this:

```
<?xml version="1.0"?>
<!DOCTYPE minutes SYSTEM "minutes.dtd">
<minutes>
<date>2000-04-01</date>
<action>
  <who>Susan</who>
  <what>Write press release</what>
  <when>2000-07-01</when>
</action>
<action>
  <who>John</who>
  <what>Submit proposal</what>
  <when>2000-14-01</when>
</action>
</minutes>
```

This immediately addresses the above processing concerns:

- The DOCTYPE directive identifies the type of data contained in the document.

- The action points are clearly identified.
- The dates in the action points are normalised.

Of course, we have lost the formatting of the information which made it easy to read, but it is possible to employ tools which apply formatting to the XML data. In fact, it would be possible to allow the user to select the presentation style they preferred. Consider how difficult it would be to convert between the two presentation styles used in the word processor documents above.

Analysis of Document Types

What is most interesting in the above example is that it is easy for a human reader to interpret the word processing documents, but very difficult to extract information from them automatically. This is because a great deal of knowledge is required to understand the documents. We have no difficulty understanding the documents because we know what a meeting is, we know that it leads to action points and we know what information is required to define an action point. The presentation of the word processing document assists us by suggesting how to apply this knowledge, but that can only be helpful if we already have the appropriate knowledge.

From the above we can generalise one of the most important differences between a word processing document and an XML document:

- A word processing document does not directly store knowledge relating to the content; rather it uses presentation to allow the reader to infer this knowledge.
- An XML document directly contains (some) knowledge relating to the content, but does not store any presentation information.

The above example shows us that we can automatically generate presentation information from the knowledge stored in an XML document, but we cannot automatically generate knowledge from the presentation information in a word processing document. This leads to the following very important observation:

An XML document stores more information than a word processing document.

This is probably the key to deciding whether a document is best created with a word processor or an XML authoring tool. An XML document is more expensive to create (because it contains more information), but can be exploited in more ways than a word processing document.

The Benefits of XML

The above conclusions allow us to characterise the documents that are best created by a particular method.

A document can benefit from being created and stored in XML if, at some point in its life, any of the following are true:

- some form of automatic processing is done on its content
- it is searched for a particular part of its content (as opposed to being accessed in full)
- it must be presented in a number of different formats

In addition, it is worth considering the expected lifetime of the document. A word processing document is only useful while you have access to the program that created it, or one that is compatible enough to read it faithfully. Since application software typically has a high turnover rate, documents with a long life are best stored in a non-proprietary, open format (i.e. XML).

All of the above points depend in some respect on the lifetime of the document. Business requirements are changing rapidly in response to the availability of new technologies, with the result that it is increasingly hard to predict how any given piece of information will need to be accessed. In other words, the longer a document is deemed to contain useful information, the more likely the flexibility provided by XML will produce a benefit.

In fact, the single most important characteristic when deciding how to create a document is probably its *durability*. Criteria such as complexity and the value of the stored information are only indirectly important, since they usually extend the anticipated lifetime of the document.

The Problem of Legacy Data

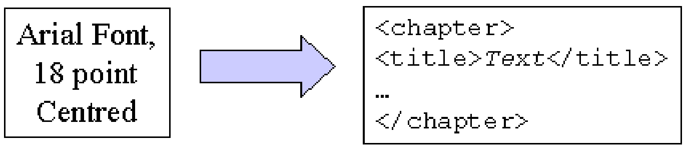
For an organisation, deciding whether it is cost-effective to create certain documents in XML is usually not difficult. The benefits can be defined and the costs can be accurately estimated. Much more difficult is the decision as to how to deal with existing (legacy) documents. A short-term strategy may be to only create new documents in XML, but most organisations will find they can only realise the full benefits of XML if both new and archival documents are processed in the same way.

Deciding what to do with legacy documents is often difficult because:

- The volume of legacy documents is likely to be large.
- The documents may be stored in a number of formats. For some of the older formats it may no longer be possible to run the associated application software.
- Unless there is a comprehensive catalogue for the documents, it may be difficult to estimate the value of the information they contain.

If the decision is to convert the documents *en masse*, there are two basic approaches. They can be manually re-authored using the legacy documents as source material, or an automated conversion tool can be developed. The latter will typically use the presentation information to infer knowledge, just as a human reader does.

To infer knowledge from presentation, it is usual to construct a number of rules which identify patterns in the presentation. Each pattern maps to a structure in the XML document. For example, a paragraph in 18 point Arial font which is centred horizontally may indicate the heading of a chapter. The following diagram shows how this pattern maps to a fragment of XML data:



```
<chapter>
<title>Text</title>
…
</chapter>
```

Both of the approaches mentioned above will inevitably be labour-intensive, and therefore expensive and prone to error. As we demonstrated above, the transition from word processing document to XML document involves the acquisition of knowledge. In the absence of cost-effective artificial intelligence, this means that any automated process must be heavily supplemented by human involvement to achieve an acceptable result.

Interactive Conversion - A Compromise

An automatic conversion from a word processing document to XML will typically run to completion and then report any errors that occurred. The operator must then decide whether to correct the output, or change the source document and run the conversion again.

An alternative approach is to acknowledge that operator intervention will be required, and integrate it into the conversion at the most appropriate time. The converter applies "rules" to transform presentation information into XML tags. When it encounters data for which it cannot find a rule, it pauses and asks the operator to intervene. The operator can choose to:

- Add a new rule, or modify an existing rule, to handle this situation.
- Modify the source document so that it matches an existing rule.
- Manually enter the correct data into the output document.

The last choice above is significant, since it means that the rules don't have to handle every possible situation encountered in the data. Being able to handle rare cases manually can greatly simplify the rules, and therefore make the converter much cheaper to implement.

The most interesting aspect of an interactive converter is that it opens the possibility of doing just-in-time conversions. If the tool is made available to those who need access to archival documents, they can choose to do the conversion when they identify a need for a particular document. Since many organisations discover that only a small fraction of archival documents are actually used, this can lead to a very cost-effective solution.

Conclusion

XML won't make word processors obsolete - they are the best and cheapest way of creating certain classes of documents, and probably will remain so for some time. However, there are many cases where authoring in XML has the potential to reduce costs and/or significantly increase the value of documents.

There is no simple way to switch to XML, although new tools and techniques have appeared recently which make it much less daunting. Now is a good time to consider how it might fit into your enterprise.