



Sinch Precedents Automation Conference 2006

Precedent documents – Increasing value and reducing costs

26 October 2006

Peter Meyer, Managing director, Elkera Pty Ltd

Elkera Pty Limited ABN 68 092 447 428
Suite 701, 10 Help Street, Chatswood, NSW 2067, Australia
PO Box 5280, West Chatswood, NSW 1515
Telephone +61 2 8440 6999
Facsimile +61 2 8440 6988
www.elkera.com

Contents

Introduction.....	1
What are precedents?	1
How are precedents managed today?	2
Precedents are a lot of work.....	2
Precedent documents.....	2
Variable codes and conditional logic.....	2
Clause libraries	3
Problems	3
Use of word processing formats.....	3
Proprietary codes for document assembly systems.....	4
Blobs.....	4
Wouldn't it be nice if	5
If it is achievable, why is it not commonplace?	5
Lessons from other content domains	6
What do other publishers do?.....	6
What is structured content?.....	6
Some recent developments	7
Benefits.....	8

Precedent documents – Increasing value and reducing costs

By Peter Meyer, Managing director, Elkera Pty Ltd
Email: pmeyer@elkera.com.au
Web: www.elkera.com

Presentation given at Sinch Precedents Automation Conference, Sydney, 26 October 2006 under the title “Extracting more value from precedents and reducing maintenance costs”.

Introduction

Lawyers create documents and publish them to their clients and other interest holders. At one level, law firms could be viewed as document factories. In this context, precedents are of critical importance.

Lawyers work predominantly in word processing software. Once WordPerfect was supreme. Today, most lawyers use Microsoft Word to create and publish documents.

In most firms, precedents are prepared and managed using the prevailing word processing technology, currently Word. This has obvious attractions. A lawyer using the corresponding software can pick up a precedent, copy it and start editing it to create a working document.

Unfortunately, this model also creates quite a few problems. This presentation examines some of these problems, considers some of the possible improvements that could be made and provides a brief insight into some of the technologies that might be involved.

What are precedents?

Precedents are documents or components of documents we store for re-use in future transactions. Some common characteristics of precedents include:

- They are usually derived from real transaction documents but a lot of work is required to create really useful precedents.
- Thus, there may be different classes or quality levels of precedents, according to how flexible and trustworthy they are regarded by their users.
- Precedents may be complete documents that a lawyer will take and modify to a particular set of instructions. In other words, a starting point.
- They may be complete documents that normally are not modified except by the addition of specific transaction data such as names, addresses, money amounts,

dates etc. This data may be added manually or it may be inserted in an automated variables substitution process.

- Precedents may be complex documents with alternative or optional components that are inserted or suppressed according to conditions set in a user interface in a document assembly programme or by values extracted from a database. Conditional processing may be very complex so that software applications can be used by non lawyers can answer questions and provide information to create documents.
- Precedents may be components (clauses) that lawyers can locate from a clause library and insert into new documents or they may be accessed only through a document assembly programs.

Precedents are essential to the efficient operation of a law firm. They enable less experienced practitioners to draw upon the skill and knowledge of more experienced practitioners. They enable the firm to provide services rapidly and cost effectively. They are a critical component in reducing risk.

How are precedents managed today?

Precedents are a lot of work

Precedents development involves a very large amount of work. There is work to ensure that the drafting is accurate, clear and consistent with firm drafting styles. There is work providing annotations or explanations to users. There is work making sure the formatting is consistent with firm styles. There is a lot of work adding metadata so that practitioners can find and use the precedents. There is work adding codes and other markup needed by automated processing systems. There is testing and checking and re-working as the law and clients change.

Precedents development and maintenance involves a lot of time and expense with input from practitioners and dedicated precedents managers.

Precedent documents

Whole documents are almost invariably maintained in the format of the current editing software. The documents are frequently stored in a document management or content management system to facilitate access control and retrieval. Precedents may be discovered through text search or through metadata associated with the precedents.

Variable codes and conditional logic

If precedents are accessed through document assembly software to automate variables substitution, codes or other markup must be added to suit the particular processing

software. Most applications define their own system of codes and markup although some may work with codes created by competitor products. Some may use XML based on proprietary schema.

Clause libraries

Larger firms in particular may maintain large clause libraries. As discrete objects, clauses in a clause library present special problems. Since the clause can find its way into many different documents and at different levels within those documents, it can create problems if the user has to re-format them each time they are used so that automatic numbering and display style is correct.

For this reason, some clause libraries hold blobs of plain text that is then formatted after insertion in a working document.

Problems

Use of word processing formats

The use of word processing formats presents many problems:

- Word processing formats store display style information with the content. If you want to change document styles, it is usually necessary to edit every document to make the changes, unless only simple style property changes are involved. On a large database, this can be a massive, costly exercise. Styles in the documents are never completely consistent so attempts to automate the process rarely achieve the expected results. Inconsistency arises from human error or from drift over time as tastes or file formats change. New documents may not be the same as older documents in the database so that a database can hold many different styles and file formats, making automated changes almost impossible.
- The use of word processing formats results in massive duplication of content in precedent systems. It is very difficult to store a clause once and have it re-used in multiple documents. Many document families, particularly in the finance area replicate boilerplate clauses. Often maintenance involves changing the same wording in many places.
- Word processing formats cause particular problems in clause library systems. How do you format a clause that may be inserted at level 1, level 2 or level 3 of a numbered hierarchy? If formatting is removed, how can it be added reliably after the clause is inserted into the document. The choice is to either manually change styles or provide macros to assist. Either way, the process is very inefficient for the practitioner because it involves unnecessary work.

- Periodically, software vendors release new versions of their software and change the file formats. Built in conversion filters don't always work perfectly. Moving documents forward to the new software version can be costly and disruptive.
- If documents are not migrated, they may be lost because they are either incompatible with the new software or they don't display correctly and fall into disuse. Over time, a lot of valuable knowledge can be lost or effort wasted re-creating it.
- It is difficult to create different outputs for word processing documents, eg, on the web, particularly if it is desired to add links between different documents. Manual processing is almost always required.

What is the ongoing cost of maintenance as a result of these problems? What costs are imposed through lost practitioner time?

Proprietary codes for document assembly systems

Every document assembly application defines its own system for codes and other document markup needed to drive conditional processing. There are no standards, although, over time, de facto standards may emerge. However, these are unstable.

The use of proprietary markup creates both a massive barrier to the adoption of a new system that uses different markup and a strong tie to the existing system. There is a great deal of work in applying this markup. It makes acquiring a new system or switching very expensive.

What is the cost of staying with an outdated or inefficient system because the cost of change seems too high?

Blobs

If precedents are stored as whole documents, how does a practitioner find and use a single clause from that document. Using word processing file formats, there is no convenient way to store metadata on each clause to facilitate retrieval. Users must rely on search. The reality is, to extract an individual clause, they must trawl through large documents to work out what is in them and then manually cut, copy, paste and re-format the content.

How much wasted practitioner time does this create? How do you monitor usage of this kind of precedent content and how do you charge for it?

Wouldn't it be nice if ...

Each and every one of these problems can be overcome. The solution is fundamentally the same for all:

- What if you could completely change the layout of your documents created from precedents as often as you wished without having to touch a single precedent?
- What if every document created from a precedent was formatted exactly to firm styles? Practitioners would not need to worry about creating contents listings or cover pages or running macros to do so.
- What if you could adopt new editing software that uses a new file format, again without having to touch a single precedent?
- Wouldn't you like to be able to publish precedent content in print and web formats completely automatically? Web content could be linked to related and explanatory materials without manual intervention.
- What if there was no distinction between clauses in the clause library and those in complete documents? Practitioners could search clauses based on metadata and content regardless where they occur. Once found, an object could be copied out and inserted into a new document without manual processing or formatting by the practitioner.
- What if you could create boilerplate clauses and share them in dozens of ready to use documents without duplication?
- Wouldn't you like to be able to much more easily introduce or change document assembly software because it would not be necessary to change the markup in existing precedents to suit.
- Finally, what if these things could be done without changing the software used by the lawyers? They could keep on using their existing word processing software, if they so wish.

What would be the savings and other benefits if these processes were standard procedure?

If it is achievable, why is it not commonplace?

There is no doubt that all the “wouldn't it be nice” scenarios could become reality. However, it would require a substantial change to existing approaches in the legal technology area. It is true that some document assembly system vendors already use the sort of technology that makes this possible. Fundamentally, the reason it is not widespread is that legal documents have some very distinctive characteristics. There are very few systems available that can readily achieve these goals and there are no

applicable standards. Where would a law firm start if it wanted to completely change its precedents storage model? It would have to build almost everything from scratch.

Lessons from other content domains

What do other publishers do?

In many other industries, the equivalent problems have been or are being addressed.

Commercial publishers, particularly those involved in legal publishing adopted structured content management strategies 10 or more years ago to enable them to more effectively meet the demands of users for content in different formats and with richer features.

Many legislative drafting agencies (Parliamentary Counsel Offices) have adopted structured content management strategies for the preparation, management and publishing of legislation. Three Australian offices have done so, despite the absence of formal standards. New Zealand is doing so at present. It is occurring increasingly in other countries. The use of structured content management strategies enables the Parliamentary Counsel Offices to maintain legislation indefinitely, independently of particular editing and publishing tools. It also enables them to automate publishing systems and create high quality print and web publications.

For many years, organizations involved in producing very large volumes of complex documentation for equipment, particularly in aero-space and the military, have used structured content for effective content management and publishing. More recently, this trend has begun to filter down to very many organizations that produce product and user assistance documentation for equipment, business processes and software. This is being aided by the development of improved standards for structured content, particularly the Darwin Information Typing Architecture (DITA) released as an OASIS standard in May 2004. It is available at: www.oasis-open.org/specs/index.php#ditav1.0.

The DITA standard seeks to achieve for product documentation many of the benefits sought for legal precedent documents described above. However, there are some important differences between product documentation and legal documentation.

What is structured content?

Structured content is content that is described in a detailed way using metadata. When applied to narrative content such as precedents, structured content generally refers to the use of XML tagging to describe the content and the relationship between individual components.

XML is a meta language used to build domain specific markup languages or grammars. These languages are described by a schema. Common forms of schema include

Document Type Definition (DTD) and XML Schema. There are very many XML languages for different kinds of documents.

Commonly available schema can be divided into several different XML markup models:

- (a) Rich, presentation based schema such as the Microsoft Office Open XML format and OASIS OpenDocument;
- (b) Web presentation schema such as XHTML 1.0;
- (c) Generic structural markup schema such as DocBook, DITA, TEI, Elker BNML and possibly XHTML 2.0 that can be applied to a wide range of documents.

These XML languages are each designed to do very different things and are very different in the way they work. It is critical to be aware that representing something as “XML” does not, on its own, mean anything. The presentation oriented XML languages such as Microsoft Office Open XML in Word, OpenDocument or XHTML are of limited value to long term precedents maintenance. They do not provide a reliable representation of the important objects within a legal document, such as a clause, section or even a paragraph. They do not effectively separate presentation from content. They are satisfactory for the production of documents with arbitrary layout. They are only marginally better than conventional word processing formats for the reliable automatic production of large numbers of documents that conform to standard layouts.

The schema most suited to achieving the goals described above are generic structural schema. These schema can completely describe documents and their components so that no formatting information needs to be stored. It is then possible to automatically transform and style generic structural XML content into any other output. Structured schema provide the most robust model for long term data storage and reliable automation of content re-use and publishing processes. The encapsulation of semantic objects within structured schema enables metadata to be reliably associated with content objects at any level to support greatly improved information retrieval.

Some recent developments

For around the past four years, the OASIS LegalXML, eContracts Technical Committee has been developing an XML schema for contract documents. Fundamentally, the purpose of this schema is to address the problems and provide the “what if” benefits described earlier.

The eContracts schema is a generic structural XML schema. It is the first standard schema designed specifically for contract narrative markup. Although it is specifically designed for contracts, it is straight forward to adapt it to most other legal documentation.

The eContracts TC is currently finalising the documentation or specification for its contracts schema. It is possible that the TC will release its committee specification for public review and comment before the end of 2006.

The eContracts schema has the potential to change the landscape of precedents management. It provides a platform on which law firms, beginning with the larger firms, can build to achieve all the “what if” scenarios described earlier.

Materials for the eContracts TC are available at: www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalxml-econtracts.

Benefits

Ultimately, it is up to the legal community to decide to build on the work of the OASIS LegalXML eContracts TC. Should it do so, it could achieve all the “wouldn't it be nice” scenarios described earlier. The result would include:

- substantially lower costs of maintenance for precedents databases over long periods;
- improved access to precedents and associated explanatory materials by lawyers, leading to reduced risks of error;
- reduced costs of precedent usage by practitioners through more efficient access;
- improved document assembly systems that could be based on the standard legal documentation schema;
- greater flexibility to choose document assembly tools, including to use different tools for different jobs.