JYVÄSKYLÄ LICENTIATE THESES IN COMPUTING
16

Eliisa Jauhiainen

Deployment of XML for Office Documents in Organizations

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ LICENTIATE THESES IN COMPUTING 16

Eliisa Jauhiainen

Deployment of XML for Office Documents in Organizations

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2014

Deployment of XML for Office Documents in Organizations

JYVÄSKYLÄ LICENTIATE THESES IN COMPUTING 16

Eliisa Jauhiainen

Deployment of XML for Office Documents in Organizations

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2014

Editor
Mauri Leppänen
Department of Computer Science and Information Systems, University of Jyväskylä

URN:ISBN:978-951-39-5600-4
ISBN 978-951-39-5600-4 (PDF)

ISBN 978-951-39-5599-1 (nid.)
ISSN 1795-9713

Jyväskylä University Printing House, Jyväskylä 2014

ABSTRACT

Jauhiainen, Eliisa
Deployment of XML for office documents in organizations
Jyväskylä: University of Jyväskylä, 2014, 63 p. (+ four included articles)
(Jyväskylä Licentiate Theses in Computing
ISSN 1795-9713; 16)
ISBN 978-951-39-5599-1 (nid.), 978-951-39-5600-4 (PDF)
Licentiate Thesis

Majority of the content in organizations is stored as documents. Structured documents, like XML documents, allow the structure definitions, document instances, and layout specifications to be handled as separate entities. This is an important feature to realize from a document management point of view. A class of similar documents with the same structure constitutes a document type. The documents are built from components that are logical units of information within the context of the document type.

Office documents are typically authored using word-processing software, they are relatively short in length, and intended for human consumption. The development of open office standards brought XML to organizations' office environments and changed the capabilities of using document content in ways that were previously impossible or difficult. The aim of this research is to explore possible benefits of using custom XML schemas in office documents and factors influencing the design of such schemas.

This study reports findings from two action research studies. Both cases involved designing custom XML schemas for office documents with the case organizations. The research resulted in the increased understanding of real-life requirements and challenges of XML schema design for office documents and provided insights of XML-based office document use after implementation.

The research area changed significantly since the study was started. When the research began in 2005, the open XML standards for office documents were still under development, yet office suites available had already XML support enabling the use of custom XML schemas. The longevity of the study has provided an opportunity to closely observe the use of an XML-based office document authoring system using custom XML schemas in a case organization through the years. The contributions of this study are the insights to schema design efforts. The two cases revealed issues motivating organizations to consider custom XML schemas for their office documents and how XML document management can be analyzed and described. The use of schema design methods was observed to be beneficial in both cases.

Keywords: document management, document analysis, schema design, office document

Author's address

Eliisa Jauhiainen
Department of Computer Science and Information Systems
University of Jyväskylä, Finland

Supervisors

Airi Salminen
Department of Computer Science and Information Systems
University of Jyväskylä, Finland

Anne Honkaranta
Digia Plc
Jyväskylä, Finland

Reviewers

Jari Multisilta
CICERO Learning
University of Helsinki, Finland

Tero Päivärinta
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology, Sweden

ACKNOWLEDGEMENTS

I would like to thank both of my supervisors, Professor Airi Salminen and Anne Honkaranta, for their insightfulness, guidance and patience. It's been a long road, but luckily filled with light and many opportunities to grow. I would also like to express my warmest thanks to the reviewers of this thesis, Jari Multisilta and Tero Päivärinta, for their contribution and insightful comments.

This thesis marks an end of an interesting journey. Luckily I have shared it with the best travelling companions possible, a group of very near and dear ones in the Department of Information Systems and Computer Science. The ladies known as the Tricksters (i.e. "huiputtajat") - Maritta Pirhonen, Minna Silvennoinen, Irja Kankaanpää, and Marjo Silvennoinen - all of you made this road more fun to travel. Thanks for the laughs, shoulders to cry on, sympathetic ears, and many great discussions. Sharing thoughts, ideas and a wide spectrum of feelings with you on research and life in general has been both enlightening and empowering experience for me. I would also like to thank Reija Nurmeksela, my "partner in crime". Writing research articles with you is always a pleasure. Your insightfulness never ceases to amaze me.

Some people enjoy afternoon tea at five o'clock, but Seija Paananen and I enjoy our seven o'clock coffee in the morning. Seija is without a doubt the heart of the Department of Information Systems and Computer Science, and one of the most significant supporters for me throughout these years. Your positivity is truly inspiring!

I would also like to express my gratitude to the hardworking people in the Faculty Office. If I have done anything to make your everyday work even a little bit easier, I'll take that as one of the most important accomplishment of these past years. Don't ever forget how important and valuable work you do for the faculty staff every day.

Jyväskylä 31.5.2013
Eliisa Jauhiainen

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF INCLUDED ARTICLES

1 Honkaranta, A., Jauhiainen, E. (2007). Two Methods for Schema Design for Intelligent XML Documents in Organization. In Witold Abramowicz, Heinrich C. Mayr (Eds.). Technologies for Business Information Systems. Dordrecht, Netherlands: Springer, 173-182.

2 Jauhiainen, E., Honkaranta, A. (2007). Aspects on XML Document Content Reuse in Organizations. In Weide Chang, James B.D. Joshi (Eds.). Proceedings of the 2007 IEEE International Conference on Information Reuse and Integration (IEEE IRI-07), Las Vegas, 588-593.

3 Nurmeksela, R., Jauhiainen, E., Salminen, A., Honkaranta, A. (2007). XML document implementation: Experiences from three cases. In Youakim Badr, Richard Chbeir, Pit Pichappan (Eds.). Proceedings of the Second International Conference on Digital Information Management. Los Alamitos, CA: IEEE, 224-229.

4 Salminen, A., Nurmeksela, R. and Jauhiainen, E. (2013). A Life Cycle Model of XML Documents. Submitted to the Journal of the American Society for Information Science and Technology (JASIST). Accepted October 2, 2013.

1 XML FOR OFFICE DOCUMENTS

A document is a unit of recorded information meant for human consumption (Levien, 1989). Today a great deal of documents are digitally created, stored and displayed. Documents are used as information carriers between people and software modules, and they are produced and exchanged in organizational processes (Sprague, 1995). According to Klischewksi (2006), documents function as representation of organization's administrative knowledge. In this thesis an *office document* has the following characteristics:

- It is composed with office suite software (typically word-processing software).
- It is relatively short in length, or assembled from documents that are relatively short in length (in contrast to book-like documents like theses, manuals, etc.).
- Most, if not all, of the document content consists of natural language (i.e. written text).
- A person authors it.
- It is targeted for human consumption.
- It functions as a record in an administrational process or business transaction.
- It may have an organization-specific document template.
- It may receive some of its content from an external content source, such as database.

XML (Extensible Markup Language) (Bray, Paoli, Sperberg-McQueen, Maler and Yergeau, 2008) has been introduced to office documents via standardization of open document formats. Using XML-based file formats instead of application dependent binary formats introduced office environments as a new area for XML research.

1.1 XML documents

XML is a restricted form of SGML (Standard Generalized Markup Language) (Goldfarb, 1990), which is a metalanguage for *structured documents.* In structured documents structure definitions, document instances and layout specifications can be separated from each other and therefore processed separately. The roots of SGML are in publishing industry where the need to establish generic typesetting codes for document manipulation in different text processing systems emerged in the late 1960s (Fierz and Grütter, 2000). The most widely used and known SGML-based language is HTML (HyperText Markup Language), the markup language for web pages. The rules of SGML were, however, too complicated to define languages needed in Internet communication.

XML was developed to meet the needs of representing information of various application domains on the Internet. A design goal of XML was to be simpler than SGML, yet compatible with it. Unlike with HTML, which provides a fixed set of element types to use, the users of XML may create and name element types of their own. The names may not necessarily indicate anything about document presentation, but about the information of document content instead.

Every XML document has a logical structure and physical structure. The physical structure consists of one or many entities. On the logical level an XML document is composed of declarations, elements, comments, character references, and processing instructions, which are indicated by explicit XML markup. Each XML document contains elements, which are delimited by start and end tags. Elements may contain other elements, text, or a mixture of both. XML documents can be perceived as hierarchical tree-structured collections of nodes where each node of the tree corresponds to an element in the XML document.

An XML document begins in the root of a document. The element tree branches to the "leaves", which consist of character data. Figure 1 illustrates a simplified example of an element tree where the rectangles represent individual elements. The root of the document tree is the root element `Memo`. Therefore, a memo document starts with the `<Memo>` tag and ends with the `</Memo>` tag. The lines between the rectangles illustrate the child-parent relationships between the elements. The element `Contents`, for instance, contains child elements `Paragraph`, `Motion` and `Decision`. These elements do not have child elements of their own and therefore they may contain the actual text content of the memo document, i.e. "the leaves". Elements can also have attributes. For example, the element `Memo` may have an attribute `date` containing the date information of the memo document. An XML document is *well-formed* if it meets all of the well-formedness constraints given in the XML Recommendation. XML processor takes care of inspecting if a document is well-formed.

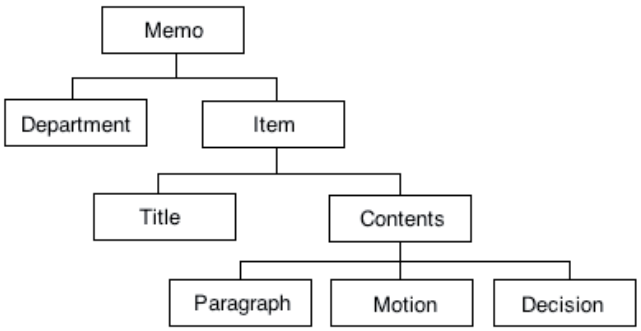

FIGURE 1 XML element tree

A schema is a set of rules describing the structure and other constraints for a class of XML documents. A class of similar documents by their content and purpose is considered as a *document type*. Typical document types in offices are, for instance, memos, reports and invoices. If an XML document has an associated schema and the constraints defined in the schema are also met, the XML document is *valid*.

There are a number of languages developed to express schemas. DTD (Document Type Definition) is defined in the XML specification and provides capabilities to define a grammar for a class of XML documents. DTDs describe the structure of a class of documents via element and attribute-list declarations. XML Schema (Fallside and Walmsley, 2004), also known as XSD, was introduced in 2001 and unlike DTD, XSD is declared by using XML-based syntax. RELAX NG (Regular Language for XML Next Generation) (Clark and Murata, 2001) is a schema language developed by a working group at the OASIS organization and it is considered to be the challenger of XSD. A major goal of RELAX NG is that it should be easy to learn and easy to use (Clark, 2001). RELAX NG has both XML-based syntax and another, more compact syntax, which is not based on XML.

When documents are authored in XML form, document content consists of the XML markup and character data. It is always readable by XML software. Marked up document can also be easily transformed to other formats such as HTML (HyperText Markup Language) or PDF (Portable Document Format). XSLT (The Extensible Stylesheet Language Transformations) (Clark, 1999) is a language for expressing human perceivable transformations. The external representation is often defined by a style sheet associated with the document.

1.2 Data-centric vs. document-centric XML

"XML shall support a wide variety of applications", is one of the design goals written in the XML specification (Bray et al., 2008). As XML has spread to multiple types of application domains, the variety of different kinds of XML docu-

ments is wide. In literature a typical way to classify XML documents is to divide them into data-centric and document-centric XML documents, based on their purpose and type of use. Office documents may have both data- and document-centric properties. Table 1 summarizes the common characteristics of data- and document-centric XML documents identified in XML research.

TABLE 1 Summary of the characteristics of XML documents

Characteristic	DATA-CENTRIC	DOCUMENT-CENTRIC
Typical use	Data exchange, data format in databases	Electronic publishing, document content reuse
Schema characteristics	Use of attributes and data types	Number of elements is greater than number of attributes, mixed element content, deep document hierarchies
Element order	Unimportant	Significant
Typical research areas	Data integration, transfer format between databases	Document standardization, structured/modular writing, e- publishing, information retrieval

Data-centric XML documents are common, for instance, in the field of e-business where data exchange between systems enables efficient business transactions. XML can also be used to manage XML data in relational and object-oriented, or object-relational databases (Elmasri, Wu, Hojabri, Li and Fu, 2002). Another example domain is web service messaging, which is machine-to-machine interaction over a network. Messages are usually XML documents that follow the SOAP (Simple Object Access Protocol) (Mitra and Lafon, 2007) standard.

Typically data-centric XML documents are regular in structure and homogeneous in content (Bertino and Catania, 2001). Elements in data-centric XML documents typically contain either character data or other elements, but not a mixture of both. The order of sibling elements is not necessarily important as long as the character data is correctly marked up. The use of attributes in order to provide additional information about elements may be common as well as constraining element content with data types. For example, an element `price` may be defined to contain numerical data, like integer, instead of textual data. Data-centric XML documents are also referred to as *transactional documents* (Glushko and McGrath, 2005), *record-like documents* (Harold and Means, 2004), or *database-oriented documents* (Megginson, 1998). Office documents like invoices, for instance, may have content that is retrieved from an external system like a database or a web service. Such content is more data- than document-centric.

Document-centric XML documents are meant for human consumption and therefore there is typically a layout attached to them. Typical examples of document-centric XML documents are books, manuals, and journal articles. In technical documentation XML may be used in *structured writing*, which involves

developing categories of document content that can be "single sourced" or re-used in various documents. Benefits of single sourcing result in more consistent layouts as well as to the possibility of multiple document authors to work in parallel (Rockley, 2001). Single sourcing approach has also been utilized for modular e-learning content, which involves XML-based document authoring (Bubenik, Hanke, & Juhnke, 2005). A great deal of office documents lean heavily towards this end of an XML document spectrum. Documents like letters, memos, and reports may have more document-centric properties than data-centric ones.

Commonly document-centric XML documents are a mixture of content and layout components, such as paragraphs, headers, lists, and formatting. (DuCharme, 2004) Document-centric XML documents, unlike their data-centric counterparts, are strict on the element order within a document type, i.e. the order of sibling elements is significant (Nambiar, Lacroix, Bressan, Lee, and Li, 2002). Significant amount of document content consists on natural language. Document-centric XML documents are also called as *narrative documents* (Glushko and McGrath, 2005), *narrative-like documents* (Harold and Means, 2004) or *book-oriented documents* (Megginson, 1998).

1.3 XML document components

XML document structures can be perceived from different perspectives. In general, XML documents have both logical and physical structures. Physically XML is built from units called *entities*. Entities are storage units of content and they are identified by an entity name. Each XML document has at least one entity, which is called *document entity*. Document entity may contain the whole XML document. Logical structure of an XML document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit mark-up.

As explained earlier, a set of XML documents similar by their arrangement of document can be defined as a document type. The ISO 8879 defines a document type as a class of documents that have similar characteristics (ISO, 1986). For example, memorandum is a typical document type in any organization. Therefore, memorandum's document instances can be, for instance, "a memo of Monday's meeting" and "a memo of Thursday's meeting".

Documents of a type are built from document *components*. In other words, components are building blocks of the document types. A component is a part of the document containing unique information compared to other parts within the same document. Where the line of uniqueness goes may be determined in document analysis, during which document components are identified. An important characteristic of a component may also be its size. The size of a component depends on *granularity*, which refers to the level of detail of a component. (Rockley, Kostur and Manning, 2002) For example, there could be a "Meeting information" component identified from memorandum document type. Whether this component ends up as a single element in an XML schema for the

memorandum depends on granularity. Meeting information could be considered as a single element, or alternatively as a container element, which has, for example, child elements `Date` and `Location`.

The concept of a component is not unambiguous, because both the structure as well as the use of components can be perceived from different perspectives. Rockley et al. (2002) see components as reoccurring pieces of information in document types and granularity determines the smallest piece of information (i.e. component) that is reusable. Granularity can change within a document and therefore components do not define how documents should be authored; they simply define how content within documents is handled with XML markup. For example, a memorandum document type may have a reusable component *Item* that occurs in all memo instances.

Maler and El Andaloussi (1996) classify document components into three kinds; content-based components, structural components, and presentational components. C*ontent-based components* contain information that has specific internal organization, but the appearance of the information may vary. For example, address information usually indicates what it contains; it has a specific internal structure that consists of a street name, zip code, country name, etc. However, in different document instances the appearance of address information may vary.

S*tructural components* rely on the presentational traditions of print-based publishing. For example, paragraphs and lists can be seen as structural components. Their structure is explicit, but nothing is known about the contents of the components. Therefore, structural components can be seen domain independent – they occur similarly regardless of a domain document types belong to.

Presentational components are bound to the visual formatting of a document. They describe how information should appear in published documents. For example, there may be phrases or regions of text in a document, which are formatted with special font or size.

According to Maler and El Andaloussi (1996), schema designers interpret components in any of the three ways, which can lead to the identification of different components. Therefore, schema designers must define components carefully and make the definitions explicit. Table 2 provides alternative interpretations for four components.

TABLE 2 Interpretations of components identified from memo document type

Content-based	**Structural**	**Presentational**
Title	Heading	Bolded, font-size 12 pt
Item	Paragraphs	Font alignment always justified, font-size 10 pt.
Motion	Paragraph or list	The word "Motion" on its own line, bolded
Decision	Paragraph or list	The word "Decision" on its own line, bolded

Megginson (1998) uses the term *logical unit* when referring to components. His classification of components is based on their use. Depending on the complexity of a document type, components in a DTD fall into following three categories:

- components that document authors must use to create useful documents
- components that are not mandatory, but are visible to authors nevertheless, and
- components that are not required by the document authors nor are visible to them.

There are also other terms used to refer to components. For example, a concept of *Learning Object* refers a component in e-learning contexts. According to Polsani (2003), the Learning Object contains an idea or a self-standing portion of text or multimedia. Also terms *content block, content item* and *content units* are used (Boiko, 2002; Hackos, 2002). According to Hackos (2002), content units describe chunks of content that are used to build document types. This study adopts the term component. The term component is also used by Maler and El Andaloussi (1996), Rockley et al. (2002), and Fiala, Hinz and Wehner (2003).

A document type may be formally defined in a schema. On the level of XML markup this means that the elements, attributes and their allowed occurrence and ordering are identical in all document instances of the same type. Identifying and understanding components is important for both schema designers and document authors. Same information may be marked up in many different ways; with new elements, with existing elements made available in new contexts, or with attribute values, for instance. The attribute values and novel contexts of elements are part of understanding schemas properly.

1.4 Document management in offices

The personal computer revolution in early 1980s altered drastically work conducted in organizations' offices. This technical cataclysm resulted also in the birth of a new research area known as office automation. According to Olson and Lucas (1982), the term *office automation* referred to the use of integrated computer and communications systems to support administrative procedures in an office environment. Office automation was expected to improve the productivity of office personnel. From technical point of view such systems were based on individual workstations connected to a local network in an office.

Text processing capabilities of automated office systems provided features for handing documents electronically. Word processing software encoded documents in a binary form at computers to process and render. Documents like this were strictly vendor dependent and if the word processing software was changed, there was no certainty of the future accessibility of the information in the documents as the capabilities for processing document content in other software were limited.

When the amount of electronic documents increased, the management of such documents became a relevant issue for organizations. In the late 1980s

structured documents and their authoring tools were mostly SGML-oriented. The research carried out at the time resulted, for example, to a few SGML-based document production tools for structure documents (Quint and Vatton, 2004). A significant change in document management in offices took place early 2000's as the nature of the most current office applications gradually changed. Common office suites, like OpenOffice.org and Microsoft Office suite, began to use XML as the native data format of documents. XML was seen to have many benefits over binary formats, such as smaller document file sizes, better data recovery, and better integration possibilities with other documents as well as business information stored in external data sources. (Garfinkel and Migletz, 2009; Petride, Tarachandani, Agarwal and Idicula, 2011) XML provided a mechanism for both systems and people to identify different parts of documents. This was a significant change from document management perspective as documents were earlier considered as sets of unstructured data objects (Blumberg and Atre, 2003). More recent studies have focused on *interactive office documents*, for instance. Boyer (2008) has examined how interactive office documents may evolve as tools for creating content on Web 2.0 applications. Office documents on such applications are perceived as a client side of a web application, providing a familiar interface for document authors.

In the following the typical characteristics of an XML-based office document management environment are described within the model presented in (Salminen, 2005) for content management environments (see Figure 2).

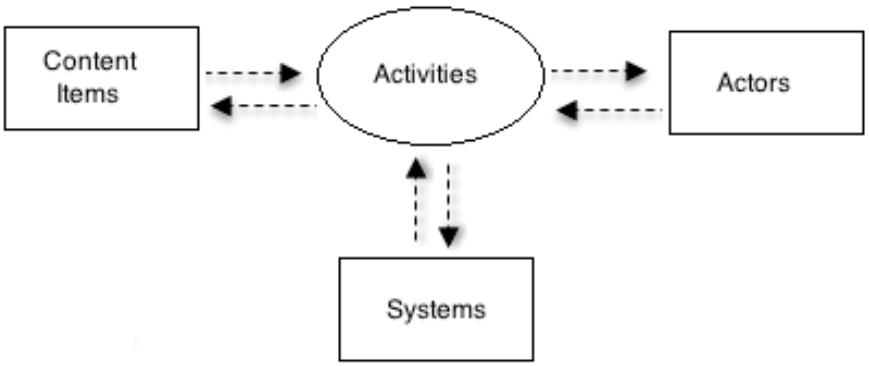

FIGURE 2 Components of a content management environment (Salminen, 2005, p.5)

The model contains two types of entities: activities depicted by an ellipse and resources depicted by rectangles. An activity is a set of actions performed by one or more actors in a process. The resources are divided into three types, content items, actors, and systems. The arrows represent information flow between the activities and resources.

From the viewpoint of this thesis, relevant *activities* are document production and XML schema design. *Content items* are created and used in activities. Document templates and individual document instances created on them are all relevant content units. If XML is deployed, relevant content items may also include schemas, XML document instances and style sheets. *Actors* are

people and organizations performing activities. In office document management document authors are relevant actors on the domain. Document creation may be a collaborative effort of many authors. Also schema designers are relevant actors on the domain, if custom XML schemas are designed in order to deploy XML-based document production.

Systems are used in document management activities. For example, office automation aimed to improve activities related to document production by introducing new systems, like electronic document management systems, in office environments. Nowadays XML and open standards may also be seen as part of relevant systems used in office document management. According to Lappin (2010), new systems and applications, however, may not make the old ones disappear. For example, new open document formats have introduced XML in office document management, yet they do not necessarily ensure the new format would automatically replace older formats entirely. If XML is deployed, relevant systems also include XML and its related standards.

1.5 Standard schemas for XML documents

Despite of the standardization of open formats for office documents, research carried out of the deployment of XML in organizations' offices has been scarce. This chapter presents the main characteristics of the three standard schemas for office documents. DocBook schema (Walsh, 2008) is a standard schema for publishing structured documentation. Open Document Format (ODF) (ISO, 2006) and Office Open XML (OOXML) (ISO, 2008) are both standards for office document formats.

1.5.1 DocBook

DocBook (OASIS, 2006) is a markup language that was created in the early 90's for documentation authored with a document processing system called *Troff*. (Walsh and Muellner, 1999) The first versions of DocBook schemas were SGML DTDs. Nowadays XML DTD is still the normative DocBook schema, but the schema is also available in XSD and RELAX NG schema languages.

According to Megginson (1998), DocBook was designed to encompass many different structures inside a set of book-like document types. The root element of a DocBook document is typically `book`, or `article`. There are different types of chapters and sections defined in the schema, such as glossaries, synopses, listings, and footnotes. Sections consist of paragraph-level elements (Walsh and Hamilton, 2010).

Figure 3 shows an example of a DocBook document. The root element of the document is `book`. The document consists of a title and one chapter, which has one section. The section contains a formal paragraph with title and element `para` for the paragraph content. The paragraph is followed by the list, which has two list items. The element representing a list item is `member`.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE book SYSTEM "U:\jyunet\protoilu\docbook-5.0\dtd\docbook.dtd">
<book annotations="this is a simple example of an XML document">
   <title>DocBook Example</title>
   <chapter xmlns="http://docbook.org/ns/docbook">
      <title>Chapter 1</title>
      <section>
         <title>Section 1</title>
         <formalpara version="1">
               <title>About this chapter</title>
               <para>This paragraph is the first paragraph in the first chapter of this book.
                     Below this paragraph is a list with two list items.</para>
         </formalpara>
         <simplelist>
               <member>This is the first list item</member>
               <member>This is the second list item</member>
         </simplelist>
      </section>
   </chapter>
</book>
```

FIGURE 3 XML document conforming to the DocBook DTD

In the DocBook schema each element declaration is associated with attribute declarations. There are 19 common attributes defined, including the attributes `annotations` and `version`, which are also illustrated in Figure 3.

The elements defined in the DocBook schema are general enough to be used in many contexts. Elements like `para` and `simplelist`, for instance, do not bind the markup with any specific context of use. The elements are equally applicable for technical manuals and e-learning content, among others. For example, DocBook schema has been deployed in the context of e-learning online course content supporting multiple output formats of the content. (Molloy, 2003; Martínez-Ortiz, Moreno-Ger, Sierra, and Fernández-Manjón, 2006.) As an open standard DocBook schema is not bound to any specific authoring tool.

1.5.2 OpenDocument Format

The OpenDocument Format (ISO, 2006), also known as ODF, is an ISO standard for XML-based office document. An OpenDocument file consists of a set of XML documents and associated binary data within a zip package. The XML documents in the OpenDocument zip package are:

- content.xml
- meta.xml
- settings.xml
- styles.xml, and
- manifest.xml.

content.xml contains the actual document content, excluding the binary data like images. meta.xml contains the metadata of the document, like creation date and editing cycles, for instance. settings.xml contains information about the document authoring tool settings. For example, information of view settings can be found from settings.xml. styles.xml contains most of the document styling information, though font style declarations and automatic office styles are also included in content.xml. Finally, manifest.xml contains information of all the XML documents and other files within the zip package it is a part of as well. Overall, there are three types of RELAX NG schemas in the ODF standard:

1. OpenDocument schema,
2. normative schema for the manifest.xml, and
3. strict schema for office documents.

The OpenDocument schema is the main schema of the ODF specification. It is defined for text documents, charts, spreadsheets, and graphical documents. The OpenDocument schema is extensive and relatively complex as it contains all the necessary definitions for text documents to spreadsheets and charts. The normative schema for manifest.xml defines necessary elements for manifest.xml. The strict schema is used when OpenOffice document is stored as one XML document instead of set of multiple XML documents in an OpenDocument package. However, this method is not supported in OpenOffice directly.

1.5.3 The Office Open XML

Office Open XML (OOXML) (ISO, 2008) is another XML-based file format for office documents. Microsoft originally developed the format for word-processing documents, spreadsheets, charts, and presentations. Therefore, the OOXML standard includes the following three markup languages:

- WordProcessingML for text and graphics in MS Word document,
- SpreadsheetML for information in a MS Excel workbook, and
- PresentationML for presentation related data in MS PowerPoint slides.

The schemas in OOXML specification are declared in both XSD and RELAX NG schema languages. The schema that defines WordProcessingML is called the WML schema (wml.xsd). WordProcessingML document is a set of separate XML documents and other necessary files inside a Zip package and the XML documents in a Zip package conform to WML schema. For example, the following XML documents may be found in a package of a WordProcessingML document:

- document.xml,
- styles.xml,
- settings.xml,
- fontTable.xml,
- app.xml,

- core.xml, and
- [Content_Types].xml.

Primary document content is stored in document.xml. Document's primary content (written text) appears within sections and its presentational features are controlled by section's properties. For example, each section can have its own headers and footers. (ISO, 2008) styles.xml contains the styling information, like fonts used in a document. settings.xml contains all the document's properties. fontTable.xml contains the fonts used in a document. When a Word-ProcessingML document is displayed on computer screen, information in fontTable.xml is used to determine which fonts to use, if the original fonts are not available (ISO, 2008). app.xml contains application-specific properties of a document like the amount of pages and paragraphs in a document. Finally, [Content_Types].xml contains a dictionary with content types for all the other parts inside the Zip package.

1.5.4 Comparison of the standards

The DocBook schema differs significantly from the OpenDocument schema in ODF standard, and from the WML schema in OOXML standard. XML markup is mainly structural; there are elements for book-like document types such as sections and paragraphs. Document authors may use a syntax-directed XML editor to author documents and therefore they should have know-how about XML and the document structure DocBook schema defines.

The ODF and OOXML standard schemas provide necessary definitions and declarations for documents. The purpose of the XML markup in this context is to store and describe document content (i.e. text), document metadata, application settings data and presentational information of documents (ISO, 2006; ISO, 2008). All this information is described within XML elements, which are defined in an open XML schema. For document's primary content there are elements for text paragraphs and lists, for instance, but on the level of markup there are no means to tell whether a text in a paragraph is associated with a document type of memo or report. Document authors have freedom to compose any kind of documents they please. Therefore, these two standard schemas are not meant for specific document types, but to store documents in XML format. Table 3 summarizes the characteristics of the three open standard schemas.

TABLE 3 Comparison of standards

	DOCBOOK	ODF	OOXML
Schema	Schema for book-like document types, books and articles. Elements for sections, chapters, paragraphs and inline-elements Normative schema is an XML DTD schema.	Schema for a text document (not for specific document type). Elements for text content like headings, text paragraphs, lists, and table. Normative schema is a RELAX NG schema.	Schema for a text document (not for specific document type). Elements for paragraphs, runs and texts. Each paragraph and run is associated with properties. No separate elements for headings, and lists. Normative schema is a XSD schema.
Document authoring	Content written with a syntax-directed XML editor with immediate validation against the DocBook schema. XML markup describes document structure, not presentation.	Document content is created and modified with an authoring tool providing a WYSIWYG interface for document authoring. XML markup is not visible for document authors. No means to add or remove specific elements and/or attributes defined in a schema. Modifying XML markup necessitates editing XML documents inside a document package manually. The authoring tool generates XML markup during document authoring.	

Where DocBook schema defines a document type, like a book or an article, OpenDocument schema in ODF standard and WML schema in OOXML standard are defined for the document format. Therefore, document author may not even be aware of the schemas whereas with DocBook the schema guides document author to produce valid documentation.

1.6 Office document standardization

Problems in document management may involve an insufficient way of reuse document content. When XML was first introduced to MS Word in the Office 2003 version, a specific kind of document template called *SmartDocs* was introduced alongside other templates. The idea of SmartDocs was to support collaborative document production and efficient content reuse via *snippets*, which were document files containing reusable document. This indicates that document content reuse in office documents is commonly an essential characteristic of office document management. Reusable document content may be searched

from the set of old documents, copied and pasted into new novel ones, which can be a time-consuming task.

If there are multiple document authors involved in the production of same document types, similar content may be created simultaneously and saved as separate documents. This may lead to a situation where there are many different versions available of the same document. Rockley et al. (2002) call a phenomenon of this sort as *content silo trap*, which can be problematic from the document management viewpoint. Computer storage space, time and money are wasted, when the amount of similar and potentially unnecessary document types and instances increases.

To solve aforementioned kinds of problems in office environments, standardization of document types is required. Document standardization involves agreeing upon the rules how information is clustered and presented in documents as well as practices involved in document management (Nurmeksela, Jauhiainen, Salminen and Honkaranta, 2007). Document standardization is not about identifying documents and their structures only, but also identifying other entities of the document environment. (Salminen et al., 2000).

With OpenOffice.org and Microsoft Office applications it is possible to design document templates for different document types. Document templates provide a unified layout and structure for documents and XML-based file format supports integration to other data sources. In addition, relying on macros, for instance, can provide means to control document structures of office documents. This approach may be, however, expensive to develop and prone to errors. (Sefton, 2007) Furthermore, XML-based standards for office documents support the accessibility to document content in future, but these standards do not, however, necessarily change or improve activities related to document authoring.

Defining custom-made XML schemas may support retrieving document content from an external system as well as multi-channel publishing of office documents. The design and deployment of custom XML schemas is a document management initiative based on the needs and requirements of organization in contrast to adopting standards.

The benefits from the deployment of custom-made XML schemas in offices are not, however, well known. According to Scifleet and Williams (2009), designing digital documents have remained unexamined or under-theorized research area. Design of documents has often been viewed as a technological challenge, even though in practice schema designers are faced with the complexity of the domain including work tasks related to document production, document life cycles, and the people who produce them. Therefore, more research is needed to understand benefits and challenges of XML deployment in offices.

This thesis focuses on the design and deployment of XML for office documents. The rest of the thesis is organized as follows. Chapter 2 introduces the research goal and methodology of the study. Schema design methods are introduced in Chapter 3. Chapter 4 provides an overview of the joint articles of the thesis by summarizing the research objectives, methods, content, and results for each article. Chapter 5 presents the results of this study and Chapter 6 concludes the thesis.

2 RESEARCH GOAL AND METHODOLOGY

In this chapter the research goal, research methodology and the research process of the thesis are introduced, and the cases involved are described. The MemoX project was carried out in the Faculty of Information Technology in the University of Jyväskylä and it resulted in an XML-based system for Faculty Council agenda production. The RAKE project was a feasibility study at the Finnish Centre of Pensions (FCP) concerning administrative documents. Both cases involved designing custom XML schemas for office documents.

2.1 Research objectives

Research on document-centric XML documents has covered technical documentation (Broberg, 2004) and modular e-learning content (Bubenik, Hanke, and Juhnke, 2005). Also XML standardization initiatives in e-government cases have been reported (Salminen et al. 2001; Salminen, 2003). In academic sector structured document management initiatives have included cases involving production of dissertations and journals (Müller, Klatt, Dobratz, and Bahnik, 2006; Sefton, 2007). Studies involving open documents have focused on the use of OpenDocument-compliant software and tools, and describing standardization process of open documents. Also reasons for the adoption of open document format in public sector have been studied (for example, Shah, Kesan, and Kennis, 2007). Karjalainen (2010) has made an innovation adoption study of the migration into the open-source OpenOffice.org software suite in the Finnish Ministry of Justice and its administrative sector. The default document format in the suite is ODF. Yet design and adoption of XML documents has remained as a scarcely studied area of XML research.

The goal of this study is to explore possible benefits of using custom XML schemas in office documents and factors influencing the design of such schemas. Because there are not many studies reported on this specific area of XML deployment, the objective of this study is to make discoveries through design ac-

tivities involving schema design. The focus of this thesis is defined by the following questions:

1. What kinds of objectives motivate organizations to deploy custom XML schemas for their office documents?
2. What kinds of guidelines schema design methods provide for designing custom XML schemas?
3. How can XML document management in an organization be analyzed and described?

This study aims to report relevant findings both for researchers and practioners in the field of XML document management. Findings are gathered by designing custom XML schemas for office documents and observing an XML document management case over the years after XML implementation.

The first research question is covered in the articles 2, 3 and 4. The second research question is covered in the articles 1 and 4, but also in the following chapter 3, which complements the article 1. Finally the third research question is discussed in detail in article 3 and in article 4. The empirical settings for the included articles contain participation in two real-life projects concerning schema design and XML implementation.

2.2 Research approach and research process

This study is qualitative in nature. The research is conducted by participating in two document management development projects. The two projects were carried out with action research approach.

2.2.1 Action research

Action research is about making discoveries through action taking (Baskerville, 2008). Action research produces knowledge to guide practice and it aims at an increased understanding of an immediate social situation. (Järvinen, 2007) According to Iivari (2007), action research is more focused on adopting technology than building it. Both MemoX project and RAKE project provided "lesson learned" types of findings from XML schema design efforts for office documents. Therefore, the dominant research approach in this thesis is action research.

Action research is seen as a method, which contributes to practical concerns of people in an organization. The approach emphasizes the utility aspect of the future system for people in case organization. (Järvinen, 2007) Like design science, also action research typically results in the building and evaluating of an artifact. (Järvinen, 2001) According to Baskerville (1999), the ideal domain of the action research method contains the following:

1. the researcher is actively involved, with expected benefit for both researcher and organization,

2. the knowledge obtained can be immediately applied, there is not the sense of the detached observer, but that of an active participant wishing to utilize any new knowledge, and
3. the research is typically a cyclical process.

The following five stages are often regarded essential in an action research cycle: 1) diagnosing stage, 2) action planning, 3) action taking, 4) evaluating, and 5) specifying learning. (Kock, McQueen & Scott, 1997)

The diagnosing stage includes the identification and definition of a general problem to be solved in the client organization. *Action planning* concerns the consideration of alternative courses of action to solve the problem identified. *The action taking* stage involves the selection and realization of one of the courses of action considered in the previous stage. *The evaluating* contains the study of the outcomes of the selected course of action. Finally, the *specifying learning stage* includes the study of the outcomes of the evaluating stage. (Kock, McQueen & Scott, 1997)

Participating in two cases was beneficial for both the researcher and the target organizations. The action research approach was appropriate, because there are not many real-life cases involving XML schema design for office documents reported in academic literature. Next the two cases are described as an action research processes. The whole research process is then summed up in Section 2.3.

2.2.2 Case 1: The MemoX system

MemoX is an XML-based document authoring system for the IT Faculty Council meeting agendas and minutes in the University of Jyväskylä. The Faculty Council meetings are held 11-14 times each year and typically an agenda contains 5-15 pages of text. Information for agenda items is received from various sources. For example, faculty departments, Administrative Office, HR Services, and Communication Services are relevant information sources for agendas. Information may be also received from job or student applicants, and some other external sources. The creation of meeting agendas and minutes is regulated by the rules concerning the decision-making process in the faculties and departments. In addition, the university has official guidelines how to carry out meetings related to administrative issues and what information agendas and meeting minutes should contain.

Before XML implementation agendas and meeting minutes were produced with MS Word and they were stored in the MS Windows file system in the network drive of the Faculty Office. Problems in document production, publishing, and delivery were the main motivations for starting the design of an XML-based solution for agendas and meeting minutes in 2004. A group of five majoring students in the IT faculty carried out the MemoX project where the author of this thesis was a member of the project group. Therefore, the case involved participatory observation. Data was also collected via end-user interviews and workshops.

The research was carried out in two action research cycles. The first cycle was carried out from September 2004 to March 2005. The second cycle was carried out between May 2005 and April 2013.

Cycle 1: Analysis and design 2004-2005

(1) Diagnosing. The IT Faculty Office personnel had problems in agenda publishing. Publishing agendas as paper copies was slow and web publishing of meeting minutes almost nonexistent. Despite of shared principles and recommendations for document styles, collaborative nature of the document authoring resulted in inconsistent layouts. Occurrence of identical content was typical in agendas and meeting minutes, and copying and pasting text portions from old documents into new ones was the method of document content reuse. This was a time-consuming and tedious task for the office personnel.

(2) Action planning. Based on literature on the subject area as well as experiences of previous document management projects, XML was seen as a promising technique to reuse document content more systematically as well as to support multi-channel publishing of agendas and meeting minutes. The task of designing a new XML-based document production for agendas was introduced to a student project group.

(3) Action taking. The RASKE methodology (Salminen, 2003) was adapted to carry out document analysis including the analysis of the agenda authoring process, the information sources connected to agendas, and the relevant roles related to the agenda production. Hierarchic content models formed the basis for XML schema design. MS InfoPath was chosen as the authoring tool of XML-based agendas. The design of custom XML schemas and InfoPath form design followed document analysis. These two tasks were carried out in parallel as schemas needed to meet the requirements of InfoPath's functionalities. Also preliminary XSL transformations for HTML output were created. The goal was to publish agendas with similar layout as used previously with MS Word documents.

(4) Evaluating. One of the most significant findings was that XML document production could be implemented with existing office applications without a need to invest to new and potentially expensive software or hardware. In 2005 this was a significant find, because XML format was not yet a native data format in common office applications. The timeframe of the project (six months) allowed the design of 16 preliminary XML schemas, InfoPath form templates, and XSL transformations for HTML outputs. (Jauhiainen, 2005) However, defining and testing final schemas required more work in future.

(5) Specifying learning. Before the MemoX project started, the RASKE methodology had been used in larger scale document management initiatives. Therefore, the document analysis phase had to be adapted to meet the needs of the smaller scale project. Document analysis provided means to gain understand-

ing of the document processing in the IT Faculty. However, defining schemas that were appropriate with MS InfoPath and testing the functionalities of the authoring tool by composing sample XML documents required more time. Nevertheless, MS InfoPath showed a great promise for XML-based office document production.

Cycle 2: Re-design, implementation and use 2005-2013

(1) Diagnosing. The work carried out in the first MemoX project remained incomplete and therefore there was a need to continue the work in order to construct an implementable XML-based agenda production in the IT Faculty Office. It was quickly realized that XML schema design in cycle one had been carried out insufficiently.

(2) Action planning. The earlier defined schemas for meeting agendas and minutes were re-examined. Similarities in schemas were identified, and places for schema component reuse discovered. This resulted in the conclusion that there were initially too many similar schemas defined for the similar items in agendas, and a plan to define one generic schema for similar InfoPath forms emerged. Furthermore, plans were made to finalize the preliminary techniques for document assembly as well as XSLT transformations for HTML and PDF outputs. These were seen essential before the system would be ready to be implemented in the Faculty Office.

(3) Action taking. A generic schema for agenda items was designed and tested in the InfoPath form. As a result, the number of required schemas decreased from 16 to four. This decision also decreased the number of the required InfoPath form templates and the XSLT files for HTML and PDF outputs. Because InfoPath did not provide WYSIWYG user interface, HTML and PDF outputs required careful design. The document layout had to remain similar with the MS Word document layout used before XML deployment. These requirements guided decisions on how elements were defined in the schemas.

The code for running document assembly was developed, and user guidelines were written and delivered for the end users in the IT Faculty Office. A training session for the office personnel was carried out after which the implementation of the new system was soon carried out in October 2005.

(4) Evaluating. It was observed that document layout had an effect on element definitions in schemas for agendas and meeting minutes. In addition, form-based user interface set its requirements on schema design and therefore schemas were further modified. In the end, the user interface was considered easy to use and the system overall beneficial. The MemoX system benefited the personnel of the IT Faculty Office by making the publishing of agendas and meeting minutes quicker and easier.

When identifying problems in the preliminary design of the solution that was carried out during cycle 1, it was concluded that document analysis should have revealed potential places of schema component reuse more efficiently in-

stead of focusing merely on the document structures. It was also concluded that the amount of schemas should be kept as small as possible in order to upkeep them efficiently as well as creating a maintainable system in general.

After the MemoX system had been used for three years, its potential benefits were evaluated in 2008 as new student project. The project analyzed agenda production in four faculties in the University of Jyväskylä. The XML-based agenda production was compared to more traditional word-processing document production. The ability of the MemoX system to produce both HTML and PDF outputs with consistent layouts was seen beneficial in environments where more than one person contributed in document production. Further evaluation of MemoX was carried out via interviews of the faculty's chief of administrative issues in 2011 and 2012. The questions aimed at discovering possible changes in the activities related to the document production, and the archival of the meeting minutes. Because of the MemoX system, the meeting agendas and minutes were now both delivered and archived digitally, in both HTML and PDF formats. Most faculty council members prefer the digital PDF version of meeting minutes to the paper version.

(5) Specifying learning. The end-user involvement in the projects was essential. Because the end-users were involved in the development process since the beginning, they had opportunities to influence on the user interface design.

From the schema design point of view, the MemoX development provided discoveries about relevant factors influencing design activities. Content reuse was seen as an essential objective. Therefore, content reuse issues should be considered during document structure modeling already. In principle, the XML document structure and external presentation can be managed as separate entities. However, it was observed during the case that layout design had a greater influence on schema design than expected. Also form-based user interface of the chosen authoring tool InfoPath set limitations on schema design decisions.

2.2.3 Case 2: RAKE project

The Finnish Centre for Pensions (FCP) is an organization that acts as a central body for private pension institutions in Finland. There are close to 400 employees at FCP and it functions under supervision of The Ministry of Social Affairs and Health. (FCP, 2013) FCP has been active in developing its document management practices (Jauhiainen and Honkaranta, 2007) and the RAKE project was one of the initiatives concerning document management development at FCP. Documents at FCP are handled as MS Word documents, but form-like documents are processed and stored in PDF format. A number of rules and regulations in Finland, EU and other countries affect on pension provisioning and therefore they also have an effect on document structures.

The RAKE project was a feasibility study on the administrational document types. The project was carried out during 2005 – 2006, but the development of document management has continued at the FCP after RAKE. The RAKE project group consisted experts from FCP and researchers from the University of Jyväskylä.

(1) Diagnosing. Management of document templates, document processing and content reuse at FCP was not effective. The main issue was the amount of document templates, which were created to support reuse of identical content in different document types. Furthermore, document templates were named unclearly and they were used from different locations (network drives, workstation hard drives), which resulted in multiple variations of same document templates and made the search of the appropriate template challenging, if not impossible. The main question in the RAKE project was to examine if the client organization could benefit from structured document management. The goal was to investigate how MS Word and MS InfoPath meet the needs of structured document authoring.

(2) Action planning. There were approximately a hundred form-like document types at FCP and their upkeep was kept separately from the document-like document types. First, candidate document types were chosen. Workshops for analyzing document types were planned. In addition, identifying the need for document content reuse and redesigning document types was carried out.

Four document types were selected, from which one was form-like. Two document types were chosen with the criteria of them having very static content, as their content and layout remained similar in all document instances. One document type contained reusable phrases, which were imported to a document from an external database, and for this reason it was chosen as a part of the analysis.

Previous document management cases were studied in order to review the central findings from them. It was decided to adapt the Maler and El Andaloussi method for document component identification before defining schemas.

(3) Action taking. The four document types were analyzed and their logical structures were modeled from the basis of existing document layout features. Instead of modeling document hierarchies in Elm diagrams, MindMap diagrams were used because the MindMap software was available and in use in the client organization. Also components and information sources connected to the documents, as well as the requirements for content reuse were analyzed.

XML schemas were designed for the selected four document types. XML schemas for the same document types were created for both MS Word 2007 documents and MS InfoPath 2007 forms and tested. Testing also required installing extension to the document management system that was used at FCP. Demonstration of XML document production for both form-based (MS InfoPath) and WYSIWYG (MS Word) user interfaces were prepared.

Schema design and schema testing was carried out in parallel, as composing a sample document against the designed schema with chosen document authoring tools fully reveals the places that may require re-defining. Because the project was a feasibility study, no real intervention was carried out in the target organization. As a result, findings from XML schema compatibility with office applications were reported and recommendations provided.

(4) Evaluating. It was concluded that the amount of document types at FCP could be decreased, if reusable document content was stored and managed as their own content units instead of managing multiple similar document templates containing identical content. Multichannel publishing is easier with XML, but XML deployment would require profound design before implementing solutions in case organization. It was also learned that retrieving data from external data sources, like databases, is possible with office software.

It became evident that schema design requires profound understanding of the document management environment including document content and operational requirements for documents. To carry out XML schema design project successfully, both XML experts and people with domain knowledge are required.

(5) Specifying learning. The RAKE project provided insights into the document types in real-life organization and the requirements for XML schemas for office documents. Following the document analysis steps of the Maler and El Andaloussi method was useful to identify document structures. Form-like document type required more detailed schema than document-like document types. Also user interface design influences how elements should be defined in custom schemas. What works in form-based documents may not be an optimal definition in document-centric document types and vice versa. Therefore, designing schema for form-based user interface differs from designing schema for WYSIWYG user interface, even for the same document type. Form-like office documents typically require more content-based schemas with detailed, document type specific element definitions whereas document-like documents may be defined with relatively generic schemas with structural element definitions such as elements for document heading, text paragraphs, and lists. These were findings that were not found in previous XML studies.

2.3 Research process

This study consists of examining existing literature on XML research, document management cases and schema design methods, as well as three empirical research cycles in two collaborative organizations. The research process is illustrated in Figure 4.

The ellipses in the figure present the main activities during the research process. Control flow arrows from up to down indicate the order of the activities. The arrows pointing right illustrate the outcomes of the activities as research articles and chapters of the thesis.

The research reported in this thesis was started with the MemoX development in September 2004. The development required two separate cycles. The work in the project was followed by a literature review.

The second case study involving XML schema design was carried out in 2005-2006. Combining the findings and empirical evidence from both cases led

to the article 2. The cases together with a more extensive literature review provided observations and findings for articles 3 and 4.

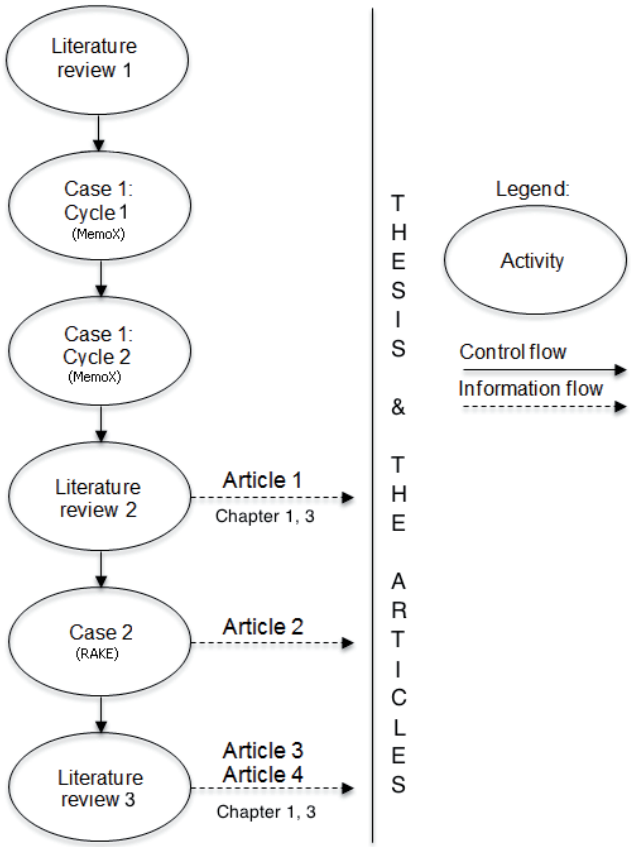

FIGURE 4 Research process of the thesis

Even though the MemoX project ended in 2005, its use and upkeep in the IT Faculty Office has been ongoing over the years. The possibility to observe system's use and evolution in a long-term scale has provided insights that were not initially expected when the study began in 2005. When MemoX project began, the standardization of ODF and OOXML was still unfinished. In addition, new versions of the office application were released during this study. The versioning had effects on the MemoX system. The longevity of the research has provided insights for the article 4 in particular. Therefore, the last activity phase in Figure 4 represents a period of time that is significantly longer than the earlier ones.

3 SCHEMA DESIGN METHODS

As a complimentary part of research question two this chapter introduces three methods containing guidelines for custom XML schema design. Schema design is seen as an activity taking place during document analysis, involving conceptual modeling of relevant document types, and resulting in physical XML schemas. The presented methods are the Maler and El Andaloussi method for DTD design (Maler and El Andaloussi, 1996), the Document Engineering Approach (Glushko and McGrath, 2005), and the Unified Content Strategy (Rockley et al., 2002). There are also other methods for document management development, such as the RASKE methodology (Salminen, Kauppinen, Lehtovaara 1997; Salminen, et al., 2000). The three methods chosen in this chapter were selected on the basis of difference in their primary application domains as well as the differences in modeling document structures.

3.1 The Maler and El Andaloussi method

The book "DTD design for SGML documents" of Maler and El Andaloussi (1996) has been considered as "the bible on mark-up language encoding rules" (Glushko and McGrath, 2005). The method presented in the book has even been considered as the best practice on the field of schema design (Thompson, 2000). The method contains five main phases from which first three are dedicated to document analysis and the latter two to design. The four phases are as follows:

- Document type needs analysis
- Document type modeling
- Mark-up design
- Validation and testing

Document type needs analysis focuses on the identification and classification of potential components (see chapter 1.3, p. 15) of document types. The aim is to find structural similarities in documents in order to find the potential components of each document type. Once components are identified, component clas-

sification may clarify possible misunderstandings schema designers may have about them. Sorting components into classes may also help to eliminate unnecessary components. The method guides schema designers construct component lists on which all identified potential components should be documented (Figure 12). It's also recommended that DTDs for similar document types should be examined after the component identification, if such documentation exists and is available. It is important to carry out this step after the components are identified, so that the earlier work won't bias the result of the current analysis work.

Figure 5 provides a simple example of a component list containing four components for a document type Memo. On a component list each component is given a definition and a short description. Potential components should also be organized in classes, which are based on schema designers' sense of the similarities between components.

Component name	Definition	Explanation and examples
Memo	Record of events from a meeting.	Memo contains 1-3 pages. Each page has a header and footer section.
Logo	Picture of the organization emblem.	Logo is presented on the first page of the memo, only. It is located on the top-left corner of the page.
Attendees	List of names and titles of people who attended to the meeting.	For example, "Director Jane Smith" and "Mr Paul O'Neill."
Item	An issue of discussion for the attendees to discuss and debate. Results in a decision.	Each memo contains at least three items, first two being "Call to order" and "Review of agenda".
Title	The descriptive heading of an item.	For example, "Proposition for the new employment strategy." Title is bolded.

FIGURE 5 Example of component list for memo document type

The phase of **document type modeling** focuses on the production of document hierarchy models, which visualize the characteristic "shape" of a document type. Document modeling in the Maler And El Andaloussi method is based on the Elm (enables lucid models) tree diagram notation. During document type modeling the relevant components are selected from the already composed component list. Figure 6 on the following page illustrates a document hierarchy model.

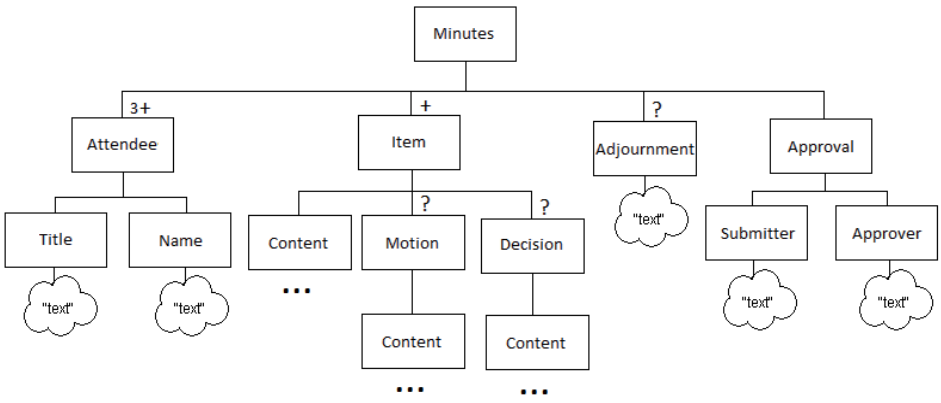

FIGURE 6 Document hierarchy model

In a document hierarchy model components are represented as rectangles and lines connecting them to each other represent the relationships between the components. Three dots underneath a component indicate that the structure of it is represented in another Elm tree diagram. The symbol "+" indicates that the component has to occur at least once, but it can also occur repeatedly, while "?" means that component is optional. The symbol "*" would represent an optional component, which occur several times. The sequence of components is written from left to right. Hierarchy ends where "text" visualized by cloud figures begin to appear. Unspecified text stands for the document content. Since each document instance is unique, the contents of these cloud figures vary. Hence the label "text".

Once document type modeling is carried out, there may still be components left on the list of potential components, which may be useful in organizing and composing the main content of documents. The remaining components may be information units, data-level components, or both.

Information units can to some degree "stand alone" in order to be understood by a reader. For example, a picture with caption may be identified as an information unit. Internal structures of information units should be modeled by using Elm tree diagrams. By contrast, data-level components are small information bits that need to be processed differently as the surrounding data. For example, a unit of measurement may be a data-level component. Data-level components should be mapped to elements and attributes in a schema.

If there are still components left outside of document hierarchy models, they may be potential link components. Some links, like references, may serve as instructions in document content assembly. There may also be links that connect two or more pieces of information (within same document type or not), like textual cross-references. **Producing the document analysis report** finalizes the document analysis phase of the method. In document analysis report all the documentation produced during the analysis from component lists to Elm tree models is assembled as one report.

After analysis phase schema development begins with **mark-up model design**. The number of required schemas should be decided. Mapping document hierarchy models into XML schemas is not, however, unambiguous task as some components in a document hierarchy model may end up as elements, others as attributes. There may even be a component, which is broken down in multiple elements and attributes. If there is a need to reuse document content, this should be taken into the consideration as well. Maler and El Andaloussi (1996) state that modularizing schemas into smaller schema components supports both document content reuse and customization of documents.

The phase of **validation and testing** involves reviewing the document analysis report. In addition, creating sample documents and validating them against defined schemas will test how well the markup design has been carried out.

3.2 Document Engineering Approach

Document Engineering Approach (Glushko and McGrath, 2005) is a method for analyzing, designing, and implementing documents in e-business software applications. One of the central goals of the approach is to support data and document exchange in business transactions on web. The main phases of Document Engineering Approach are the following eight:

- Analysis of the context of use
- Analysis of business processes
- Designing business processes with patterns
- Document analysis
- Document components analysis
- Assembling document components
- Assembling document models
- Implementing process and document models

Analysis of the context of use includes identifying relevant requirements for both documents and processes. Such requirements are typically rules for document exchange. There may be usage requirements as well as presentational requirements (i.e. how information should appear) attached to documents. **Analysis of business processes** involves examining business transactions between business partners. **Designing business processes with patterns** involves identifying and choosing appropriate patterns, which may set requirements for XML schema design. Patterns are general models of how processes are usually carried out. For example, if there is a pattern of how staff meeting memo is produced in an organization, generalizing pattern may lead to a common practice of how all memo types should be produced.

Document analysis covers the identification of relevant sources from which documents and their components may be found. This is typically an iterative task. Tasks carried out in a document life cycle should be identified in order to understand how documents are used. The goal of **document component**

analysis is to find the relevant components from the sources that were identified in the previous phase. Components may have both presentational and semantic structures, from which semantic structures are perceived more relevant in the context of e-business. However, separating semantic meaning from its presentation necessitates that also presentational components are recognized. Candidate components are listed in "harvest tables". Each identified component should be named meaningfully and semantic descriptions of them should be provided. This promotes common understanding about the components between schema designers and to encourage the use of reusable components.

The phase of **assembling document components** involves constructing document component models. *Document component models* are network models that provide a "domain view" of the relevant components and their associations (Figure 7). If components are more data-centric, classical data analysis approach including normalization is recommended when constructing the models. Normalization involves analyzing the associations between components and as a result, components may be generalized. For example, if there was address information included in two components, like Attendee and Approval, it could indicate a common pattern, which should be modeled as its own component in a document component model. In such case there're could be a component "Contact" present in the document component model.

Analysis may also include identification of primary key components which values are unique in every document instance. Also functional dependencies between components should be identified. For example, if a value of component "Price" changes because of a value of "Quantity" component has changed, there is a functional dependency between the two components.

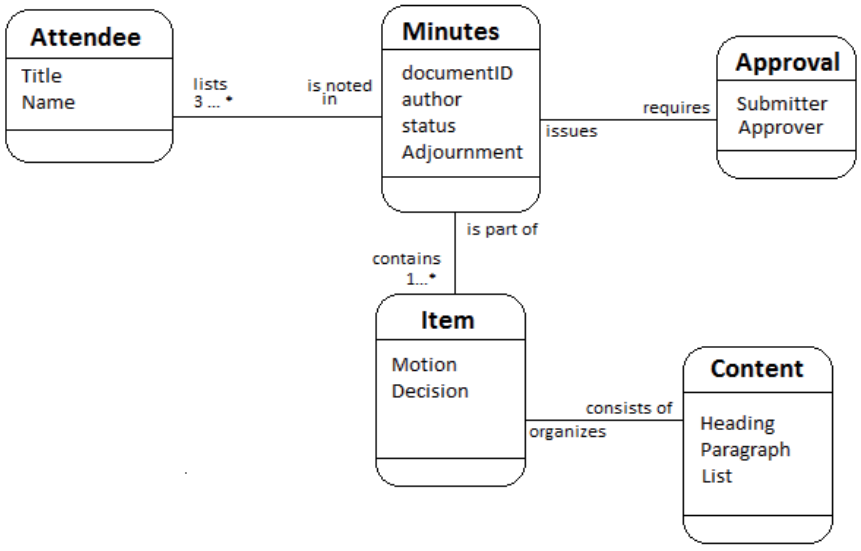

FIGURE 7 Document component model

According to Glushko and McGrath (2005), document component model is a generalized conceptual model defining all the necessary components to maximize reuse and minimize redundancy. Figure 7 illustrates a simplified version of a document component model, where components of a Minutes document type and their associations are presented. Components are modeled as rounded rectangles and one component in the model may represent entire document type or part of it. In Figure 7 there are five document components; Meeting Information, Memo, Approval, Attendee and Item. The model illustrates the cardinalities between components as well as the nature of the association between components. For example, Memo has to contain at least three items and each individual item is part of a Memo.

Assembling document models involves creating hierarchical document assembly models. This means that each component in document component model (Figure 7) is modeled in more detailed level. Recommended notations are Elm tree diagrams (Figure 6), UML class diagrams, and tables. If document assembly models share common structures, it may indicate places for document content reuse.

Implementing process and document models follows the design, and it covers encoding the created logical document models into physical schemas. The utilization of XML is recommended. According to Glushko and McGrath (2005), reusing schema components should be carried out whenever possible.

3.3 Unified Content Strategy

Unified Content Strategy (Rockley et al., 2002) is an approach for content management. Its goal is to create a strategy that enables an efficient content reuse. The method consists of five phases:

- Analyzing the content life cycle
- Performing a content audit
- Information modeling
- Designing metadata, dynamic content and workflow
- Implementing design

Analyzing content life cycles from creation to content delivery helps to identify the areas in current content management practices requiring improvement. In order to gain understanding of content itself, *a content audit* is performed. The purpose of content audit is to reveal the state of content use, and reveal requirements for more efficient content reuse. During top-level analysis content units, like documents, are examined to find common pieces of information, and the repeatable pieces of content are listed on a table. For example, if contact information occurs on multiple document types, it constitutes a repeatable piece of content. The findings of top-level analysis are compiled in a table. Figure 8 illustrates a simple table example that could be seen resulting from a top-level analysis. The table contains four document types: Agenda, Memo, Order and Invoice.

Content	Agenda	Memo	Order	Invoice
Logo	X	X	X	X
Contact information	X	X	X	X
Product description			X	X

FIGURE 8 Top-level analysis of four document types

In figure 8, both company logo and contact information appear in all document types on the table. Product description, however, occurs only in two document types, Order and Invoice.

In content audit's in-depth analysis repeatable pieces of content found on top-level analysis are further examined in order to find out, if they are reused identically, or with minor differences. For example, if organization's contact information occurs differently on memos and invoices, it should be determined, if there is a valid reason for the differences between them, or if such inconsistencies should be corrected. Finally a reuse map is created, which illustrates potential reuse of content components as well as the types of reuse, these being *identical* and *derivative*.

Figure 9 presents a reuse map containing four document types; Agenda, Memo, Order and Invoice. The letter I sands for *Identical reuse*, the letter D for *Derivative reuse*. For example, the letter I for Logo in all document types indicates that logo occurs identically in all document types. Also the contact information of the organization is a typical content component, which is being reused identically. However, in the case of Invoice, the contact information is reused derivatively. Derivative reuse means that the document author may edit the component's contents. Derivative reuse allows changes in component's contents, like ordering the content differently, inserting emphasis, etc.

Content	Agenda	Memo	Order	Invoice
Logo	I	I	I	I
Contact information	I	I	I	D
Product description			D	D

FIGURE 9 Simple example of reuse map for four document types

Design phase in the Unified Content Strategy includes **information modeling**. Figure 10 provides an example of information model for Minutes document type that corresponds to the earlier Minutes examples.

	A	B	C	D	E	F	G
1			Base element		Architectural		Production
2			XML	Text processor	Reuse	Guidelines	
3	Semantic structure	Element type					
4	Attendee	Container	Container	Heading 2	Systematic	Semantic	List of names retrieved
5	Title	Element	PCDATA/string	Normal	Systematic	Semantic	Retrieved automatically
6	Name	Element	PCDATA/string	Normal	Systematic	Semantic	Retrieved automatically
7	Item	Container	Container			Semantic	
8	Content	Container	Container			Semantic	Content component is reused
9	Heading	Container	Container	Heading 2		Semantic	
10	Paragraph	Element	PCDATA/string	Normal		Semantic	
11	List	Container	Container			Semantic	
12	Header	Element	PCDATA/string	Heading 3		Semantic	
13	List item	Element	PCDATA/string	List		Semantic	
14	style	Attribute	numbered|bulleted			Semantic	Style indicates bulleted or numbered list
15	Motion	Container	Container			Semantic	Motion should be bolded
16	Decision	Container	Container			Semantic	Decision should be bolded
17	Adjournment	Element	PCDATA/string	Normal		Semantic	Style: Italic
18	Approval	Container	Container	Heading 2		Semantic	Approval should be bolded
19	Submitter	Element	PCDATA/string	Normal		Semantic	
20	Approver	Element	PCDATA/string	Normal		Semantic	

FIGURE 10 Example of an information model

In Figure 10 the first column lists the components of a Minutes document type. The name of the document type is not visible in the model. The first column illustrates the semantic information that defines the structure of a document type. The second column, labeled as 'Element type', presents element types for components these being either "element" or "container" in XML terms. *Base information* guides schema designers to map logical structures with a chosen authoring tool. Base elements are listed for both XML documents and word-processing templates. For example, component `Attendee` may end up as container element in a XML schema, or as `Heading 2` style in a word processing software. *Architectural information* gives details about the type of reuse as well as guidelines how component should be defined in a schema. For example, all components are marked as "*Semantic*" indicate that semantic tags should be used in an XML schema. This means that an element should be given an understandable name. *Product information* is provided for the creation of style sheets, for instance. Product information may contain information about how certain component should be presented on web or on paper prints.

The goal of information modeling is to identify the structure of a document type, because it unifies the content, regardless of who is composing it. The level of detail these models have depends on the granularity. Tables or worksheets are utilized during the modeling.

In addition to information modeling, the design phase of the method may involve designing appropriate metadata, dynamic content, and workflows. **Implementing the design** follows the design activities. If XML is used, custom XML schemas for document types are required. The information models produced during design are "XML-ready", so therefore binding information models from logical level of abstraction into physical models (i.e. schemas) is supported. The selection of schema language, however, depends on the need of data typing as well as the tool used in content authoring.

3.4 Comparison of the methods

The Maler and El Andaloussi method is targeted for publishing industry whereas the Document Engineering Approach is targeted for developing automated e-business processes. The Unified Content Strategy is directed to the organizations content management practices. Despite the different application domains, the methods share a common goal: creating XML-based solutions with custom XML schemas. In this chapter the three methods are compared against the following three features of office documents:

- Office document is authored by a person for human consumption.
- Office document functions as a record of an administrational process, or business transaction.
- An office document may receive part of its content from an external content source, such as database.

Table 4 summarizes the characteristics of the three methods via three identified features for office documents. The features are presented in the three rows. M&A stands for the Maler and El Andaloussi method, DEA for the Document Engineering Approach, and UCS for the Unified Content Strategy.

Office documents have document-centric properties as they are mainly targeted for human consumption. Therefore, there is a layout attached to them. Examining document layout is the key task in component identification in all three methods. Process analysis is covered only in the Document Engineering Approach. This is a clear distinction between the Document Engineering Approach and the two other methods. The Unified Content Strategy emphasizes identification and utilization of reusable content more than the Maler and El Andaloussi method and the Document Engineering Approach. Both of these methods have adapted principles from the Maler and El Andaloussi method to identify and model components that constitute a document-centric document type.

More research and especially findings from case studies involving XML-based production of office documents is needed to reveal what needs to be taken into consideration when designing schemas for office documents. In addition, there may also be unique characteristics and/or requirements to be found from the context of office documents, which are not included into the existing methods and approaches containing guidelines for XML schema design.

TABLE 4 Comparison of the methods

	Feature 1. Office documents are authored by people for human consumption.	**Feature 2.** Office documents function as a record in an administrational process, or business transaction.	**Feature 3.** Office documents may receive some part of their content form external data source, like database.
M&A	The basis of the analysis is on how people perceive document content. Components reflect information about document content's meaning and about its formatting, or both. Therefore, analysis is based on how people understand document content.	The method focuses on how to carry out schema design project. As a result of document analysis, documents' hierarchical structures are identified. Processes are not seen to influence on document type modeling. Hence, there are no guidelines to examine processes.	If such content exists, it is imported to documents by using entity declarations in DTDs. The possibility of such content is not, however, explicitly expressed in Elm tree diagrams.
DEA	Some documents in e-business context may be targeted for people instead of systems. Therefore, the method includes component identification and modeling from document-centric document types, too.	Business processes are analyzed and components may be identified during this phase. Process analysis may also reveal documents that are relevant in the context. The models presenting document structures are constructed after processes are analyzed.	Some components may not come in traditional document form, but from an external data source. Data-centric components may come in a form of labeled data entry fields, or as source code of an application processing documents. Document component models do not exclude the possibility of this feature, yet they don't explicitly illustrate the feature either.
UCS	Documents are meant for human consumption by default. The goal is to make content as reusable as possible and content life cycles as effective as possible. Therefore, components that are identified from document-like information sets are in the scope of analysis and design.	Schema designers should analyze existing content life cycles in the target organization and identify possible issues in them. Document life cycles reveal the lifespans of documents, yet this has only little impact on the analysis and design goals of the method; effective content reuse. Therefore, processes do not influence on document structure modeling.	The method focuses on components that are identified from document-like information sets. The possibility of documents containing data outside of traditional document set is not, however, necessarily excluded.

4 SUMMARY OF THE INCLUDED ARTICLES

This chapter describes the research objectives, methods, and results of the four included articles. Three of the articles are published in conference proceedings and the fourth article is submitted to a journal. Three of the four articles are based on case studies on schema design and implementation of XML document production. Each of the articles published in conference proceedings reflect the theme of each conference, yet their connection to XML schema design and the research of this thesis is indisputable.

4.1 Article 1: "Two Methods for Schema Design for Intelligent XML Documents in Organizations"

Honkaranta, A. & Jauhiainen, E. (2007). Two Methods for Schema Design for Intelligent XML Documents in Organizations. In Technologies for Business Information Systems. Dordrecht, Netherlands: Springer.

4.1.1 Research objectives and methods

The aim of this article was to review and compare schema languages for XML documents and compare methods on how to design custom XML schemas for intelligent documents in organizations. By intelligent documents we refer to textual office documents, which utilize domain-oriented XML schemas with meaningful element names.

The three schema languages chosen for the comparison were DTD, RELAX NG and XSD, because they are the most widely used on different application domains. Therefore, they were seen potential languages for office document design as well. From the guidelines for designing document-oriented XML documents the guidelines of Maler and El Andaloussi (1996) and Kennedy (2003) were selected for comparison. Maler and El Andaloussi (1996) was selected because the method they have developed is considered as the best practice on the field of document analysis and DTD design. Kennedy (2003) was

chosen because it provided predefined steps for designing document-centric RELAX NG schemas. The goal was to compare guidelines in order to find similarities in them, even though they were targeted for different schema languages. The article addresses the research question 2

4.1.2 Content and results

A comparison of schema languages was provided with an example XML markup that was validated against corresponding schemas in DTD, XSD and RELAX NG. DTD and RELAX NG are primarily meant for document-centric XML use whereas XSD supports both document- and data-centric use.

The guidelines for schema design carrying out document analysis and document type modeling in Maler and El Andaloussi (1996) are originally developed for DTD design. Document components are identified and classified before modeling them in Elm tree diagrams. The diagrams show how element content should be organized in XML documents.

Kennedy's guidelines for schema design can be seen as a revisited version of the Maler and El Andaloussi guidelines, but Kennedy targets his guidelines for RELAX NG schemas. Like Maler and El Andaloussi, Kennedy recommends carrying out data analysis first, which corresponds to component identification and classification tasks by Maler and El Andaloussi. Kennedy recommends leaving any presentational elements to style sheets and his recommendation to use collection elements is also based on style sheet design. He also encourages avoiding mixed element content, as such definitions would make schemas too general, and avoiding deep element hierarchies. Maler and El Andaloussi do not make similar suggestions. They simply introduce a set of different modeling considerations on how to convert Elm diagrams into DTDs without recommending any type of element declaration over another.

The comparison of the methods revealed many similarities. Both acknowledge the importance of document components and modeling them in Elm tree diagrams. Elm diagrams provide a visual representation of the document structures, which are seen easy to convert into corresponding schemas. Both acknowledge defining schemas as an iterative task because schemas can truly be tested only after content has been authored with a defined schema. Also, defining meaningful element names is recommended as they make schemas more human-readable. The similarities in the two introduced guidelines suggest that both of them are adoptable for office documents. In addition, the DTD syntax makes the language easy to learn and adapt. As the modern XML editors provide a quick and easy conversion from one schema language to another, DTD is a relevant schema language for office documents, even though authoring software would support XML-encoded schema languages.

4.2 Article 2: "Aspects on XML Document Content Reuse in Organizations"

Jauhiainen, E., Honkaranta, A. (2007). Aspects on XML Document Content Reuse in Organizations. In Weide Chang, James B.D. Joshi (Eds.). Proceedings of the 2007 IEEE International Conference on Information Reuse and Integration (IEEE IRI-07), Las Vegas, 588-593.

4.2.1 Research objectives and methods

This article aimed to discover ways document content is reused in office documents. The article is based on an empirical data derived from two case organizations described earlier in chapter 2.2. The two cases involved designing custom XML schemas for office documents. Both authors of the article were involved in the design in both cases. Participatory observations were made from design activities and schemas resulting from them. Overall, six document types were involved in schema design; two document types in the MemoX case (called case of FIT in the article) and four document types in the RAKE case (called FCP in the article). This article addresses the research questions 1 and 2.

4.2.2 Content and results

Designing schemas for the document types in the two projects provided the basis for the findings presented in the article. The findings from the two cases strongly suggested that document content reuse is an essential goal in XML deployment in office documents. It was observed that there were different ways to reuse document content, which also guided how schemas were designed.

Four types of document content reuse types were identified from the two cases: 1) A document may be reused as a part of another document. 2) Some reusable components are utilized to modify content for a certain medium. 3) In organizational domain there might be repeatable phrases and/or text portions, which may be reused across multiple documents. 4) There may be both internal and external databases or services that provide content to documents. As the reuse types were identified and explained, the effect they had on designing custom XML schemas for office documents were also described. If custom XML schemas for office documents are designed for effective document content reuse, the schema modularization has to be carefully considered.

Properties of software used in XML document authoring may set requirements and limitation for document content reuse. Document layout may indicate reusable components as the same document instance may have more than one layout depending on delivery medium. In order to gain understanding of factors influencing schema design for office documents, and their dependencies, a model was constructed. The four factors of schema design were identified as 1) schema, 2) component, 3) layout, and 4) the software used in XML document production. For example, the granularity of a component and document layout has an influence on a custom schema. The software used in

document production sets its own requirements on how content reuse can be realized, and user interface dictates how much emphasis should be placed on document layout design. The findings suggest that each of these four aspects may guide schema design and therefore they should be included in analysis work.

4.3 Article 3: "XML Document Implementation: Experiences from Three Cases"

Nurmeksela, R., Jauhiainen, E., Salminen, A. & Honkaranta, A. (2007). In Youakim Badr, Richard Chbeir, Pit Pichappan (Eds.). Proceedings of the Second International Conference on Digital Information Management. Los Alamitos, CA: IEEE, 224-229.

4.3.1 Research objectives and methods

This article compared the findings from three cases involving re-engineering of document management by implementing SGML/XML documents production. The goal of the comparison was to discover motivators behind SGML/XML adoption. Furthermore, possible changes in document production as well as how XML adoption process was realized were on the scope of comparison.

Case 1 concerned legislative documents within and between organizations. The document types of case 2 were meeting agendas and memos. In both of these cases the document types were considered as document-centric documents. In case 3 the document type involved was an invoice, which had both data- and document-centric properties. Nevertheless, invoice can be seen as an office document as well.

Data was collected via participatory observations and project work via which real-life experiences of schema design and XML document authoring were obtained. Nurmeksela and Salminen were involved in case 1. Honkaranta and Jauhiainen were involved with case 2. Nurmeksela was involved with case 3. Comparing cases was carried out in co-operation with all the contributors of the article. This article addresses the research questions 1, 2 and 3.

4.3.2 Content and results

The paper first described the characteristics and types of document production. A model of XML standardization process was then described and used as a framework for the comparison of the cases. Also information about the content types of the cases was given.

Our findings suggest that implementing production of XML documents is a domain-specific task that involves co-operation between schema designers and organizations involved. The motivation to implement included inconsistencies in content management, content reuse issues and automating processes related to document use. If XML document production can be embedded in ex-

isting processes or systems within organizations, the need to change work practices related to document production decreases.

Document layout formed a significant aspect of the schema design in all three cases. In Cases 1 and 2 documents were primarily targeted for human consumption and therefore document layout was an important part of the standardization process. In Case 3, the invoice as the document type had both document- and data-centric characteristics. Nevertheless, layout was important in this case as well, because a unified layout was required for all parties involved.

The analysis of cases revealed that schema design was an iterative process in all the cases. In Case 1, document schemas were designed incrementally within two years. In Cases 2 and 3, document schemas were designed within months. In Cases 1 and 3 usability requirements for the authoring tool impacted on schema design. In both cases layout requirements were strict and therefore document's presentational properties had to be kept in mind during schema design. In both cases XML markup was hidden from the document authors, which made adapting new type of document production easier for the end-users and decreased user resistance. In fact, if there is many document authors involved, a major emphasis has to be given to the usability of authoring tools.

4.4 Article 4: "A Life Cycle Model of XML Documents"

Salminen, A., Nurmeksela, R. and Jauhiainen, E. (2013). A Life Cycle Model of XML Documents. The Journal of the American Society for Information Science and Technology (JASIST). Accepted October 2, 2013.

4.4.1 Research objectives and methods

The main objective for the article was to provide a framework for analyzing and describing XML document management in organizations throughout their life cycles. The design science method was followed in the creation of this article. Data was collected in a number of different ways. An extensive literature review was carried out covering document management, records management, and content management. Empirical data was collected by means of observation, expert interviews and literal sources from two cases. One of them concerned the State Budget Proposal of the Finnish Government, the other the Faculty Council Meeting Agenda at the University of Jyväskylä. The cases were also used to demonstrate how the framework introduced in the paper could be used to describe XML document management in an organization. The author of this thesis was responsible for the analysis and description of the Faculty Meeting Agenda case. The case in question has also been utilized in other joined articles of this thesis, but in this article the viewpoint was shifted from XML schema design to XML document management. This was because of the timeframe; analyzing the case after years of implementing XML in the case environment enabled making observations from XML document management perspective

instead of evaluating the design of XML deployment only. Interviewing the end-users in the case organization provided updated data and insights from the case. This article addresses the research questions 2 and 3.

4.4.2 Content and results

First, the concepts related to XML document management were introduced. Four methods available for analyzing and describing structured document management were briefly described and compared. The comparison showed differences in the application domains and modeling methods of the methodologies. All of the methodologies were seen to lack support for the analysis of XML document management throughout document life. The main contribution of the paper was provided as a new life cycle model of XML documents. The model consisted of five phases: design, content production, capture and dissemination, use, and finally retention. For each of the phases the typical activities, actors, systems, and types of content items related to the management of XML documents were described. The model utilized the concepts of the RASKE methodology and can therefore be seen as an extension of the methodology. The life cycle model together with the earlier introduced RASKE components provides a framework to analyze and describe XML document management from different perspectives.

The applicability of the framework was demonstrated by two case descriptions. The benefits of the framework were evaluated by informal argumentation as well as by utilizing the feedback from the developers and decision makers involved in the cases.

4.5 About the Joint Articles

The four articles included in the thesis are a result of co-operation of digital media study line staff at the University of Jyväskylä. The articles 1 and 2 were co-authored with Anne Honkaranta, a supervisor of this thesis. Article 1 is a book chapter based on a conference paper. Article 2 reports research findings from two projects – MemoX and RAKE projects. Article 3 combines research findings from three case environments: the Finnish Parliament, the Faculty of Information Technology at the University of Jyväskylä, and an international ICT service provider, which was the employer of one of the contributed researchers of the article. Article 4 combines results from long-term RASKE development efforts and experiences from two cases: the State Budget Proposal of the Finnish Government and the Faculty Council Meeting Agenda at the University of Jyväskylä.

In article 1 both Eliisa Jauhiainen and Anne Honkaranta conducted the literature review of existing schema languages and methods. Anne Honkaranta contributed in organizing the paper as well as providing significant insights in the introduction and discussion parts of the article. In article 2 Eliisa Jauhiainen was

the responsible author while the iterative writing process as well as profound discussions on the topic took place with Anne Honkaranta.

Article 3 was a result of co-operation between four digital media study line researchers – Reija Nurmeksela, Eliisa Jauhiainen, Airi Salminen, and Anne Honkaranta. The responsible author was Reija Nurmeksela who came up with the idea of the paper. Eliisa Jauhiainen analyzed and described the second case of the article. The evaluation of the cases and conclusions were written co-operatively with all authors. Article 4 was co-authored with Airi Salminen and Reija Nurmeksela. Airi Salminen was the main author of the paper. Eliisa Jauhiainen contributed on methodology comparison and in particularly in the analysis and description of case 2 presented in the article.

5 RESULTS AND IMPLICATIONS

This chapter summarizes the contributions of this study. The researcher took part in two research studies from which data was collected. In addition, three methods for XML schema design were reviewed and compared. The focus of the study was characterized by three research questions, which are revisited in this chapter of the thesis. The discussion of the implications and shortcomings of the study finalizes the chapter.

5.1 Contributions

The objectives of this study were to explore factors influencing the design of custom XML schemas for office documents, and to identify possible benefits of using such schemas.

What kinds of objectives motivate organizations to deploy custom XML schemas for their office documents?

In the case organization deploying custom XML schemas for their office documents the main motivators were improvements related to document publishing and content reuse. In the MemoX case collaborative document authoring lead to incoherent document layout. Each document author had used, at least occasionally, her freedom to style her documents in her own way. In addition, publishing documents on intranet in HTML format was a challenge for document authors. These were issues that were seen resolvable with the adoption of XML. For example, XML-based document authoring enabled publishing document content in both HTML and PDF formats, and reusing document content more efficiently with XML was achieved. After implementation these were regarded as the main benefits achieved.

Reusing content more efficiently was seen as an important goal in both MemoX and RAKE project. Content in office documents can be reused in different ways. For example, a document author may retrieve manually the piece of

content and add it to the document. Content can also be reused automatically by a system. (Jauhiainen and Honkaranta, 2007)

Content reuse may also involve reusing schema components. Maler and El Andaloussi (1996) call this approach as *schema modularization*, which means that element and attribute declarations in a single schema component are applicable for more than one document type. Schema modularization has its benefits. For example, updating definitions in reusable schema components may be more efficient than making modifications to multiple schemas for different document types containing similar schema definitions.

What kinds of guidelines schema design methods provide for designing custom XML schemas?

This thesis reviewed three methods for schema design:

- the Maler and El Andaloussi method (Maler & El Andaloussi, 1996),
- the Document Engineering Approach (Glushko & McGrath, 2005) and
- the Unified Content Strategy (Rockley et al., 2002).

The Maler & El Andaloussi method (1996) provides a solid approach to identify and classify components from document types. Documents' presentational features are seen as the starting point of document analysis in each three methods. Document analysis involves identification of document components, which are modeled in document structure models.

Document content reuse issues are addressed in all methods. The Unified Content Strategy emphasizes the importance of reuse the most. There may be content in office documents that is imported from external systems. The Document Engineering Approach is the only method from the three reviewed that includes the possibility of such content explicitly.

To summarize, the methods reviewed in this study provided following guidelines for designing custom XML schemas:

- Schema design involves identification of components that build a document type.
- Components may be derived from the external presentation of existing documentation, but also from sources outside of a document set, if document has data-centric properties.
- Gaining understanding of processes related to document use may reveal essential characteristics of document content and document use.
- Document type modeling provides a good basis for designing custom XML schemas.
- Document content reuse is a central goal in schema design. Therefore, reusable content should be identified during document analysis and modeling.
- Schema design is an iterative task and schemas should be tested by end-users before deploying them.

– People creating and using documents have know-how on both domain and document use and therefore they should be included in the process of schema design.

How can XML document management in an organization be analyzed and described?

In article 4 we developed a framework to analyze and describe the management of XML documents throughout their life, from design to retention. The framework provided models to depict the core components of the document management environment and the life cycle of documents. In the article the management of the Faculty Council Meeting Agenda of the MemoX case of the thesis is used.

From the activities of an XML document management environment, schema design is in the focus of this thesis. XML schema design can be perceived as an activity that takes place in an environment including three kinds of information sources; content items, actors, and systems. Therefore, XML schema design is an activity that

- involves analysis of existing documents and other information sources related to them,
- should acknowledge the people who use documents as sources of information as well as end-users to whom schemas are designed, and
- identifies existing systems and possible new systems in XML deployment.

The life cycle model of article 4 provides means to consider the whole life cycle of XML documents already at the schema design phase. Schema design initiatives may address the document life cycle phases only until publishing, even though upkeep and archival issues might also need consideration. In the reviewed methods of this study the archival and retention of documents is left outside their scope.

5.2 Implications

The two reported cases suggest that document layout has a significant role in designing custom XML schemas. Layout requirements for office documents might be very strict. Nurmeksela et al. (2007) have stated that in situations where traditional, binary file documents are transformed into XML documents, a typical requirement is to keep document layout similar to that before XML deployment. Both MemoX and RAKE projects proved that document layout is important for end-users and layout design has an effect on schemas. Roisin and Vatton (1993) stated already 20 years ago that style properties of documents are related to the logical structure of documents. Also Walsh (2002) has stated that a perfect separation of content and presentation is not always possible in all cases, even though it is often possible to come very close. Therefore, schema

design and layout design are both essential parts of XML schema design for office documents.

None of the compared methods in this thesis discusses the role of authoring software. The two cases revealed, however, that software used in document authoring may set restrictions on how schemas are defined. For example, if the chosen authoring tool provides form-based user interface, recursive element declarations may be difficult to deploy. On the other hand, deep and detailed document hierarchies may not be ideal for WYSIWYG interfaces, which typically allow document authors type document content relatively freely.

The two cases and the three compared methods imply that the following phases are necessary in schema design efforts for office documents.

1. Begin with document analysis during which relevant document components are identified. Principles of the Maler and El Andaloussi method are applicable.
2. Model document structures; hierarchical structure of a document type provides a preliminary idea of the schema that is required for a document type.
3. Remember content reuse. Study your document structure models and look for similarities. They may suggest places of reuse. Reusable content may also be identified when examining existing documents; the principles of the Unified Content Strategy are applicable.
4. If a part of document content is retrieved from external systems, identify them. In what form is data imported to documents? Is XML used? Can XML be adopted? The principles of the Document Engineering Approach may be applicable to analyze such content.
5. Define your schema. The authoring tool used for XML document production may guide the choice of schema language as well as how elements should be defined. Keeping the number of schemas as low as possible may be beneficial from the document management viewpoint.
6. Test your preliminary schemas with the selected authoring tool, preferably with the end users. This may reveal places in schemas that require modification. This phase may require multiple iterations.
7. Do not ignore the layout requirements. Does the chosen software used provide the layout directly (WYSIWYG user interface), or is there a need to provide a print view for document authors? Is there a need for multiple views? Are the documents going to be published on more than one media? Be ready to re-design schemas to meet the requirements of layout(s).

6 CONCLUSION

This study focused on designing custom XML schemas for office documents. XML and its related standards and tools support data interoperability, content manipulation, content sharing and reuse, document assembly, document security, document filtering, and document formatting for all types of devices and applications (Adler, Cochrane, Morar, and Spector, 2006). Office documents are one of the latest big application areas for XML use as the open standards ODF and OOXML have brought XML to office environments by replacing the earlier binary file formats.

Office documents are typically composed with word-processing software and they are relatively short in length for human consumption. From the viewpoint of XML use and XML document management, office documents may have both technical requirements familiar from data-centric XML use, yet they also contain requirements for human consumption that are typical in document-centric XML use. Despite of XML being the default format in the latest office applications, document management issues are not resolved by adoption of the default XML. In fact, document standardization may be needed in order to get the most out of the possibilities that XML has to offer. Document standardization may require designing and deploying custom XML schemas for office documents. The main motivators to utilize custom XML schemas have a lot to do with making activities related to document creation and handling quicker and easier. The main motivators are:

- content reuse, and
- document publishing.

The research area itself underwent significant changes since this study was started. In 2005, when the research initially began, the open XML standards for office documents were still under development. The knowledge of such standards becoming part of office suites was already acknowledged, yet there was no means to predict what kind of impact these standards would have on the research topic of this study. However, in 2005 there were already office suites available, which supported the use of XML. For example, using custom XML schemas in MS Word and MS InfoPath was possible.

The two cases of this study – MemoX and RAKE - involved analyzing office documents and designing custom XML schemas for them. Both cases were carried out as an action research study. Action research is an ideal research method to gain more accurate understanding of the XML deployment in office environments. By studying XML schema design via action research method, understanding of what is really taking place in real organization was gained.

The initial scope of this research was to focus on schema design for office documents specifically and research papers based on these projects reflect the scope, too. However, as the time went by the focus of the research shifted. Instead of focusing merely on the design phase of office document standardization involving schema design, the scope expanded to the use and management of such documents, too. Especially the MemoX project provided an opportunity to make such observations. After its implementation in 2005 it has been continuously used in the IT Faculty at the University of Jyväskylä and a few updates have been done. The work has also continued at the FCP after the RAKE project. MemoX observations were reported in article 4.

The first research question of this study considered about possible motives organizations may have to deploy custom XML schemas for their office documents. The articles 2, 3 and 4 provided answers to this question. The second research question focused on schema design methods and how they guide designing custom XML schemas. Answers to this question were provided in the introductory part of this thesis and article 1.

The third research question focused on the description of XML document management in an organization. This question was answered in articles 3 and 4. It was concluded that document management should be considered beyond the implementation of XML-based solution. How XML documents are maintained and how they will be archived, for instance, may have an effect on schemas on the element definition level. The introduced methods in this study do not take this viewpoint into consideration.

The contributions of this study are the insights and identified characteristics of custom XML schema design for office documents. More empirical evidence of the real-life XML schema design cases is needed to reveal more specific details about the subject.

REFERENCES

Adler, S., Cochrane, R., Morar, J. F., Spector, A. (2006). Technical context and cultural consequences of XML. IBM Systems Journal, 45(2), 207-223.

Baskerville, R. L. (1999). Investigating information systems with action research. Communications of the AIS, 2, 1-32.

Baskerville, R. (2008). What Design Science Is Not. In the European Journal of Information Systems, 17, 441-443.

Bertino E., Catania B. (2001). Integrating XML and Databases. Internet Computing, IEEE, 5(4), 84-88.

Blumberg, R., Atre, S. (2003). The Problem with Unstructured Data. DM Review. Accessed 14.11.2012. Available at: http://soquelgroup.com /Articles /dmreview_ 0203_problem.pdf.

Boiko, B. (2002). Content Management Bible. New York: Hungry Minds, Inc.

Booch, G., Rumbaugh, J., Jacobson, I. (1997). The Unified Modeling Language User Guide. (1st Edition). Massachusetts: Addison-Wesley.

Boyer, J.M. (2008). Interactive Office Documents: A New Face for Web 2.0 Applications. In Bulterman & L.F. Gomes Soares (Eds.). Proceeding of the Eighth ACM Symposium on Document Engineering, DocEng '08, 8-17. New York, ACM.

Bray T., Paoli J., Sperberg-McQueen C.M, Maler E., Yergeau, F. (Eds.). (2008). Extensible Mark-up Language (XML) 1.0 (Fifth Edition), W3C Recommendation. Available at: http://www.w3.org/TR/REC-xml/

Broberg, M. (2004). A Succesful Documentation Management Systems Using XML. The Journal of the Socitey for Technical Communication. 51(4), 537-546.

Bubenik, W., Hanke, I., Juhnke, N. (2005). XML-Based Authoring: From Concepts, via Compromises to Applications. ELPUB2005 Conference on Electronic Publishing, 51 - 56. Accessed 14.11.2012. Available at: http://elpub.scix.net/data/works/att/210elpub2005.content.pdf

Clark J. (1999). XSL Transformations (XSLT) Version 1.0. W3C Recommendation 16 Nov. 1999. W3C Consortium. Available at: http://www.w3.org/ TR/xslt

Clark J. (2001). The design of RELAX NG. Accessed 10.5.2010. Available at: http://www.thai open source.com/relaxng/ design.html

Clark J., Murata M. (2001). RELAX NG Specification, OASIS Specification Available at: http://www.oasis-open.org/committees/relax-ng/spec-20011203.html

DuCharme B. (2004). Documents vs. data, schemas vs. schemas. XML 2004, 1554–4648.

Elmasri R., Wu Y-C., Hojabri B., Li C., Fu J. (2002). Conceptual modelling for customized XML schemas. In 21st International Conference on Conceptual Modelling (ER 2002), 429–443. Springer Berlin Heidelberg.

Fallside D.C., Walmsley P. (2004). XML Schema Part 0: Primer Second Edition. W3C Recommendation 28 October 2004. W3C Consortium. Accessed 12.4.2006. http://www.w3.org/TR/xmlschema-0/

Fiala Z., Hinz M., Wehner F. (2003). A Component-based Approach for Adaptive, Dynamic Web Documents. Journal of Web Engineering, Rinton Press, 2(1&2), 58–73.

Fierz, W., Grütter, R. (2000). The SGML Standardization Framework and the Introduction of XML. In Journal of Medical Internet Research, 2(2).

FCP. (2013) The Finnish Centre for Pensions. Accessed: 7.3.2013. Available at: http://www.etk.fi/en/ service/about_us/1222/about_us

Garfinkel, S.L., Migletz J.J. (2009). New XML-based Files; Implications for Forensics. In IEEE Security & Privacy, 7(2), 38–44.

Glushko R.J., McGrath T. (2005). Document Engineering; Analyzing and designing documents for business informatics & Web services. Massachusetts: The MIT Press.

Goldfarb C.F. (1990). The SGML handbook. Oxford: Oxford University Press.

Hackos, J. (2002). Content Management For Dynamic Web Deliver. Indianapolis: Wiley Publishers.

Harold, E.R., Means, W.S. (2004). XML In A Nutshell (Third Edition). Sebastopol, CA: O'Reilly Media.

Honkaranta, A., Jauhiainen, E. (2007). Two Methods for Schema Design for Intelligent XML Documents in Organization. In Technologies for Business Information Systems, 173-182. Dordrecht, Netherlands: Springer.

ISO (1986). 8879:1986 Information processing -- Text and office systems -- Standard Generalized Markup Language (SGML). International Organization for Standardization, Geneva, 1986.

ISO (2006). ISO/IEC 26300:2006 Information Technology – Open Document Format for Office Applications (OpenDocument) v.1.0, ISO/IEC International Standard.

ISO (2008). ISO/IEC 29500-1 Information Technology – Document Description and Processing Languages – Office Open XML File Formats – Part 1: Fundamentals and Markup Langauage Reference. ISO/IEC International Standard Under Construction.

Iivari, J. (2007). A Paradigmatic Analysis of Information Systems as a Design Science. Scandinavian Journal of Information Systems, 19(2), 39-64.

Jauhiainen, E. (2005). RASKE-menetelmän soveltaminen: Havaintoja kahdesta Jyväskylän yliopiston opiskelijaprojektista. Master's Thesis. University of Jyväskylä.

Jauhiainen, E., Honkaranta, A. (2006). A Review on XML Document Schemas and Methods for Schema Design. In Abramowicz, W., Mayr, H.C. (Eds.). 9th International Conference on Business Information Systems (BIS 2006), 201-214. Bonn, Germany: Köllen Druck+Verlag GmbH.

Jauhiainen, E., Honkaranta, A. (2007). Aspects on XML Document Content Reuse in Organizations. In Weide Chang, James B.D. Joshi (Eds.). Proceedings of the 2007 IEEE International Conference on Information Reuse and Integration (IEEE IRI-07), 588-593. Las Vegas, CA: IEE Computer Society.

Järvinen, P. (2001). On Research Methods. Tampere: Opinpajan kirja.

Järvinen, P. (2007). Action Research Is Similar With Design Science. Quality and Quantity 41(1), 37–54.

Karjalainen, M. (2010). Large-scale migration to an open source office suite: An innovation adoption study in Finland. Dissertation. University of Tampere.

Kennedy D. (2003). Relax NG with XML data structures. In Mann, S., Williamson, A (Eds.). Proceedings of the 16th Annual NACCQ, 91-102. New Zealand: Palmerson North.

Klischewski, R. (2006). Ontologies for e-document management in public administration. Business Process Management Journal, 12(1), 34-47.

Kock N.F., McQueen R.J., Scott J.L (1997). Can action research be made more rigorous in a positivist sense? The contribution of an iterative approach. Journal of Systems & Information Techonology, 1(1), 1-24.

Lappin L. (2010). What Will Be The Next Records Management Orthodoxy? Records Management Journal, 20(3), 252-264.

Levien, Roger E. (1991). The civilizing currency: documents and their revolutionary technologies. In Derek Leebaert (Eds.). Technology 2001, 205-239. Cambridge, MA: MIT Press.

Maler E., El Andaloussi J. (1996). Developing SGML DTDs. From text to model to mark-up. Upper Saddle River NJ: Prentice Hall.

Martínez-Ortiz I., Moreno-Ger P., Sierra, J.L., Fernández-Manjón B. (2006). Using DocBook and XML Technologies to create adaptive learning content in technical domains. International Journal of Computer Science and Applications, 3(2), 91–108.

Megginson, D. (1998). Structuring XML Documents. Charles F. Goldfarb Series on Open Information Management. [Subseries:] The Definitive XML Series from Charles F. Goldfarb. Upper Saddle River, NJ: Prentice Hall PTR.

Mitra N., Lafon Y. (2007). SOAP Version 1.2 Part 0: Primer (Second Edition). W3C Recommendation 27 April 2007. W3C Consortium. Accessed 13.12.2007. Available at: http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

Molloy, D. (2003). Single-source Interactive And Printed Content Publishing Using the DocBook XML Standard. Proceedings of the 2nd International Conference on Multimedia and Information & Communication Technologies in Education, 1800-1804. Badajoz, Spain.

Müller, U., Klatt, M., Dobratz, S., Bahnik, S. (2006). Electronic Publishing at Humboldt University Berlin: Concepts, Tools and Services. Proceedings of ELPUB2006 Conference on Electronic Publishing, 219-228. Bansko, Bulgaria.

Nambiar, U., Lacroix, Z., Bressan, S., Lee, M.L, Li Y. (2002). Current approaches to XML management. In IEEE Internet Computing, 6(4), 43-51.

Necasky M. (2006). Conceptual Modelling for XML: A Survey. TR 2006-3, Department of Software Engineering, Faculty of Mathematics and Physics, Charles University, Prague, 2006. Accessed 7.1.2008.Available at: http://www.cs.cas.cz/ semweb/download.php?file=06-09-necasky3& type=pdf

Necasky, M. (2007). XSEM - A Conceptual Model for XML. Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM2007), 37–48. Ballarat,

Australia. Accessed 19.10.2012 Available at: http://crpit.com/complete/Vol67.pdf

Nurmeksela, R., Jauhiainen, E., Salminen, A. & Honkaranta, A. (2007). XML document implementation: experiences from three cases. In Y. Badr, R. Chbeir, P. Pichappan (Eds.). Proceedings of the Second International Conference on Digital Information Management, 224–229. Los Alamitos, CA: IEEE Computer Society.

OASIS. (2006). DocBook v 5.0 [OASIS 200101]. Accessed 30.10.2012. Available at: http://www.oasis-open.org/specs/

Olson, M.H., Lucas H.C. Jr. (1982). The Impact of Office Automation on the Organization: Some Implications for Research and Practice. Communications of the ACM, 25(11), 838 – 847.

Petride, S., Tarachandani, A., Agarwal, N., Idicula, S. (2011). Managing and Processing Office Documents in Oracle XML Database. The Third International Conference on Advances in Databases, Knowledge, and Data Applications, 89-95. Redwood Shores CA: Oracle Inc.

Polsani, P.R. (2003). Use and Abuse of Reusable Learning Objects. Journal of Digital Information. 3(4), article 164. Available at: http://journals.tdl.org/jodi/index.php/jodi/article/viewArticle/89/88.

Quint, V., Vatton, I. (2004). Techniques for Authoring Complex XML Documents. In Jean-Yves Vion-Dury (Eds.). Proceedings of DocEng'04, 115-123. Milwaukee, Wisconsin, USA: ACM Press.

Raggett D., Le Hors A., Jacobs I. (1999). HTML 4.01 Specification. W3C Recommendation, 24. December 1999. Accessed 7.5.2010 Available at: http://jamsb.austms.org.au/courses/CSC2406/semester3/resources/html/html40.pdf

Rockley, A. (2001). The Impact of Single Sourcing and Technology. Technical Communication, 48, 189-193.

Rockley, A., Kostur P., Manning S. (2002). Managing Enterprise Content: A Unified Content Strategy. Indianapolis: New Riders.

Roisin, C., Vatton I. (1993). Merging logical and physical structures in documents. In Electronic publishing, 6(4), 327-337.

Salminen, A. (2000) Methodology for document analysis. In Encyclopedia of Library and Information Science, 67(30), 299-320.

Salminen, A. (2003). Document analysis methods. In C.L. Bernie (Eds.). Encyclopedia of Library and Information Science, , 916-927. NY, USA: Marcel Dekker, Inc.

Salminen, A. (2005). Building digital government by XML. In R.H. Sprague, Jr. (Ed.), Proceedings of the Thirty-Eighth Hawaii International Conference on System Sciences. Los Alamitos, CA: IEEE Computer Society.

Salminen, A., Kauppinen, K., & Lehtovaara, M. (1997). Towards a methodology for document analysis. Journal of the American Society for Information Science. Special Issue on Structured Information/Standards for Document Architec-tures, 48(7), 644-655.

Salminen, A., Lyytikäinen V., Tiitinen P. (2000). Putting documents into their work context in document analysis. Information Processing and Management, 36(4), 623-641.

Scifleet, P., Williams, S.P. (2009). Practice theory & the foundations of digital document encoding. Proceedings of the 27th Annual ACM Conference on Design of Communication, 213-220. Bloomington, Indiana: ACM.

Sefton, P. (2007). An Integrated Approach to Preparing, Publishing, Presenting, and Preserving Thesis. Proceedings of the 10th International Symposium on Electronic Thesis and Dissertations. Uppsala, Sweden.

Shah, R., Kesan, J., Kennis, A. (2007). Lessons for open standard policies: A case study of the Massachusetts experience. Journal of Information Technology & Politics, 5 (4), 387–398.

Sprague, R.H. (1995). Electronic document management: challenges and opportunities for information systems. MIS Quarterly, 19(1), 29-49.

Thompson, H. S. (2000). XML schema types and equivalence classes reconstructing DTD best practice. XML Europe 2000 Conference.

Walsh, N. (2008). The DocBook Schema Version 5.0. Accessed 20.11.2012 Available at: http://www.docbook.org/

Walsh, N., Hamilton, R. (2010). DocBook 5: The Definitive Guide. O'Reilly Media. Accessed 30.10.2012. Available at http://www.docbook.org/tdg5/en/html/docbook. html/

Walsh, N., & Muellner, L. (1999). DocBook: The Definitive Guide (Vol. 1). Sebastapol, CA: O'Reilly Media, Inc.

YHTEENVETO (FINNISH SUMMARY)

Merkittävä osa organisaatioissa tuotetusta sisällöstä on tallennettuina dokumentteihin. Rakenteiset dokumentit mahdollistavat rakennemääritysten, dokumenttiinstanssien sekä ulkoasumääritysten hallinnoinnin omina sisältöyksiköinään. Tämä on dokumenttien hallinnan näkökulmasta oleellinen rakenteisten dokumenttien ominaisuus.

Samanlaisten dokumenttien joukkoa kutsutaan dokumenttityypiksi. Dokumenttityypit rakentuvat komponenteista, jotka ovat dokumenttityypin rakenneosia. Dokumenttityyppi voidaan määritellä formaalisti skeemassa.

Avointen toimistostandardien kehitys toimistodokumenteille toi XML-kielen käytön organisaatioiden toimistoihin. Tämän tutkimuksen tavoitteena oli tutkia toimistodokumentteja ja löytää niille suunniteltujen XML-skeemojen suunnittelutyöstä yhtenäisiä piirteitä.

Skeemojen suunnittelua toimistodokumenteille voi tukea tarkoitukseen soveltuvan menetelmän käyttö. Tässä tutkielmassa esitettiin menetelmiä, jotka sisältävät skeemasuunnittelua. Lisäksi tutkielmassa raportoitiin havaintoja kahdesta tapaustutkimuksesta, joissa tehtiin skeemasuunnittelua toimistodokumenteille. Tutkimuksen empiirinen osuus suoritettiin toimintatutkimuksena. Ensimmäisessä tapaustutkimuksessa suunniteltiin skeemat IT-tiedekunnan tiedekuntaneuvoston esittelylistalle ja pöytäkirjalle, sekä luotiin kohdeorganisaatiossa käyttöönotettu XML-pohjainen toimistodokumenttijärjestelmä edellä mainittujen dokumenttien laadintaan ja julkaisuun. Tutkimuksen pitkäaikaisuudesta johtuen tapauksen seuranta tarjosi mielenkiintoisia näkymiä XML-kielen käyttöönottoon toimistoympäristössä. Näin ollen tutkimuksen painopiste muuttui ajan myötä tarkastelemaan skeemasuunnittelun lisäksi myös XML-dokumenttien käytön analysointia ja kuvailua toimistoissa. Toisessa tapaustutkimuksessa eläketurvakeskuksen neljälle dokumenttityypille laadittiin XML-skeemat ja niitä testattiin kahdessa erilaisessa asiakirjojen laadintaan tarkoitetussa toimisto-ohjelmassa, joista toisessa oli WYSIWYG-käyttöliittymä ja toinen oli XML-kieltä tukeva lomake-editori.

Merkittävimmät syyt kohdeorganisaatioissa tapahtuneeseen skeemasuunniteluun olivat ongelmat toimistodokumenttien sisällön uudelleenkäytössä sekä dokumenttien julkaisussa. Tutkimus osoitti, että XML-skeemojen suunnittelu toimistodokumenteille vaatii dokumenttien sisältökomponenttien tunnistamista, luokittelua sekä niiden mallintamista. Nämä vaiheet ovat keskeisiä myös referoiduissa ja vertailluissa skeemasuunnittelumenetelmissä. Skeemasuunnittelun nähtiin myös sisältävän dokumenttien ulkoasusuunnittelua sekä dokumenttien laadintaan käytettävän työkalun käyttäjäystävällisyyden huomioon ottamista. Skeemasuunnittelumenetelmien käyttö koettiin hyödylliseksi molemmissa tapauksissa.

ORIGINAL PAPERS

I

TWO METHODS FOR SCHEMA DESIGN FOR INTELLIGENT XML DOCUMENTS IN ORGANIZATION

by

Anne Honkaranta & Eliisa Jauhiainen, 2007

In Witold Abramowicz, Heinrich C. Mayr (Eds.). Technologies for Business Information Systems. Dordrecht, Netherlands: Springer, 173-182.

15 Two Methods for Schema Design for Intelligent XML Documents in Organizations

Anne Honkaranta, Eliisa Jauhiainen

University of Jyväskylä, Finland
anne.honkaranta@it.jyu.fi, raelurja@cc.jyu.fi,

XML markup language provides means for incorporating semantics, i.e. "meaning" of logical content parts residing within documents and other kinds of mainly textual data. Therefore it has become the *lingua franca* for Semantic Web, e-Business applications and for application integration. In order to realize novel, intelligent XML-based document applications in organizations schemas defining the domain-oriented semantics for markup are needed. So far, the potential of XML has not bee fully utilized in organizational documents, due to lack of XML support in common and inexpensive office software. Due to the arrival of XML support on common software such as Microsoft Office 2007 and Open Office 2.0 organizations need knowledge about schema languages and methods by which they may design intelligent XML documents for their needs of document content reuse, content integration across documents, and document content retrieval from Web Services and other source. This paper introduces three schema languages and features of two potential methods for document-oriented schema design.

15.1 Introduction

XML (Bray 2006) was initially developed as a mark-up language for documents for electronic publishing and technical documentation – for managing the reuse, management and publishing of broad and complex collections of documents and publications. XML is becoming a standard to represent any kind of content in any application domain, because it is able to represent any kind of structured or semi-structured documents (Psaila 2000). XML has been widely adapted to enterprise system integration, as e-Business format, and for Web Services. XML is becoming increasingly used for storing and exchanging all kinds of data on the Internet as well (Elmasri 2002).

The use of XML for common organizational documents has been limited due to the fact that XML-capable software has until now been quite specialized and

expensive. Introduction of new XML software for office use also requires training, and may be resisted by users who are used to their common-purpose, sophisticated office tools. Along with the introduction of XML into common-purpose office software such as Microsoft Office 2007 (Microsoft 2006) and OpenOffice 2.0 (Open Office Org 2006) the future office documents are already stored as XML documents, providing a base for developing intelligent XML document applications based on organizational requirements.

Realization of intelligent, domain-oriented document applications relies on the possibility to develop and utilize domain-oriented schemas for XML documents.

This chapter is focused on the XML schema languages and schema design for organizational documents. It presents updated and revised findings based on the paper presented at the BIS 2006 conference (Jauhiainen and Honkaranta 2006). Section 2 introduces three commonly known schema languages – DTD (Maler and El Andaloussi 1996), XMLSchema (Bray 2000) and RelaxNg (Clark 2001). Section 3 introduces two schema design methods; the Maler and El Andaloussi method (Maler and El Andaloussi 1996), and the Kennedy method (Kennedy 2003). The Maler and El Andaloussi method (1996) is generally recognized as the "best practice" for DTD design (Thompson 2000). It has also been incorporated into the RASKE document management methodology (Salminen 2003) and it has clearly influenced the schema design methods developed for the Document Engineering (Glushko and McGrath 2005) and the Unified Method for Content Management (Rockley et al. 2003) approaches. Section 4 discusses the findings and sums up the chapter.

15.2 Schema languages

A schema, such as DTD (Maler and El Andaloussi 1996) or XML Schema schema (Fallside and Walmsley 2004) describes the names of elements and their order for a document type, an attributes adding incremental information about the elements (Bokottaya et al. 2004)

Following subsections introduce the most popular schema languages, the DTD (Bray et al. 2006), and the XML Schema (Fallside and Walmsley 2004) schema for intelligent XML documents. A possible schema language replacing DTDs in future – RelaxNG – is also described. The three aforementioned schema languages are grammar-based languages providing context-free grammar according to top-down production rules in a specified form..

15.2.1 DTD

DTD (Document Type Definition) is a part of XML language definition itself, and perhaps the most commonly known schema language at the time (Marinelli et al. 2004). Figure 1 illustrates the DTD syntax by giving an example of a XML document considering Addressbook and the DTD schema defining markup rules for it.

XML Document (Addressbook)	A DTD for Addressbook document type
<Addressbook> <Card> <Name>Jane Doe</Name> <Email>jane.done@arganization.org</Email> </Card> <Card> <Name>Tom Smith</Name> <Email>tom.smith@company.com</Email> <Phonenumber>+358 14 123456</Phonenumber> </Card> </Addressbook>	<!ELEMENT Addressbook (Card+)> <!ELEMENT Card (Name, Email, Phonenumber?)> <!ELEMENT Name (#PCDATA)> <!ELEMENT Email (#PCDATA)> <!ELEMENT Phonenumber (#PCDATA)>

Fig.1. And Addressbook XML document and a DTD schema for it

DTDs provide a sophisticated regular expression language for defining elements, their ordering and attributes for a document type. It has its own syntax for declaring content models. DTD schema was designed for textual documents and is therefore rather limited by its capabilities to control the values of attributes and elements (Marinelli et al. 2004) In other words, the built-in set of data types for elements in DTD are limited in practice to three: an element contains either parseable data (PCDATA), non-parseable data (NDATA) or no content at all, i.e. element is EMPTY. DTD provides 7 data types for restricting the attribute content. All together DTD supports 10 data types.

15.2.2 RelaxNG

RelaxNG (Clark 2001) schema language is built on two preceding schema languages, TREX (Tree regular expression for XML) and RELAX (Regular language of description for XML).

While the DTD's content model is an expression over elements, the RelaxNG schema defines patterns as expressions over elements, text nodes and attributes. RelaxNG schemas may be declared by XML syntax or by using compact non-XML syntax (Clark and Murata 2001).

The RelaxNG schema defines patterns that a document must match. Patterns may appear as components of the main schema or reside separate from the content model declarations (Clark and Murata 2001). In RelaxNG attribute lists are declared apart from element an pattern which enables the specification of dependencies between elements and attributes and between their values. This feature is a significant difference between RelaxNG and any other schema language. (DuCharme 2004) One of the limitations of the RELAX NG is its inability to define default values for element and attributes (Marinelli 2004).

RelaxNG is considered to be simpler and even as technically superior to the XML Schema language discussed on the next subsection.

15.2.3 XML Schema

XML Schema (Bray et al. 2006) schemas are themselves XML documents, which improves the programmatic processability of the XML Schema schemas. The use of XML syntax for a schema language is not totally a positive feature for a schema: the XML syntax is rather verbose and makes the schemas long and rather difficult to read on their textual form (Marinelli et al. 2004)

XML Schema language is superior to DTDs by its capability of restricting the content types of element and attribute content. It supports even 44 different data types such as date, integer, string, and so on, and also allows regular expression patterns to be used for posting even more strict requirements for element content (Bray et al. 2006). For example, our address book "Mail" element must contain the "@"-sign. XML Schema language has other additional features over DTD schemas, like support for namespaces and possibility to define elements as global or local, or as reusable data types.

In practice only small amount of schemas use the advenced features provided by XML Schema language. This means that majority of the schemas found in the Web may be expressible by DTD schemas, even though the modeling power of the XML Schema is notably higher. (Bex et al. 2005) Figure 2 illustrates the XML Schema schema and its counterpart as RelaxNG schema for the Addressbook XML document.

XML Schema	RELAX NG
<?xml version=1.0" encoding="ISO-8859-1" ?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"> <xs:element name="Addressbook"> <xs:complexType> <xs:sequence> <xs:element name="Card" maxOccurs="unbounded"> <xs:complexType> <xs:sequence> <xs:element name="Name" type="xs:string"/> <xs:element name="Email" type="xs:string"/>	<?xml version=1.0" encoding="ISO-8859-1" ?> <element name="addressBook" xmlns="http://relaxng.org/ns/structure/1.0"> <oneOrMore> <element name="Card"> <element name="name"> <text/> </element> <element name="Email"> <text/> </element>

```
<xs:element name="Phonenumber"
type="xs:string" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element></xs:schema>
```

```
<optional>
<element name=”Phonenumber>
<text/>
</element>
</optional>
</element>
</oneOrMore>
</element>
```

Fig.2. Example of XML Schema and RelaxNG schemas for the Addressbook document type

XML Schema is generally considered as complex (van der Vlist 2002). However, XML Schema is the most widely used among all the schema languages. Possible reason for the popularity of XML Schema language is that it is supported by a number of popular software. Microsoft, for example, relies on XML Schema schemas for custom schema use on its Office 2007 software. Also a number of XML development environments provide support for XML Schema schema design (f.ex., Altova XMLSpy; www.altova.com). More profound comparisons on schema languages may be found in Lee and Chu (2000) and Jauhiainen and Honkaranta (2006).

15.3 Methods for schema design

This section describes the two potential schema design methods; the Maler and El Andaloussi (1996) method and the Kennedy (2003) method.

15.3.1 The Maler & El Andaloussi method

Maler and El Andaloussi (1996) method has been considered as a best practice on the field (Maler & El Andaloussi 1996, Thompson 2000). The methodology (Maler and El Andaloussi 1996) can be divided in three phases, which are:

- Document analysis
- Document type modeling, and
- Producing a document analysis report

The document analysis phase consists of the three steps that are: 1.) Identifying potential components, 2.) Classifying components, and 3.) Validating the requirements.

Document component is broader by its content than an individual element consisting of text. For example, in our Addressbook a possible document component is a Card-element. Document component should be a logical unit of document content which may, for example, be reused from elsewhere or need to

be presented to a human reader as a unit. For example, a "list of attendees" is a potential component for a memo document type rather than a "attendee_name".

Step 2 considers classifying the *potential components* with respect to their content-based and structural similarities. For example, "list-of-attendees" and "list-of-items" may be broadly classified as a list component types. The third step considers studying the available schemas for similar kinds of document types and deciding if a domain-oriented schema is needed, or if a related, existing schema or its components may be tailored for the purpose at hand.

The document type modeling phase consists of seven steps, which are 4.) Selecting semantic components, 5.) Building the document hierarchy, 6.) Building the information units, 7.) Building the data-level elements, 8.) Populating the branches of the document hierarchy, 9.) Declaring connections between elements, and 10.) Validating the design.

In step 4 the component lists are revisited for dropping out all but necessary components. Step 5 considers modelling the components into *document hierarchy* model, which represents main components for the document type. Figure 3 illustrates a document hierarchy model as Elm-diagram for the Addressbook document type. In the figure the components are depicted by rectangles. Three dots underneath components depict that that the content model is represented in another diagram. The document hierarchy ends to "cloud" symbols depicting the portions of more-detailed level content models. "+" , "?" and "*" are occurrence symbols indicating that a component may repeat from one to many, zero to one, or from zero to many times.

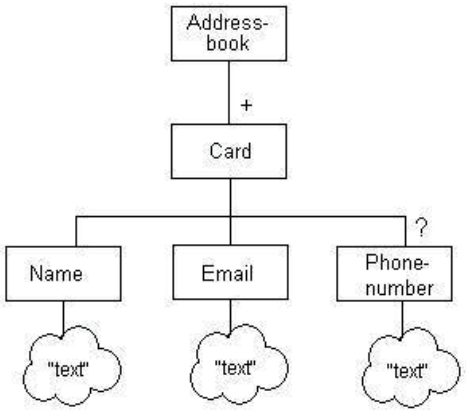

Fig 3. An Example of document hierarchy model for an addressbook.

Steps 6 and 7 concern the design of Information unites and data-level components. Information units are groups of logically related elements that should be modelled as a whole; such as list heading and list item, or a figure and its caption. Data-level elements consider markup within text, such as if concepts need to be separated form other text in order to specify differing layout for them. Information units and data-level elements are revisited on step 8 for populating the branches of the tree

diagrams with information units and data-level elements defined. Step 9 considers the identification of requirements for cross-references and links within the document type and across documents and other content sources.

Finally in step 10 the whole design is revisited. This step ends the second phase of the Maler & El Andaloussi method (1996). The last phase of the method considers producing the document analysis report.

After the schema design the actual schema is declared by using the schema language selected for the purpose.

15.3.2 The Kennedy method

Kennedy (2003, 92-94) lists nine rules for defining XML schemas. The rules consider:

1. Data analysis
2. Coding the data
3. Use of container elements
4. Use of group or block elements
5. Use of subelements for multi-value data
6. Avoiding mixed content
7. Use of meaningful names
8. Correct use attributes and elements, and
9. Reviewing the design.

The first rule, data analysis, follows the Maler & El Andaloussi method. It consists of a) identifying the document components and b) classifying them into logical groups. The data analysis in the Kennedy method covers additional tasks with regard to the Maler and El Andaloussi method. These consider deciding upon the recurrence or optionality of a document component, and about its importance.

The second rule is about coding the data. This rule considers if a component is actually a logical content component or whether it is merely a presentation kind of component, in which case the component should be considered on the style sheet design rather than on content design.

Third rule is about using “collection” elements. By the collection element Kennedy refers to element that may occur multiple times at the same level, such as list items. If the aforementioned elements are identified, one should create a container element for them. For example, we may define a “list” element s a container for list items. Kennedy declares that use of collection elements makes the XML more human readable and easier to process by recursive loops, such as the “for-each” element use in the XSLT transformation language. Another example of a collection element is the “author” element; if it repeats multiple times within a document type, one should define container element “authors” to group the “author” elements.

Kennedy’s fourth rule is about remembering that XML markup forms commonly a tree structure within documents. The tree should be wide instead of

tall. The fifth rule states that the best way to represent multi-valued data is via subelements.

Sixth rule exhorts that one should avoid mixed element content and the seventh one that the element names should be meaningful. The eight rule emphasizes the correct use of elements and attributes. According to Kennedy, XML document schemas commonly contain lots of elements and fewer attributes than elements whereas schemas for data-oriented XML documents and data transfer may consist even greater amount of attributes than elements. Therefore, the application domain and document type may affect the schema design considerations.

The final rule is about recognizing that defining the schemas is similar to database design in respect that the design process is commonly iterative one by its nature. As database schemas are defined by inserting test data into table structure defined, the XML document schemas are tested by writing example documents, and testing de design against style sheet definitions and content manipulation needs.

Kennedy's method is similar to Maler and El Andaloussi method also by its modeling notation; both of the methods use the so called "Elm" models for visual schema design. Figure 3, the document hierarchy model provides and example of the Elm modeling notation.

15.4 Discussion and conclusion

Although XML is currently used as a standard for any kind of content, its use on common organizational documents has been limited by the lack of inexpensive software with XML support. However, along the introduction of novel office software such as OpenOffice 2.0 and becoming Microsoft 2007 the situation changes in a radical way. Organizations may start benefiting from the use of intelligent XML documents, which provide means for content integration, retrieval and reuse across document collections and external content sources such as databases or WebServices. Realization of the intelligent XML document applications rely on the domain- and application-oriented markup. Schema provides a base for XML processing, content filtering, and content organization. Therefore, schema design is abase for document-oriented XML document management and its importance should not be overlooked.

There are several schema languages for XML document schemas. Perhaps the most common one is W3C's XML Schema - a rich schema language with multiple features and datatypes. However, XML Schema definition is considered as complex. Even though many applications support the use of XML Schema, the expressive power of DTD may be quite enough for defining narrative, document-oriented document types. XML Schema language features and the capability to define strict rules for XML content on a data-type level may be needed for example for technical documentations consisting lots of measures and measure units, or for document types that exchange content with databases or applications with stricter requirements for data typing.

Even if an organization would not need the sophisticated features of XML Schema schema language, many organizations may use it due to its broad support on software tools. It remains to be seen if the RelaxNG as an "improved version"

of the DTD schema is capable to challenge the XML Schema language as it seems to be gaining more software support. The technical capabilities of the RelaxNG and the XML Schmea language in terms of expressiveness and the ability data typing are slightly different, yet sufficient for document schemas.

This chapter reviewed the Maler and El Andaloussi (1996) method that was originally defined for DTD design and the Kennedy (2003) method targeted for RelaxNG schema design. There are many similarities between the Kennedy and the Maler & El Andaloussi method. The Kennedy method adopts the first two steps of the Maler and El Andaloussi method. The Kennedy's method also applies the Maler and El Andaloussi method for defining container elements, which are typical for document-oriented XML document types. Both Kennedy and Maler and El Andaloussi emphasize the importance of domain-oriented, meaningful element names. Both methods also remind DTD or schema designers that schema design is usually an iterative process and the document structures should be tested before implementation.

It may be concluded that both the Kennedy and the Maler & El Andaloussi method may be adoptable for organizational XML document type schema design. More recent schema design methods, such as the one included in the Document Engineering approach (Glushko and McGrath 2005) and the one utilized in the Unified Content Strategy (Rockley et al. 2003) are influenced by the Maler and El Andaloussi method. Even though the DTDs are nowadays often replaced by XML Schema schemas the Maler & El Andaloussi method may provide a solid starting point for organizational schema design, or one for developing methods for novel design requirementsTherefore schema design is a base for document-oriented XML document management and its importance should not be overlooked.

References

1. Bex G.J, Martens M., Neven F. & Schwentick T. (2005) Expressiveness of XSDs: From Practice to Theory, There and Back Again. In Proceedings of the Fourteenth International World Wide Web Conference, Chiba, Japan, May 2005, pp-.712–721.
2. Boukottaya A., Vanoirbeek C., Paganelli F. & Abou Khaled O., Automating XML documents transformations: a conceptual modelling based approach. 1st Asia-Pacific Conference on Conceptual Modelling, ACSW 2004, Dunedin, New Zealand, January 2004.
3. Bray T., Paoli J., Sperberg-McQueen C.M & Maler E. Extensible Markup Language (XML) 1.0 (Third Edition) [online], W3C Recommendation. <http://www.w3.org/TR/ 2004/REC-xml-20040204/>
4. Clark J. (2001) The design of RELAX NG. [Online]Available at http://www.thaiopensource.com/relaxng/design.html [16.1.2006]
5. Clark J. & Murata M. RELAX NG Specification.
6. DuCharme B. Documents vs. data, schemas vs. schemas. XML 2004 conference, Washington D.C.
7. Elmasri R., Wu Y-C., Hojabri B., Li C. & Fu J. Conceptual modeling for customized XML schemas. In 21st International Conference on Conceptual Modeling (ER), Tampere, Finland, volume 2503 of Springer LNCS, Springer, 2002, pp. 429–443.

8. Fallside, D. C., & Walmsley, P. e. (2004, 2 May). XML Schema Part 0: Primer. 2nd Edition. W3C Recommendation, 28 Oct. 2004. Retrieved 16 June, 2005, from http://www.w3.org/TR/xmlschema-0/
9. Glushko, R.J. & McGrath, T. (2005) Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services. MIT Press.
10. Jauhiainen, E. & Honkaranta, A. (2006) A Review on XML Document Schemas and Methods for Schema Design. In the Abramovicz, E. & Mayr, H. (Eds.) Proceedings of the 9th International Conference on Business Information Systems. Series of the Gesellschaft fyr Informatik. Bonn. Vol P-85. 201-210.
11. Kennedy D. Relax NG with XML data structures. National Advisory Committee on Computing Qualifications, Palmerston Nth, July, 2003.
12. Lee D. & Chu W. Comparative analysis of six XML schema languages. SIGMOD Record, 29(3):76–87, 2000.
13. Maler E., El Andaloussi J. Developing SGML DTDs. From text to markup. Upper Saddle River NJ: Prentice Hall, 1996.
14. Marinelli P., Sacerdoti Coen C. & Vitali F. (2004). SchemaPath, a minimal extension to XML schema for conditional constraints. In Proceedings of the 13th International Conference on the World Wide Web (WWW13), New York City, U.S.A. ACM, 2004.
15. Microsoft. (2006). Microsoft Office 2007. Available at http://www.microsoft.com/office/preview/default.mspx [14.9.2006]
16. OpenOffice Org. (2006). Open Office Organization Home Page. Available at http://www.openoffice.org/ [20.7.2006]
17. Psaila G. ERX: A conceptual model for XML documents. In In Proc. of ACM Symposium on Applied Computing (SAC), Villa Olmo, Italy, 2000, pp. 898-903.
18. Rockley, A, Kostur, P., & Manning, S. (2003). Managing Enterprise Content: A Unified Content Strategy. U.S.A.: New Riders.
19. Salminen A. Document analysis methods. In Bernie C.L. (Ed.) Encyclopedia of Library and Information Science, Second Edition, Revised and Expanded. New York: Marcel Dekker, 2003, pp. 916-927.
20. Thompson H. S. XML schema types and equivalence classes reconstructing DTD best practice. XML Europe 2000 conference, 2000.
21. van der Vlist E. XML Schema languages. In Proceedings of XML Europe 2002, Barcelona, Spain, May 2002.

II

ASPECTS ON XML DOCUMENT CONTENT REUSE IN ORGANIZATIONS

by

Eliisa Jauhiainen & Anne Honkaranta, 2007

In Weide Chang, James B.D. Joshi (Eds.). Proceedings of the 2007 IEEE International Conference on Information Reuse and Integration (IEEE IRI-07), Las Vegas, 588-593.

Aspects on XML Document Content Reuse in Organizations

Eliisa Jauhiainen
University of Jyväskylä
raelurja@jyu.fi

Anne Honkaranta
University of Jyväskylä and SYSOPENDIGIA Plc
anne.honkaranta@[it.jyu.fi; sysopendigia.com]

Abstract

Designing the reuse of information residing in documents is more complex than for information in databases. Document content is designed for humans and organized with regard to communicational purposes for organizational work. In addition, content organization within documents is affected by the requirements of multi-channel publishing and layout design for content presentation. Efficient content reuse in organizational documents requires that the ways the content is created and stored within and across documents and other content resources, such as databases, should be identified. XML provides technological means for document content reuse. The designers of XML document production need to be aware of the requirements for content reuse in order to facilitate reuse across XML documents and other content sources. This paper describes the content reuse identified in two e-Government cases and proposes a classification of document content reuse types. The impacts for XML schema and document production design are also discussed.

1. Introduction

Content reuse refers to the practice of using existing pieces of content for developing new novel content. The majority of reusable content in organizations is text-based, but any type of content can be reused [1]. As much as 80-90 % of content used and produced in organizations may be considered as *documents* [2] or similar non-structured or semi-structured text-based content, which is primarily targeted for human consumption [3][4]. A great deal of the organizational content in documents is reusable. Therefore document content reuse is an important part of organizations' content management strategy. It may provide dramatic improvements for organizational work.

Document content reuse in general may concern the reuse of an existing document as a base for a novel one, or simple copy and paste –operations of text from other content sources into a document to be produced. Manual copying and pasting is time-consuming and prone to error. Content reuse may be automatized by using computer programs, which requires a through examination of content reuse.

XML [5] has become a standard for presenting virtually any kind of information, from processes (e.g., [6], [7]) to Web Services [8], and for Enterprise Application Integration [9]. XML documents' logical structure, i.e. the names of XML document content components may be formally described by a schema such as DTD [10] [5], or XML Schema [11]. XML documents contain a mixture of document content and XML markup defined by the schema, which allows the identification of content components for content reuse. In order to support content reuse, XML documents and their content sources have to be analyzed for schema design.

The need for content reuse in organizational documents is obvious, yet research on content reuse revealing the types of content reuse across documents and other content resources in organizations is scarce. Even though XML has been adopted widely, research on XML documents in organizations has largely been overlooked due to lack of XML support in common office software. The newest office applications such as Open Office 2.0 [12] and Microsoft Office 2007 have adopted XML as their native data format, as Open Document Format [13] and ECMA 376 Office Open XML [14] respectively, thus providing the technological means for document content reuse as XML. Research illustrating the requirements for content reuse and its impact on XML schema design, as well as in XML document production, is therefore needed.

This paper describes the types of document content reuse that were identified in two e-Government organizations, and proposes a classification of reuse types. In this paper we analyze the requirements for XML schema design and XML document production. The paper is organized as follows. Section 2 discusses related research on content management, content reuse, and XML. Section 3 provides examples of document content reuse and presents a classification of reuse types in organizations. Section 4 analyzes the requirements for XML document production and schema design. Section 5 concludes the paper and provides avenues for further research.

2. Content reuse in organizations

2.1 Content and component

Enterprise content management (ECM) is a research area concerned with both social and technical aspects of managing organizational content. The "content" covers a wide spectrum from documents to web content intended primarily for human consumption, as well as other content sources related to them. Content management may be realized by utilizing files in a variety of formats, databases, XML, and other technologies [15].

While information resource and architecture design [16] in enterprises focuses on identifying the information resources and describing the information landscape from the viewpoint of logical groupings of information items, the design of information for documents is more complex. Rather than following a pure logic of information itself, the documents are constructed to fulfill communicational purposes in complex work settings [17].

Content to be managed within organizations may come in many forms; it may be loosely or tightly structured and it may have a large or small grain size. The concept of grain size refers to the breadth of information content and the physical storage space it requires. For example, a long document, such as a paper machine operating manual, may have a large content grain size, and a snippet of a Web page content may have a small grain size (e.g. [15]). There is no universal agreement on what a reusable piece of content should be called or what its grain size should be. In e-Learning, a reusable piece of learning content is called a Learning Object (LO; [18]). An LO should contain one topic as a stand-alone portion of text or multimedia [16]. In ECM, a reusable piece of content within a document or a web page may be called a content block, content item or component [19].

This paper adopts the term *component* from Boiko [19] and Rockley [1] as signifying a reusable piece of content which may refer to a document or a web page, but in most cases addresses content with a smaller grain size than these. A class of documents that are similar in content and purpose is considered a document type. Content for a document may be created by picking up existing content components and combining them in a correct and suitable order ([20], [1]).

2.2 Document and component reuse

Document content reuse may take a number of forms. For example, Guerrieri [21] has identified four types of document reuse. These are 1) Document content reuse, 2) Document structure reuse, 3) Document style reuse, and 4) Document rendering reuse. Document content reuse concerns the reuse of document content. Document structure reuse means that the generic structures of the document, such as titles, paragraphs, and appendices are reused. Document style reuse concerns the reuse of style information, such as font size or face. Document rendering reuse refers to reusing document content and style for a different type of displaying media [21]. The Guerrieri classification does not explicitly point out if content reuse concerns document content as a whole or only a portion of it. The classification does, however, regard the kinds of reuse that are essential in designing templates for document types regardless of being utilized in a Web content management system or in office software.

Manual copying and pasting is a common way of reusing document content. Making reuse more effective and automated is possible with content components. In order to reuse content components, one must both identify them and understand the relationships between them [21, 22]. The Unified Content Strategy by Rockley [1] manifests the need for reuse, and presents a method for Web content reuse by utilizing components and XML. Boiko's approach to Web content reuse is similar, but relies largely on the utilization of metadata for content composition. Yet, what constitutes a component and how one may be identified remains largely a domain-specific issue. To exploit the aforementioned approaches, examples of potential components and their types in organizational settings are required. Therefore, examples from real life organizations and document component types in them may help schema and reuse designers.

XML supports content reuse in multiple ways. It provides means for identifying content components within documents for reuse. It also allows separating content from layout, thus supporting the first three of the four reuse types defined by Guerrieri [21]. XML also has a number of companion languages, such as XSL and XSLT that may be used for content rendering reuse and content filtering [1].

3. An example and classification of document reuse types in organizations

This section proposes the classification of document content reuse types based on observations in two e-Government organizations; The Faculty of Information Technology (FIT) in the University of Jyväskylä and The Finnish Centre for Pensions (FCP). At FIT the reuse was observed as a part of a project that developed an XML publishing system, and at FCP as a part of a feasibility study concerning the adoption of XML for document production.

3.1 Document components in the two organizations

The Faculty of Information Technology (FIT) at the University of Jyväskylä has three departments with approximately 2200 students and 200 employees. FIT is administered by the Faculty Council. The personnel of the faculty office produce **agendas** and **memorandums** (memos) for and about the council meetings. These

documents deal with numerous administrative issues, such as study requirements, recruiting, granting diplomas and so on. Agendas and memos were produced with Microsoft Word but their preparation was considered problematic. A great deal of content was copied manually from a number of old documents. This process was time-consuming and prone to error. In addition, printing the agendas and memos took time because they were saved in several different files, each agenda item and appendix in one file. These files were stored separately to allow items to be prepared simultaneously by several people. Also Web publishing was laborious due to multi-phased transformation and file transfer processes, and separate text files. Time and resources for Web publishing were insufficient even though it was considered important.

An agenda for a Faculty Council meeting consists of a front page, a table of contents listing the items for a meeting, and a number of separate items to be dealt with. Each item is prepared separately, stored, and printed as separate files. Items were gathered as an agenda order for printing only right before the meeting, when all the items were finished and the table of contents was updated according to the items and their ordering for the meeting at hand. After the meeting, the decision text was appended to each item for producing a memo. Each item in an agenda has its own component structure, as visualized in Figure 1.

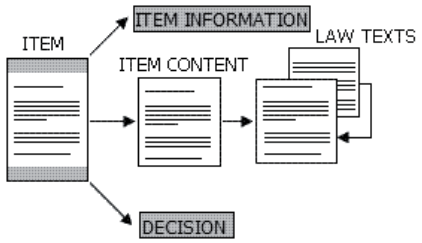

Figure 1. Document components of item

Each item comprises a unique component containing item information with a standardized layout and content structure. Item information includes the date of the meeting and a list of council members for marking who was present and absent when the issue was decided, among other things. The decision component of the document contains elements typical of memos, such as the decisions made in the meeting.

Each item content component contains a combination of novel and reused text previously copied from existing pieces of legal texts. These legal texts occur repeatedly within the agenda and memo items. For example, if an employee of the faculty applies for absence for a leave, a paragraph containing legal phrases related to norms of applying for an absence is reused. These phrases were manually copied and pasted from old agenda items. After a meeting, the agenda was transformed into a memo by appending meeting notes and decisions into the content, among other modifications.

The Finnish Centre for Pensions (FCP) acts as a central body for numerous private pension institutions in Finland. There are nearly 400 employees working at FCP, primarily lawyers, pension schema experts, and pension register system experts. The FCP has been active in developing its content management, which has also included document redesign [22].

A myriad of administrative documents (i.e. records) are produced at FCP, such as statements, decisions, and directives. Similar to FIT, a number of FCP document types involve content that is either directly copied or derived from legal texts. For example, a **decision document** may contain portions of legal texts related to laws and norms regulating how and when a pension may be obtained, and a **directive document** may contain texts derived from normative legal texts, but re-written to enable easier comprehension for laymen readers.

The FCP had already facilitated the reuse of legal text portions re-formulated for laymen by collecting them into a database. The FCP had also founded an address database containing the names and addresses of the organizations with which they mostly communicated. At FCP, the address database and the legal text database were integrated into document production by a plug-in application which was built into Word software and the document templates utilized in it. Since the lawyers were very strict with pension-related communications, they had also prepared hundreds of different document type templates and variations of these consisting of model texts for various standard cases to be dealt with. Furthermore, each template contained placeholders showing the changeable text content to be edited and checked while preparing the legal documents. While the documents printed on paper commonly had a header and footer, the layout of which was prepared according to a national layout standard, parts of the information content in the header, and often the entire footer, was removed when the documents were delivered via Inter- or Intranet. One reason was that the information found in the footer was already available on the Inter- or Intranet site.

3.2 The classification

Four different types of reuse may be identified in both FCP and FIT documents. The four types of document content reuse are the following:

1) **Document content reuse as a whole.** A document content as a whole may be reused as a part of another document.
2) **Document component reuse for multi-channel delivery.** Reusable components are utilized to modify content for a certain medium.
3) **Reuse of repeating text portions.** In an organizational domain, there may be repeatable

phrases and/or text portions, which are reused across multiple documents.

4) **Reuse from databases.** There may be both internal and external databases or services that provide content to documents.

The use of agendas as a base for memorandum documents at FIT represents **document content reuse as a whole**. Similar kinds of patterns may also be identified in business contexts. For example, an order for a product containing information about the items to be delivered may be appended into an invoice. A number of question-answer document pairs may share the same content portions as well.

Document component reuse for multi-channel delivery considers tailoring the content for multi-channel distribution. Delivery via a different medium commonly affects both layout and content. Both paper prints of the document and its online equivalents need to have some kind of layout for human consumption. However, a document delivered via the Internet may require a slightly different kind of layout to be more comfortably read on the screen, and also the content might be slightly different than in its paper counterpart. In many organizations, such as FCP, document components such as the header and footer are only required in the paper print versions of documents. In addition, the header text components may be saved in separate files and reused for multiple documents at the advent of publication.

Reuse of repeating text portions concerns the reuse of repeating portions of legal texts in our examples. At FIT the legal texts for agendas and memos were previously copied and pasted manually from one document to another. In the novel XML publishing system, the legal texts were collected into an XML text database for reuse. At FCP the re-written legal texts were stored into a relational database and accessed via the Word plug-in.

Reuse from databases refers to the reuse of highly structured text. Content can be imported from databases or other external sources, as well as from other documents. At FCP, the names and addresses of document recipients were imported from an address database, which was also plugged into the Word document templates. At FCP, the imported names and addresses already had a database structure due to which the database delimiters had to be transformed into line feeds or spaces when the texts were imported to a document in the editing phase. The example illustrates that one document type may have a number of content sources and involve multiple types of content reuse.

4. Requirements for XML schema design

A schema is a formal model for an XML document type. In XML schema design, analysis of the domain may reveal connections and similarities that have not been realized before between documents, such as the requirements for reuse in the two case organizations.

4.1 Reuse types and their impact on schema design

Document content reuse as a whole can be supported by saving the schema components required in multiple document types into one file, and defining the document components which are needed for only one document type as optional, whereas those that are needed in all document types are defined as mandatory. This kind of schema may be called an "interchange schema" [10]. At FIT only one schema was used to define both agenda and memo document types; the schema components that only belonged to the memo document type were grouped into container elements and defined as optional in the agenda/memo interchange schema.

Document component reuse for multi-channel delivery may be supported through a similar approach: by appending optional schema components to interchange schema. Optional schema components may be grouped into a container element that is used only for document delivery via specific publishing media. For example, at FCP, the footer of a document was necessary only in the paper format; therefore the footer element may be defined as an optional component in the interchange schema. The interchange schema is independent from single document types and may be efficiently reused by several document types. This makes schema versioning and maintenance easier, but poses a multitude of requirements for document processing to prevent erroneous content. For example, if the authoring software is capable of hiding the footer part of a document when necessary, the author cannot add text into it accidentally, even though the schema would allow it.

Reuse of repeating text portions in our classification concerned the reuse of pieces of legal texts that were imported into documents. At least one container element has to be added into a schema to hold a position to the imported text portions. At FIT, the legal texts for agenda items were imported from external XML document.

Sometimes the reusable content component contains a structure itself, such as XML markup or database delimiters. This is an example of **reuse from databases**. Instead of importing some unstructured text into an element, a group of elements may be imported into the content reuse process. For schema design, this exhorts the use of namespace declarations for avoiding possible collisions in element names. For avoiding the collisions both documents and schema components must be analyzed. Also the host application for reuse has to have means for import operability.

The four reuse types form the basis for two schema design strategies. The first strategy is to design interchange schema to cover multiple XML document types. This schema may have optional components for

different document types and publishing purposes. The second strategy is to assemble a schema file from a collection of schema components either manually or dynamically. Assembly may also be carried out by using <import> declarations in the "host" schema file. Each organization chooses its own strategy based on the existing content and the needs they have.

4.2 XML document production

Schemas for XML documents must meet the requirements set by both human consumption and software for document authoring. Glushko and McGrath [20] even perceive documents as interfaces. This means that an XML is not merely a data exchange mechanism.

Software selected for document authoring may bring about significant limitations and requirements for XML schemas. In the case of FIT, Microsoft InfoPath was selected as the software for XML document authoring. Each element defined in the agenda/memo schema had a corresponding control in the InfoPath form template. The most common control was a text box in which the users would type and edit texts to be stored in an element.

At FIT, the InfoPath software interface with its form controls required several changes to the schema used for agendas and memos. For example, the InfoPath template did not provide a suitable control for presenting tabular layout structures required in the document layouts in a convenient way for the end users. Another constraint arose from text inline formatting requirements. In the first version of the agenda/memo schema, the text-paragraph sections were defined by only one element which was tied into a text box control in the InfoPath form. Therefore, the end users could only define a bold or italic font face to the text paragraph as a whole. The inline formatting could have been supported by using a Rich text control in InfoPath, but this control required for the text paragraphs to be defined as an HTML text portion in the schema. The optional element group used in the schema allowed one paragraph of text to contain a text-paragraph, a list or a table. It had to be linked to the "Optional Choice Group" control in the InfoPath form. However, using the Optional Choice Group control made the form look confusing and messy from the end users' perspective, and therefore the schema had to be re-designed.

At FCP, document type templates utilized in Word carried out a number of tasks related to content reuse. The plug-in applications for importing content from the address and the legal text databases were implemented as a document type template which also acted as a container for model texts. The plug-in facilitated content reuse but, at the same time, required that the plug-in application be updated by tailoring the template code each time the FCP renewed its version of Word.

When people use XML documents, the XML schema design principles for the data exchange do not apply. Human consumption and the software used set new kinds of requirements for the schemas. The examples show that layout features and the host/authoring software for document production, as well as content reuse have to be considered in schema design.

5. Discussion and conclusion

As the majority of organizational content is stored in documents, document content reuse is essential for organizational work. XML supports content reuse by providing a mechanism by which content components may be marked up for automated reuse. Even though many scholars, such as Boiko [19] and Rockley [1], have emphasized document content reuse, there is a lack of knowledge about which kind of content is reused in the organizations and how.

Guerrieri [21] has defined four kinds of document content reuse. These reuse types mainly concern document reuse as a whole. This corresponds to the first type of content reuse in our classification, "document content reuse as a whole". This paper proposed for reuse to be considered more thoroughly and suggested three additional kinds of document content reuse. The impacts on schema design were also discussed.

Our classification of reuse types perceives both document and schema reuse. Instead of defining one schema for each XML document type, the schema designer may use the interchange schema strategy or compose a schema from several schema components. Some of the schema components may be optional to serve multi-channel publishing. Schema component design requires that the document types, content reuse, and the layout, as well as the software features be analyzed. Therefore, schema design is a demanding and complex task affected by the grain size of the components and several domain-specific requirements as presented in Figure 2.

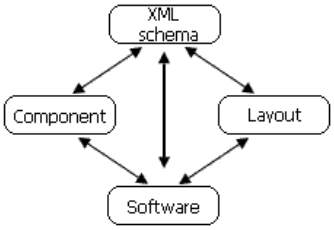

Figure 2. Aspects of XML document production and reuse

Software is used to support human comprehension of documents; therefore the layout features and authoring support for reuse must be considered. These may impact schema use and, consequently, schema design as well. A schema can be divided into several schema components forming a library of small schema components or it may

be designed as an interchange schema for multiple document types.

Experiences from the two case organizations allowed the formulation of the four document content reuse types introduced. All of them affect schema design and design strategy selection, as well as the domain and software used for document production. Unlike data-centric XML files, XML documents are not a simple data exchange mechanism, but also require for the layout and authoring process to be taken into consideration.

If the amount of documents in an organization is great, XML document implementation is not a simple task. XML schema design must be carried out with care so that relevant components and their interrelated connections support effective reuse. There is a need for schema design methods that support the identification and classification of content components within documents and related schemas for document types. Such a method should also describe the process of identifying these components in a domain. One potential avenue for future research is to scrutinize and potentially adopt the Unified Content Strategy [1] or the Document Engineering [20] method to incorporate support for schema design strategies and content reuse more thoroughly.

Acknowledgements

This research would not have been possible without an active collaboration with the Faculty of Information Technology (FIT) at the University of Jyväskylä, Finland, and the Finnish Centre for Pensions (FCP). We are especially grateful to Mrs. Sanna Hirvola and Mrs. Eija Ihanainen from FIT, and Mrs. Riitta Ahovaara, Mrs. Tarja Gauriloff, Mr. Samuel Rinnetmäki and Mr. Jarkko Nikulainen from FCP for their collaboration. Mr. Tuomo Peltola from SYSOPENDIGIA, Plc gave his support to the FIT XML development projects.

References

[1] A. Rockley, P. Kostur and S. Manning. “Managing enterprise content: a unified content strategy”. *Pearson Education*, 2002.

[2] R.H Sprague. “Electronic document management: challenges and opportunities for information systems” MIS Quarterly, Vol. 19(1), 1996, pp. 29-49.

[3] A. Salminen. “Document analysis methods”. In Bernie C.L. (Ed.) Encyclopedia of Library and Information Science, second edition, revised and expanded. New York, Marcel Dekker, 2003, pp. 916-927.

[4] A. Honkaranta. “From genres to content analysis. Experiences from four case organizations.” *University of Jyväskylä*, 2003.

[5] Bray, T., et al., Extensible Markup Language (XML) 1.1 (2nd ed.). W3C Recommendation 16 Aug 2006. 2006, W3C Consortium. Retrieved from: http://www.w3.org/TR/2006/REC-xml11-20060816/. [2006 1 Sept.]

[6] ebXML, ebXML Consortium. 2005, OASIS (Organization for the Advancement of Structured Information Standards). Retrieved from: http://www.ebxml.org. [2005 10 June]

[7] RosettaNet, RosettaNet. 2005, RosettaNet Consortium. Retrieved from: http://www.rosettanet.org/RosettaNet/Public/PublicHomePage. [2005 10 June]

[8] R. Chrinnici, et al., Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft 3Aug. 2005. 2005, W3C; World Wide Web Consortium. Retrieved from: http://www.w3.org/TR/2005/WD-wsdl20-20050803/. [2005 10 Nov.]

[9] D. Linthicum D., *Enterprise Application Integration*. 2000, Reading, Massachusetts: Addison-Wesley Longman, Inc.

[10] E. Maler and J. El Andaloussi. “Developing SGML DTDs. From text to markup.” *Prentice Hall*, New Jersey, 1996.

[11] Fallside D.C & Walmsley P. (2004) XML Schema Part 0: Primer Second EditionW3C Recommendation 28 October 2004. W3C Consortium. Retrieved from http://www.w3.org/TR/ xmlschema-0/ [12.4.2006]

[12] OpenOffice Org., *Open Office Organization Home Page*. 2006, OpenOffice Org., Retrieved from: http://www.openoffice.org/ [2006 20 July]

[13] Open Document Alliance, Open Document Format Alliance for ODF (ISO/IEC 26300:2006 standard for Office documents). 2006. Retrieved from: http://www.odfalliance.org/. [2007 15 Feb]

[14] *ECMA-376. Office Open XML File Formats*. 2006, European Computer Manufacturers Associations. Retrieved from: http://www.ecma-international.org/publications/standards/Ecma-376.htm. [15 Jan 2007]

[15] A. Honkaranta and P. Tyrväinen, *Managing Converging Content in Organizations (in print)*, in *Encyclopedia of Information Science and Technology*., M. Khosrowpour (ed.). 2008, Idea Group Publishing, Inc.: Hershey.

[16] R. Evernden and E. Evernden. "Third-generation information architecture". Commun. ACM, 2003. 46(3): p. 95-98.

[17] S. Fowell. “Bridging the Gap between Information Resource Design and Enterprise Content Management”, Lecture Notes In CS; Vol. 2555, 2002, pp. 507-515.

[18] P.R. Polsani, "Use and Abuse of Reusable Learning Objects". Journal of Digital Information, 2003. 3(4): p. 10.

[19] B. Boiko, *Content Management Bible*. 2002, New York, U.S.A: Hungry Minds, Inc. 924

[20] R.J. Glushko and T. McGrath. “Document engineering. Analyzing and designing documents for business informatics & web services.” *The MIT Press, Cambridge Massachusetts, London, England, 2005*.

[21] E. Guerrieri, “Software Document Reuse in XML,” Proceeding of IEEE fifth International Conference on Software reuse, 1998, pp. 246-254.

[22] A. Honkaranta, T. Peltola and A. Salminen. Challenges in the Redesign of Content Management: A Case of FCP. International Journal of Cases on Electronic Commerce, Vol. 1(1), 2005, pp. 53-69.

III

XML DOCUMENT IMPLEMENTATION: EXPERIENCES FROM THREE CASES.

by

Reija Nurmeksela, Eliisa Jauhiainen, Airi Salminen & Anne Honkaranta

In Youakim Badr, Richard Chbeir, Pit Pichappan (Eds.). Proceedings of the Second International Conference on Digital Information Management. Los Alamitos, CA: IEEE, 224-229.

XML Document Implementation: Experiences from Three Cases

Reija Nurmeksela Eliisa Jauhiainen Airi Salminen* Anne Honkaranta**
University of Jyväskylä, Department of Computer Science & Information Systems, Finland
** University of Toronto, Faculty of Information Studies, Canada*
***SYSOPENDIGIA Plc, Finland*
rekorhon\raelurja\airi.salminen@jyu.fi, anne.honkaranta@sysopendigia.com

Abstract

Implementing production of XML documents is a rarely discussed topic in academic literature even though it is an important issue in many contemporary organizations. This paper describes findings from three case organizations where different kinds of XML documents were implemented. Our findings suggest that the implementation is a domain-specific task related to various kinds of organizational activities from document authoring to business processes. As expected, the amount and complexity of document types as well as the number of people and organizations involved affect the challenges in the implementation process. Hiding the XML format from the software users and training the end users are important means to reduce the user resistance against structured document authoring and novel tools.

1. Introduction

A great deal of the information resources in organizations consist of documents produced in organizational business processes. Documents serve a number of different purposes, for example, as tools for supporting communication and decision making and as recordings of business activities. Some documents are information carriers meant primarily for human readers, while some others are targeted for software systems. Recently *Extensible Markup Language (XML)* [1] has been adopted in many organizations to support more systematic *Enterprise Content Management (ECM)*, i.e., the management of information content in various kinds of assets like documents, Web sites, intra- and extranets across the organization and between parties involved in business processes [2]. XML is a standard *de* facto by W3C consortium. It is a metalanguage that provides a way to exchange information between software applications in standardized formats.

Adopting the standardized format for documents by means of XML is motivated, for example, by the needs for interoperability, data integration, improved information access, and reuse of information content [2, 3, 4, 5]. The XML *standardization* in an organization refers to the adoption of XML standard which includes agreeing upon rules for the ways information is clustered and represented in documents as well as those for content production and management practices. The implementation of standards hence requires both technical and also organizational solutions, possibly including extensive re-engineering of information systems and document production practices (e.g. [3, 5]). ECM environments of organizations are varying and thus the ways XML is used in the environments, too. Therefore also the efforts needed for implementing the XML documents vary.

This paper describes and compares three XML document implementation cases and focuses on the changes in document production practices. The research is conducted by using qualitative case study method [6]. The case analysis is targeted on finding answers to the following questions: What motivates organizations in document standardization? How the implementation process is realized in different kinds of organizations? What kinds of changes the implementation causes in the document production practices?

The paper is organized as follows. Section 2 describes the concepts related to the adoption of XML in document production. Section 3 introduces the three cases, all of which were realized in Finland. The cases include one private sector and two public sector organizations. In the first case the implementation started by using the SGML (Standard Generalized Markup Language [7]), the predecessor to XML language, and the latter two cases were realized using XML. Findings from the cases are discussed in Section 4. Section 5 concludes the paper.

2. Production of XML documents

This section introduces the core concepts related XML documents, the various ways XML documents are produced in organizations, and a model for XML standardization process. The model will be used as an analysis tool to describe the three cases in Section 3.

2.1. Characteristics of XML documents and their use

SGML and XML documents are *structured documents* where the structure definitions, document instances, and layout specifications can be managed as separate content items. The logical structure of an XML document is hierarchical, the structure and other constraints for document content are described by an *XML schema*. In the document instance the logical structure is indicated by markup which follows the schema rules. The layout specification is typically defined with a *stylesheet*.

XML documents are often divided into data-centric and document-centric (see e.g. [8]) ones, based on their purpose and type of use. *Document-centric* XML documents are designed primarily for human understanding, and the content of these documents usually consists of natural language. *Data-centric* XML documents are primarily designed for software processing and data exchange. They are typically shorter than document-centric ones and without layout specifications. XML documents in both categories may serve as recordings of business activities. An alternative categorization for XML use is proposed by [2], dividing XML into two categories: use as information assets and use as data interchange format. In the first case the XML use results a persistent XML repository. The information assets are further divided into documents and metadata.

2.2. XML document production

XML documents may be produced in a number of ways [9]; by human actors or by software systems. Human authoring may be supported by multiple ways, such as described in the following.

Using document templates with styles that may be mapped to XML document schema by a software application which post-processes the document after the human authoring. For example, a Word .dot document template may have styles “title” and “bodytext”. The author marks the text up by using these styles in Word. A markup application takes the document as input and maps the style definitions into schema elements to produce an output XML document.

Using a generic, schema or syntax directed XML editor such as Altova XMLSpy. This editor allows the document to be produced as valid or well-formed XML. The author types the text into table view in which each cell is a placeholder for an element content, or author types the text in between the element start and end tags directly. The editor may show the schema structure and provide hints of the elements that may be added into the document with regard to the current position in XML document type schema structure.

Using a generic word processor with XML support. For example Microsoft Word 2007 and OpenOffice 2.0 support XML. Word 2007 allows a form of schema-directed editing by using content controls and element markers into which element content may be typed into. Open Office 2.0 Writer uses Open Document Format, which allows content to be typed in a WYSIWYG-interface with styles to produce an XML document which conforms to Open Document Schema, and may be transformed to another XML document markup language such as Docbook or XHTML.

Using a custom-designed interface developed for a certain document type separately. One may for example define one .edd-template for each document type schema for FrameMaker+XML, develope a tailored form according to one’s schema in Microsoft InfoPath (2003 or 2007), utilize InfoPath forms and Microsoft Forms Server and transform the InfoPath forms into Web forms, or utilise Altova’s StyleVision to define a Web form for each document type schema separately for document authoring. Some of the aforementioned solutions may be mapped with specific content controls and sources to facilitate content retrieval and reuse from external sources, such as databases or Web Service-interfaced content sources on the Web. Such functionality is provided by Altova XML Spy, Microsoft Word 2007, Microsoft Infopath and Web Forms transformed from them, for example.

In all of the cases above the document schema sets constraints to the authoring, and the author must be more or less familiar with the schema for the document type. In traditional document authoring the document content is strictly tied to the external presentation visible to the author. The authors may find the schema-guided authoring too restrictive and feel uneasy about the separation between the logical structure and layout [10]. Therefore new solutions for XML editing have been developed (e.g. [11, 12]).

2.3. Standardization process

XML document implementation in an organization results from an XML standardization process. For our case analyses we use the standardization model depicted in Figure 1. The model has been adapted from [13]. The circles represent process phases and the arrows show the order for starting the activities. The small black circle indicates that all of the following three activities may be started either in parallel or in any order.

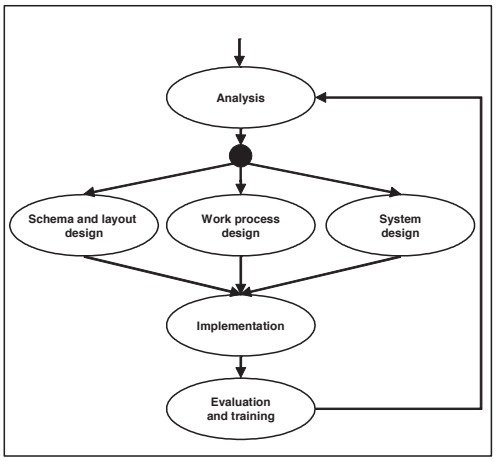

Figure 1. The XML standardization process.

The process starts with an *analysis* phase producing descriptions of the actors, business processes, systems, and documents of the domain as well as a requirements analysis report. The users of the future solutions may be involved in the process from the analysis phase.

The design of the new solutions usually requires a *schema and layout design*, carried out in parallel with *systems design* and possibly with *work process design*. The schema design is accompanied by layout design for facilitating the representation of documents to human authors and readers (e.g. [3, 5]). The system design may include, for example, selection of XML software products, their customization, and designing transformations between different data formats. Collaboration with the future document authors during the design facilitates reactivity to the anticipated problems in the schema design and systems customization [3, 5, 10].

The *implementation* of the new XML-based solution requires both technical and organizational implementation of the new solutions, possibly including major changes in document processing, as pointed out in previously reported cases (e.g. [3, 5, 9, 10]).

Evaluation and training is an important phase for successful adoption of new solutions. Evaluation may lead to further redesign. For example, XML schema design is not completed until schemas have been used by end users [5, 10]. After some time of operational use, standardization may continue by iterating the process.

The type of standardization domain most obviously affects the extent and challenges of the standardization process and the implementation of document production. Following section describes the three cases and compares them.

3. XML document implementation in the cases

In each of the cases the activities in the organizations have been supported by one or two of the authors of the paper. Most of the data were originally collected by participating in the standardization activities, interviewing people involved, and analyzing documents and schemas.

Case 1 concerned standardization of 35 **parliamentary document types** in the Finnish Parliament and 13 ministries. One of the authors was involved in the analysis and evaluation phases of the case. The description of the case is based on data analyzed and reported earlier in [3], which was updated by interviews and schema analysis for the paper. *Case 2* concerned **agendas and memorandums** of the Faculty of Information Technology in the University of Jyväskylä, in which two of the authors were involved throughout the standardization process. *Case 3* concerned standardization of **invoice documents** in an international ICT service provider and its customers. One of the authors participated in the design and implementation phases of this case. Table 1 summarizes the cases and their characteristics.

The Finnish Parliament and Government (Case 1) decrees governmental and administrative matters with the President of the Republic. Standardization activities took place during 1994-2007. **The standardization was motivated by** incompatibilities of systems, inconsistencies in representations, heterogeneity in retrieval techniques, and uncertainty of the future usability of archived digital documents. In 1994 XML was not yet published. At the end of the analysis phase SGML was chosen as the basis for standardization and preliminary schemas were designed. Redesign and implementation projects took place 1998-2001. Iteration of the standardization for replacing SGML and style-based authoring with XML was started in 2004 and is still going on.

Table 1. XML standardization in the three cases.

	Case 1: Finnish Parliament and Government, 1994-2007	Case 2: Faculty of Information Technology in the University of Jyväskylä, 2004-2006	Case 3: An international ICT provider company and its customers, 2000-2007
Analysis	The initial analysis phase during 1994-1998, including extensive data gathering. The analysis concerned the Parliamentary documents, people involved and the tools used in the document production. New analysis in the Government during 2004-2006 for adopting XML.	Fall 2004, including interviews and studying existing documents. The analysis was carried out by a student group. The analysis concerned the people involved in document production, the two document types and the tools used in document production. The analysis was iterated in department of the faculty 2006.	2000-2002 by the ICT provider. The core group consisted of project managers with support of technical consultants. The analysis concerned different invoice types as well as people, organizations and systems involved in invoicing activities. Interchange message standard for invoice was modeled. Iteration of the analysis four times during 2003-2007.
Design Schema and layout	20 preliminary SGML schemas were designed by researchers and selected companies designed the final SGML/XML schemas and layouts.	Schemas for agendas and memorandums were designed. Strict requirements for layout.	The selected subcontractor and later on the system analysts of the provider developed schema and layout for the invoice. Strict requirements for layout.
Work process	Work processes were redesigned.	Existing work processes were supported by the adaptation of the system.	Work processes were not redesigned.
Systems	Adobe Framemaker +SGML was selected as the authoring tool at the Parliament, Microsoft Word in the Government.	Microsoft InfoPath replaced Word as document authoring software.	Invoicing system produced XML documents. Significant changes in invoicing, purchase ledger and workflow systems in the first standardization process.
Implementation	During 1998-2000 in the Parliament and 2000-2001 in the Government. Transfer to SGML production in the Parliament, use of a word processor with style editor in the Government.	XML-based document production was implemented in 2005. The department version of the system was implemented in 2006.	Provider implemented incrementally exchange service for invoices during 2000-2007. Parallel related systems were implemented. Implementations in the customer organizations 2001-2007.
Evaluation and Training	Gradual improvements on the systems. Training offered to people whose work changed.	Office personnel were trained briefly before they started using the novel XML application.	Training offered to users and developers of the invoicing, purchase ledger and workflow systems.

Major challenges on the implementation have been in co-operation between different organizations, large amount of document types and instances, strict usability requirements, and user resistance. **As a result**, quality of documents has been improved. Standardization had affected both in and out of the organizations involved.

The initial project for the XML implementation in the University of Jyväskylä was carried out during 2004-2005. **The standardization efforts were motivated by** the need to improve content reuse and enhance the laborious document preparation and publishing process. **Major challenges on the implementation were** caused by the limited timetable and the lack of XML competence on the project group. **As a result**, quality of documents i.e. the consistency on the content and layout was improved and document publishing as well as content reuse were enhanced. The project was followed by another standardization project at a department of the faculty in 2006. This time the implementation was smooth and easy, since the solution developed in the faculty was tested and successfully implemented before the decision for its adaptation in the department was done.

Case 3 considered XML implementation for invoice documents in an international ICT service provider company and its customers. **The standardization efforts were motivated by** the need to improve data integrity between invoicing and purchase ledger systems in small and medium enterprises (SME), speed up the handling of invoices, and to reduce the costs. The case was started in 2000 and a new XML-based solution was implemented for the first customers in 2001. The implementation of a new version of invoicing system for the first customer organization took a few days. During 2003-2007 the standardization process was iterated four

times, because of the need for new schema versions. **Major challenges on the implementation** have been caused by the large amount of document instances, disagreement of identification standards with different business partners, and importance of layout. **As a result**, quality of invoice data has been improved and the invoicing process streamlined.

4. Evaluation

The motivation for SGML/XML implementation and standardization varied from case to case. In Case 1, standardization was activated by inconsistencies in content management, incompatibilities of tools, and uncertainty of the future usability of archived digital documents. In Case 2, requirements for content reuse and difficulties in document publishing were the main motivations. In Case 3, automation of invoice processing in SMEs was the key motivator.

The SGML/XML implementation in the three cases has major differences. Case 1 was clearly the most challenging. It started at the time before XML, the experiences about the use of SGML in public domain were limited, the SGML tools were expensive, and there were not many choices for the tools. Compared to the other cases, the number of document types and document instances was much bigger, as well as the number of people affected by the standardization. The documents in question were nationally very important and the standardization was expected to have many impacts both in the work environment and in the society as a whole. These expected impacts most probably gave extra motivation to the persons involved. Wide impacts have also been realized as reported in [3].

In Cases 1 and 2 the XML documents were targeted for human consumption due to which the document layout design was an essential part of the standardization process. The analyses conducted in the cases focused on similar aspects; documents and tools used as well as people involved in document production were analyzed. In Case 3, the document type to be standardized (the invoice) had both document- and data-centric characteristics. Thus the analysis was focused on the data stored in the accounting systems and the organizations involved in the invoicing process, instead of people as end users of the system. In all cases the layout requirements had a significant impact on the schema design.

In Case 1 the document schemas were designed incrementally within years. In Cases 2 and 3 the document schemas were designed within a few months. In each of the cases the schema design was an iterative process. The tools utilized for content reuse in Case 2, and the data integration requirements between systems in Cases 1 and 3 had effects on the schema design. In all cases usability requirements had an impact to schemas [e.g. 10]. For example, in Case 2 significant requirements and limitations came from the authoring tool, which provided the visual layout for authors. The layout had to support the functions and easy to use. Therefore schemas were modified several times before the implementation. In Cases 1 and 2 the changes in the schemas reflected further in the authoring tools.

In each of the cases new formats were implemented to support multi-channel publishing. Changes in work practices vary between the cases. In Case 1, the standardization changed significantly work tasks of different groups of people including new publishing practices. In Case 2, work practices remained in essence the same, only the authoring tool changed. Furthermore, because the process of preparing agendas and memorandums was similar in the Faculty of Information Technology and in another faculty, the same XML-based system was soon implemented in the other faculty as well. Only minor refinements in the XML schemas were needed and the change of the document authoring software was not resisted due to the awareness of the user satisfaction gained by the novel system and due to the fact that the benefits were already manifested by the neighboring faculty. Easy and simple launching of new systems based on the same content is also reported in [5].

In Case 3, the redesign of work was also avoided. Only the new functionalities of the new versions of the invoicing, purchase ledger and workflow systems were introduced, together with the new invoice exchange service. As in Case 2, the same versions of the systems were quickly implemented in many other organizations, because the invoicing process is similar and also the same systems are used in those organizations.

The case comparison reveals that if the business process and document types involved in it are long and complex, and the amount of documents is large, as in Case 1, the standardization is a time-consuming and complex task. Some of the complexity in the case was caused by the large number of actors involved. This was also observed in Case 3 where negotiating the agreement of identification standards between several organizations was one of the main challenges. Similar findings are reported e.g. in [4].

If there is a great number of document authors involved, a major emphasis has to be given to the usability of authoring tools. The usability requirements were a significant challenge in Case 1, because the authors had to learn new ways for authoring, guided by logical structure of documents.

Minimizing user resistance required some efforts. In Case 2, usability remained important, but building XML support the authoring tools helped the implementation. The form-based user-interface hided the logical structures from the authors. In Case 3 the logical document structures were hidden from the users by generating XML documents automatically from databases.

If the production of XML documents can be embedded in existing systems and processes within organizations, the need to change work practices decreases as in Cases 2 and 3. In Case 1, the authoring tool was new and cumbersome to use. Another major challenge was the adoption of the novel publishing tasks. Thus implementation in Case 1 was more challenging than in Cases 2 and 3 where corresponding changes did not occur.

5. Conclusion

The paper described three cases where the goal was to improve enterprise content management by the adoption of SGML/XML. Consistency in content management practices, automation of business processes, and more effective content reuse were important motivators of the adoption but the emphasis of the goals clearly differed in the cases.

The findings suggest that XML document implementation is domain-specific task that requires co-operation of people and organizations. Cases 2 and 3 show that it is possible to produce XML schemas and embed XML-based production into software and thereby lower the end-user resistance.If the benefits of the XML document production have been earlier demonstrated, the adoption of a novel system may be quite fast and fluent, as demonstrated in Cases 2 and 3. The XML standardization model by [13] was successfully utilized as the framework for analyzing the three cases. The framework may therefore be utilized as an analytic tool both for XML standardization development and for case research.

Our study focused only on document-like content of the ECM environment [2] and neglected the use of XML for metadata. Metadata standardization and implementation for XML document production is a possible avenue for further research, as well as the comparison of the findings with other studies.

6. Acknowledgments

The research was facilitated by the active cooperation and support of a great number of experts in the Finnish Parliament and Government, Faculty of Information Technology in University of Jyväskylä, and the international ICT service provider compary of Case 3. The authors are grateful to all of them. Two anonymous reviewers provided invaluable comments for improving the paper.

7. References

[1] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., and Cowan, J. 2006. *Extensible Markup Language (XML) 1.1. (2nd Edition)* W3C Recommendation, W3C Consortium. http://www.w3.org/TR/xml11/ [February 8, 2007]

[2] Salminen, A. 2005. "Building digital government by XML". In R. Sprague, Jr. (Ed.). *Proceedings of the Thirty-Eighth Hawaii International Conference on System Sciences*. Los Alamitos, CA: IEEE Computer Society, 122b- 122b.

[3] Salminen, A., Lyytikäinen, V., Tiitinen, P., & Mustajärvi, O. (2004). "Implementing digital government in the Finnish Parliament". In W. Huang, K. Siau, & K.K. Wei (Eds.), *Digital Government: Strategies and Implementation* (pp. 242-259). Hersley, PA: IDEA Group Publishing.

[4] Nurmilaakso, J.-M., Kettunen, J. and Seilonen, I. 2002. "XML-based supply chain integration: a case study." *Integrated Manufacturing Systems 13 (8),* 586-595.

[5] Weitzman, L., Dean, S., Meliksetian, D., Gupta, K., Zhou, N., and Wu, J. 2002. "Transforming the content management process at IBM.com", *Case studies of the CHI2002/AIGA Experience Design Forum*, ACM Press, New York, 1-15.

[6] Yin R. 1994. *Case Study Research: Design and Method*, Sage, Beverley Hills.

[7] Goldfarb, C. F. 1991. *The SGML Handbook*. Oxford, Oxford University Press.

[8] Smith, H., and McKeen, J. 2003. "Developments in practice viii: enterprise content management". *Communications of AIS 11 (33)*, 1-26.

[9] Braa, K and Sandahl, T. 1998. "Approaches to standardization of documents". In Wakayama et al. (eds.) *Information and process integration in enterprises: rethinking documents,* Kluwer Academic Publishers, Cambridge, Massachusetts, 125-143.

[10] Sandahl, T., and Jenssen, A. 1997. "The First Steps in Designing an SGML-Based Infrastructure for Document Handling". *Scandinavian Journal of Information Systems, 1997, 9(2)*, 25–44.

[11] Müller, U., & Klatt, M. 2005. SCOPE – An XML Based Publishing Platform. In *Proceedings of the 8th International Symposium on Electronic Thesis and Dissertations*.

[12] Sefton, P. 2007. An Integrated Approach to Preparing, Publishing, Presenting, and Preserving Thesis. In *Proceedings of the 10th International Symposium on Electronic Thesis and Dissertations*.

[13] Salminen, A., Lyytikäinen, V., and Tiitinen, P. 2000. "Putting documents into their work context in document analysis". *Information Processing & Management 36 (4),* 623-641.

IV

A LIFE CYCLE MODEL OF XML DOCUMENTS

by

Airi Salminen, Reija Nurmeksela & Eliisa Jauhiainen

Accepted to the Journal of the American Society for Information Science and Technology (JASIST). October 2, 2013.

A Life Cycle Model of XML Documents

Airi Salminen*
Department of Computer Science and Information Systems
University of Jyväskylä, FI-40014
P.O. Box 35 (Agora)
Jyväskylä, Finland
phone: +358 50 518 6284
fax: +358 14 260 3011
email: airi.salminen@jyu.fi

Reija Nurmeksela
Tieto Finland Oy
Mattilanniemi 6, P.O. Box 163
FI-40101 Jyväskylä, Finland
phone: +358 20 7257618
fax: +358 20 5496955
email: reija.nurmeksela@tieto.com

Eliisa Jauhiainen
Department of Computer Science and Information Systems
University of Jyväskylä, FI-40014
P.O. Box 35 (Agora)
Jyväskylä, Finland
phone: +358 40 8053094
fax: +358 14 260 3011
email: eliisa.jauhiainen@jyu.fi

* corresponding author

A Life Cycle Model of XML Documents

Electronic documents produced in business processes are valuable information resources for organizations. In many cases they have to be accessible long after the life of the business processes or information systems where they have been created. To improve the management and preservation of documents, organizations are deploying the Extensible Markup Language (XML) as a standardized format for the documents. The goal of the paper is to increase understanding of XML document management and provide a framework to enable the analysis and description of the management of XML documents through their life. We have followed the design science approach. We introduce a document life cycle model consisting of five phases. For each of the phases we describe the typical activities related to the management of XML documents. Furthermore, we also identify the typical actors, systems, and types of content items concerned in the activities of the phases. We demonstrate the use of the model in two case studies: one concerning the State Budget Proposal of the Finnish Government and the other concerning the Faculty Council Meeting Agenda at a university.

Introduction

A great deal of the information resources in organizations consists of documents produced in business processes. Documents serve a number of different purposes, for example, as tools for communication and decision-making. They also serve as recordings of business activities and have to live, in many environments, through generations of

technologies, systems, users, and surrounding organizations (e.g., Volonino, Sipior, & Ward, 2007; Borglund, 2008). Today Extensible Markup Language (XML) has become the *lingua franca* of the data interchange on the Internet and its use is becoming increasingly widespread also for representing documents produced in business processes. In some organizations XML is adopted simply as a document format of an office application. Standard XML-based, open formats for office documents are ODF (OpenDocument Format for Office Applications, ISO/IEC 26300:2006) and OOXML (Office Open XML File Formats, ISO/IEC 29500-1:2008). Today many public domain organizations have published policies to support or enforce the use of open document formats. Examples of them are the government of Norway (Ministry of Government Administration, Reform and Church Affairs, 2007) and the state government of Massachusetts in the United States (Shah, Kesa, & Kennis, 2008). An important motivation for the adoption of open file formats is to achieve vendor-independency as stated by Cerri and Fuggetta (2007):

> The technology supplier cannot claim any right on the customers' data and information or impose limitations and constraints on their manipulation. *The customer must have the true possibility to switch to another supplier and to access its own information without being anyhow limited*. (p. 1936)

Especially in public sector open standards are also an important means to release data as machine readable open data to support transparency and data sharing (see e.g. Peled, 2011).

When using an XML-based format of an office application document instances are encoded as marked-up text where the markup shows the structures identified by the office

application (like titles, paragraphs, and lists). This is not always sufficient: there may be a need to define a domain-specific markup language to incorporate semantic information in document markup. The definition capabilities related to XML are available for the purpose. Semantic markup provides several possibilies, including improved accessibility, reusability, and information integration (e.g., Bernstein & Haas, 2008; Salminen & Tompa, 2011). An open file format together with semantic markup has also been seen as a way to improve the persistence of information through time (Brooks, 2001).

Since the publication of XML as a W3C (World Wide Web Consortium) standard in 1998, the development and research related to XML has been extremely active. In Google Scholar the search term XML provides over 2,000,000 results. A great deal of the practical development concerns software development and development of XML-based standards for particular application domains. The emphasis in the academic research has been in the technologies used for processing and retrieving XML data. Research considering the management of XML documents over their life, including the life after the active processing and use of documents, is rare. This concerns document management research also more generally. Respectively, in the information management of organizations there often seems to be a gap between the active and passive life of documents (Barker, Cobb, & Karcher, 2009). Our goal in the paper is to provide a model to enable the analysis and description of XML document management over the whole life of documents. Another goal of the paper is to increase understanding of XML document management in organizations.

Our research approach is design science. Design science has been widely used in many research disciplines, especially in engineering and computer science but also in

information science and information systems. Design science attempts to create artifacts that serve human purposes (March & Smith, 1995). March and Smith (1995) have divided the artifacts into four kinds: constructs, models, methods, and instantiations. In our case the developed artifact is the life cycle model of XML documents. We demonstrate the use of the model in two different cases: one concerning the State Budget Proposal of the Finnish Government and the other concerning the Faculty Council Meeting Agenda in a university faculty. The case descriptions can also be seen as expository instantiations used to test and help explaining the model (Gregor & Jones, 2007).

Peffers, Tuunanen, Rothenberger, and Chatterjee (2008) have derived from a number of prior design science methodologies (e.g. Nunamaker, Chen, & Purdin, 1990; Hevner, March, & Park, 2004; Gregor & Jones, 2007) a nominal sequence of six steps in design science research (DSR) process: (1) problem identification and motivation, (2) defining the objectives for a solution, (3) design and development of the artifact, (4) demonstration, (5) evaluation, and finally (6) communication. The steps also provide a template for the structure of research outputs. Figure 1 depicts how we have used the process to structure the paper. The figure also shows techniques that we have adopted for the research. Outputs of the communication phase are provided in the paper.

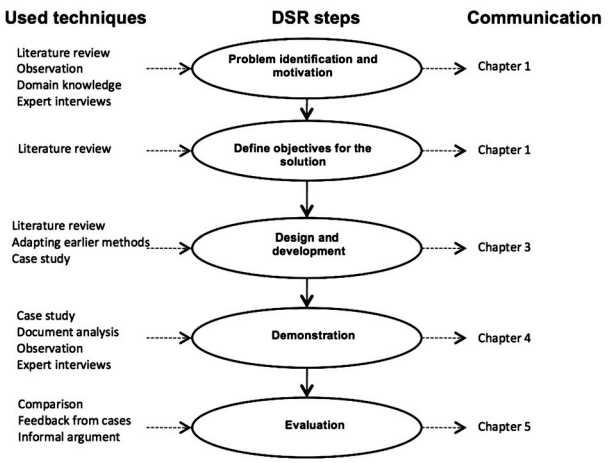

FIG. 1. The design science research steps of the study.

In our case, the design of the new artifact continues the earlier RASKE methodology development. RASKE methods and use cases have been communicated to researchers in scientific articles (e.g. Järvenpää, Virtanen, & Salminen, 2006; Salminen, Kauppinen, & Lehtovaara, 1997; Salminen, Lyytikäinen, & Tiitinen, 2000; Salminen, 2005; Salminen, Nurmeksela, Lehtinen, Lyytikäinen, Mustajärvi, 2008; Salminen, 2010; Tiitinen, Lyytikäinen, Päivärinta, & Salminen, 2000) and to practitioners in project reports, seminars and workshops. Before introducing the life cycle model we discuss the concepts and methodologies related to XML document management.

XML Document Management

This section provides the background intended to support understanding of the life cycle model and related case examples later in the paper. We first introduce the main concepts of XML document management. Then we briefly describe and compare

methods that have been proposed for the analysis and description of XML document management. Finally we introduce the core components of an XML document management environment.

Concepts

The term *document management* is used to refer to the creation, storage, organization, transmission, retrieval, manipulation, update, and eventual disposition of documents in organizational context, to fulfill organizational purposes (Sprague, 1995). *XML document management* refers to the management of documents in XML format. XML documents are *structured documents* where a document instance is described as a hierarchic structure of named parts (Bray, Paoli, & Sperberg-McQueen, 1998). The parts are explicitly indicated by systematic markup that enables applications to identify, retrieve, and process those parts in a similar manner as data in databases.

Document management can be regarded as a special kind of *enterprise content management* (ECM). Smith and McKeen (2003, p. 648) define the term as "the strategies, tools, processes and skills an organization needs to manage all its information assets (regardless of type) over their life cycle". A term closely related to document management is records management. In the records management standard 15489 (ISO 15489-1, 2001, p. 3), a *record* is defined as "information created, received, and maintained as evidence and information by an organization or person, in pursuance of legal obligations or in the transaction of business" and *records management* as "field of management responsible for the efficient and systematic control of the creation, receipt, maintenance, use and disposition of records, including processes for capturing and maintaining evidence of and information about business activities and transactions in the form of

records". In contrast to document and content management systems that are primarily intended to support on-going business processes involving editing or versioning of content, records management systems are primarily intended to provide secure repository of authentic records (DLM Forum Foundation, 2011). A record captured into a records management system may consist of one or more documents (ISO 15489-1, 2001) or of other kinds of components. In records management systems modifications of records are prevented or strictly controlled whereas in content or document management systems updates and versioning are typical operations.

Similarly to databases, the structure and other constraints for a class of XML documents on a domain is described by a *schema*. The schema defines a markup language for the domain. A great number of schemas have been defined for specific sectors or application domains. For example, the earlier mentioned ODF and OOXML for office applications, HL7 standards for the health sector (www.hl7.org), and the Universal Business Language (UBL, www.oasis-open.org) for business documents. Organizations sometimes adopt a pre-defined schema like ODF or OOXML as such for documents produced in their processes. In other times they define their own schemas either from a generic schema earlier developed by a standardization organization, for example, from HL7 or UBL, or from the very beginning.

The layout of XML documents on an output medium is usually defined by means of *style sheets*. This enables the separation of structure, layout, and content of a document from each other, and processing each of them separately (see Figure 2). There are different *schema languages* available to define the structure and different *style sheet languages* to define the layout. The figure lists three of the available schema languages

and two of the available style sheet languages. From the schema languages DTD (Document Type Definition) is defined in the XML specification. The style sheet languages CSS and XSL, as well as the schema language XML Schema, have been developed and published by W3C. RELAX NG is the schema language developed by Clark and Murata through OASIS (Advancing open standards for the information society; Clark & Murata, 2001).

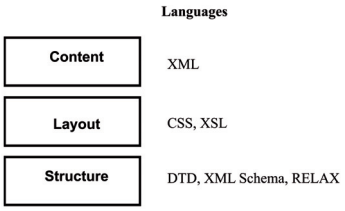

FIG. 2. Three facets of an XML document.

From the point of view of XML document management it is important to realize that XML documents actually have two structures at the same time: logical structure and physical structure. The core components of the *logical structure* are *elements* indicated in the marked-up document by tags of the form <...> and </...>. The *physical structure* consists of storage units called *entities*. The entities of a document are either separate files or named pieces of text reused in different places. The entities are connected to each other by entity references. The management of XML documents requires the management of both structures, logical and physical.

Systematic management of any kind of content items requires *metadata*. Metadata may be related to collections of items or to individual items (Gilliland, 2008). In the first case we use in this paper the term *class metadata*, in the latter *instance metadata*. Instance metadata is often regarded as a surrogate of the instance like, for example, a library record for a publication is a surrogate of the publication (Greenberg, 2010).

Examples of class metadata in an XML document management environment are schemas specifying the structural and other constraints for a class of documents, or *ontologies* defining concepts and their relationships to be used, for example, for annotating documents. In fact, also schemas define ontologies for the structural components of documents. Instance metadata attached to an XML document might be described, for example, by using Dublin Core (DC) standard (Dublin Core Metadata Initiative, 2010) and embedded in documents or stored externally, possibly in XML format (Powell & Johnston, 2003).

Methodologies

Methods and techniques from various information, information systems, and business process analysis methodologies can be used to analyze and describe XML document management. Since XML was derived by a number of restrictions from the older meta markup language SGML (Standard Generalized Markup Language; Goldfarb, 1990), the methods originally developed for SGML are applicable also in XML environments. In the following we briefly describe and compare four widely published methodologies that have been introduced and used to support the deployment of SGML or XML documents: the DTD development methodology of Maler and El Andaloussi (1996), the RASKE methodology (Salminen et al., 1997; Salminen et al., 2000; Tiitinen et al., 2000; Salminen, 2010), the Unified Content Strategy (Rockley, Kostur, & Manning, 2003), and the Document Engineering approach (Glushko & McGrath, 2005). Each of the four methodologies has its own emphasis, adopting and adapting older methods. The comparison of the methodologies is summarized in Table 1. In addition to the four aforementioned methodologies, we have included in our comparison Dirks,

guidelines for the management of digital records (State Records Authority of New South Wales, 2007). In the following we briefly describe some characteristic features and core concepts of each of the five methodologies.

TABLE 1. Summary of the five methodologies.

	Primary application domain	Types of documents	Document format	Modeling concerns
Maler&ElAndaloussi	publishing industry & technical documentation	narrative	SGML	document and component structures
RASKE	electronic document management in organizations	narrative	SGML/XML recommended	organizational environment, business and work processes, roles, documents, metadata
Unified Content Strategy	content management and multichannel publishing	narrative	XML recommended	content life cycles, element structures, reuse, documents, workflows, content management processes
Document Engineering	automated business processes	narrative/transactional	XML recommended	business processes, documents, components
Dirks	records management	narrative/transactional	various formats, XML recommended for archival format	modeling is not explicitly included in the methodology

The methodology of Maler and El Andaloussi (1996) has emphasis on the design of SGML DTDs providing means for the DTD project management, document type needs analysis, document type design, DTD development, validation, and documentation. The tree structures of documents are modeled by *elm diagrams*. They have been widely adopted and adapted in SGML/XML syntax-oriented editors and also in other methodologies for describing SGML/XML structures.

The emphasis in the RASKE methodology is in the holistic analysis of a document/content management environment. Documents, metadata, and other content items

are considered in the organizational context of business processes. The RASKE methodology includes methods for requirements analysis, modeling, and evaluation. It has adopted and adapted some object-oriented modeling methods for process modeling, document modeling, and role modeling. From the process models the most extensively used in the methodology have been the *input* and *output models* showing the resources used or produced in business processes. *Document modeling* is divided into object, state, and content modeling.

The fundamental concept in the Unified Content Strategy is *reuse*. The main purpose is to avoid content "silos" by effective content reuse. The implementation of the strategy is divided into eight phases starting with analysis, design, and selecting tools and technologies (e.g. XML). The analysis phase includes the analysis of content life cycle processes but no particular process modeling technique is suggested. The most common life cycle is considered to extend from creation to delivery; problems related to content archival and retention are outside the scope of the methodology.

While the previous methodologies have been developed to support the management of narrative documents that are primarily intended for human use and human communication, the Document Engineering Approach provides concepts and methods to design effective business transactions and Web services by means of documents, and in particular, by means of XML documents. The central concept of *model matrix* is used as a roadmap to advance from models at different levels of abstraction and granularity to effective implementations of transactions. No modeling technique is inherently required in the approach but the use of XML-encoded implementation models is emphasized. Similarly to RASKE, documents are regarded as inputs and outputs of business

processes. RASKE, however, considers process models as descriptive tools to support human communication and understanding, not as prescriptive tools to enable automated business processes. The problems related to document publishing, archival, and retention are left outside the scope of Document Engineering.

Dirks methodology was especially designed for building good recordkeeping systems into Australian Government agencies. In the methodology a recordkeeping system may be a separate information system but often it is (or should be) embedded in a business information system, in order to keep records of business transactions. The methodology includes a great number of guidelines, for example, for capturing, storing, and securing digital records, creating metadata about them, determining how long to keep them, enabling business continuity also in disaster situations, preserving digital records for long term, and finally disposing them. Dirks also defines a process of eight steps for designing and implementing recordkeeping systems. In contrast to the previous four methodologies, Dirks does not provide methods and techniques for analyzing and designing documents. The use of open standards for records is introduced as a possible design strategy and XML in particular as a possible archival format to which records are converted and then packaged together with related metadata.

XML Document Management Environment

In this section we describe the components of an XML document management environment using the concepts and models that we have earlier introduced in the RASKE methodology. The main components of a document management environment can be divided into activities and resources (see Figure 3). In the figure the activities are depicted by the oval, resources by rectangles, and information flow between resources and

activities by arrows. An *activity* is a set of actions performed by one or more actors in a work process. An activity can be divided into smaller activities and described by means of a *process model* and instantiated in a *process case*. The resources are of three types: actors, systems, and content items. An *actor* is an organization, a person, or a software agent. A *system* is a tool that is used to support the performance of activities. Various kinds of systems are needed in computerized environments, including hardware, software, networks, standards, and mandates. Mandates can be, for example, regulations and legislation governing the document management of the domain. *Content items* are documents and other units of stored data accessible as meaningful pieces of information by means of systems. All resources are information repositories in which the information produced in an activity can be stored or from which information can be taken and used in an activity. Thus information is not stored only in documents and other content items but also in the heads and experience of people, in the organizational culture, and in systems.

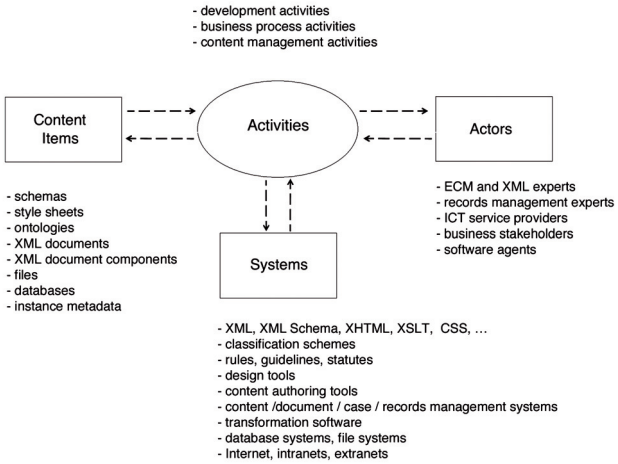

FIG. 3. An XML document management environment.

By analyzing documented cases (e.g. Aversano, Canfora, de Lucia, & Gallucci, 2002; Broberg, 2004; Sartor, Palmirani, Francesconi, & Biasiotti, 2011; Sohn, Ko, Lee, Kim, Lim, & Choy, 2002) as well as using the information gathered by observations and interviews from a number of XML deployment cases (e.g. Nurmeksela, Jauhiainen, Salminen, & Honkaranta, 2007; Salminen et al., 2004), we have identified the typical components in an XML document management environment. The *activities* may be divided into three categories: development activities, business process activities, and content management activities. *Actors* include organizations and experts needed in the three kinds of activities.

The most typical *content items* accessible from the data repository of an XML document management environment are XML documents, their schemas and style sheets. The components of the physical structure of documents are stored as files or in a database. The types of metadata resources vary: in some environments all instance metadata associated with XML documents is embedded in documents and in their file names, in others the instance metadata is stored externally, for example, as Dublin Core metadata in self-contained XML files. In simplest cases the structural concepts expressed in schemas are the only ontologies managed systematically as content items. Some environments maintain term dictionaries, some others more complex ontologies. Also these ontologies can be represented in XML format, for example, using the XML syntax of RDF (Resource Description Framework). Information from the content items that are stored in open formats should be accessible by several systems, not only by those available in the environment at the time the items have been created.

The *systems* needed in an XML document management environment include a great number of standards besides XML, for example, XML Schema for schemas, XSLT for transformations, HTML and XHTML for Web pages, and CSS for style sheets. In many cases also sectoral standards like Open Document Format (ISO/IEC 26300, 2006) or Office Open XML (ISO/IEC 29500, 2008) for office documents, or ebXML (www.ebxml.org) standards for business documents are needed. In all organizations there also are rules and regulations concerning the document management. Some of the rules are produced internally, some of them are external rules expressed, for example, in the legislation of the country or as best practices of the industry. Classification schemes are needed for organizing content units. Also specification of access rights may require classification schemes. Especially in the case of inter-organizational business processes, the number of different software systems involved in a process case may be great, including different authoring tools, transformation software, content or document management systems, database systems, workflow systems, case management systems, and records management systems.

XML Document Life Cycle

Since documents produced in a business process have to be accessible longer than the life of the process, it is important to consider the special features of XML document management from the point of view their own life, not only their role in the business process. In this section we introduce a document life cycle model of five phases. We describe the typical activities related to the management of XML documents in each phase. Furthermore, we identify the typical actors, systems, and types of content items

concerned in the activities of the phase. The model utilizes the concepts that we have earlier introduced in the RASKE methods.

In Figure 4 we have used the RASKE process modeling technique to depict the document life cycle activities. The major content units produced in the activities are shown on the right. Conceptually the units are stored in a repository accessible by different systems. In practice, however, a number of different repositories are often in use and different software systems have access to different repositories during the process. The activities in the figure are connected to each other with *weak control flow* meaning that a control flow arrow (solid line) from an activity A to activity B indicates that activity B starts after activity A has started. The modeling technique allows iteration and parallel activities without showing them explicitly. For example, the design activities may continue parallel with all other activities.

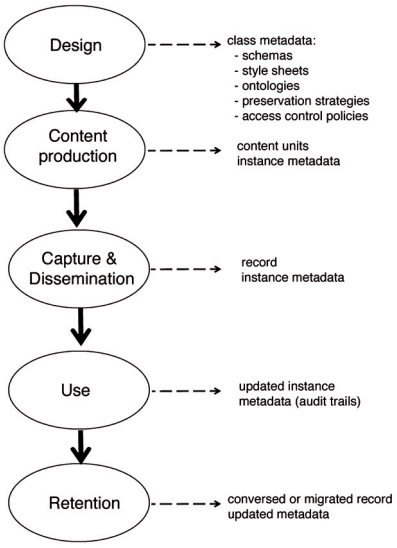

FIG. 4. Activities of an XML document life cycle.

The *design* activities produce the class metadata for the document repository, including schemas for documents and metadata, style sheets, and ontologies. Also preservation strategies and access control policies may be formally defined and stored as content units. *Content production* activities produce the content units from which documents are constructed. The content units are stored with related metadata. The *capture and dissemination* activities refer to the capture of documents as records, making them accessible for use by some system, and publishing the documents for use. The *use* activities refer to the access of documents and information from documents. The *retention* activities are intended to maintain the accessibility of documents and information in them, as well as their integrity and authenticity as records. Retention also includes the possible disposal of documents. Below each of the activities is described in more detail, emphasizing the XML-related features in all of the phases.

Design

The design includes the planning, implementation, and maintenance of the document management solutions for the environment. The schema design lies in the core of the design activities. Besides schemas, the design concerns the external layout of documents, tools to be used (e.g. for document authoring, distribution, and exchange), business and content management processes, responsibilities, the ways people and organizations use documents to collaborate and communicate with each other, as well as the ways various software systems communicate with each other. The design also concerns ontologies and metadata to be associated with documents. All methods briefly introduced earlier are applicable for the design activities. Additional methods or methodologies may be adopted to support special design areas like enterprise architecture (Liu, Wang, & Quan, 2009),

access control (Bertino & Ferrari, 2002; Bhatti, Ghafoor, Bertino, & Joshi, 2005; Bertino, Ferrari, Paci, & Provenza, 2007; Kundu & Bertino, 2008), security (Bernard, 2007), records management (DLM Forum Foundation, 2011), ontology (Noy & Hafner, 1997), or digital preservation (Stanescu, 2005; Runadotter, Mörtberg, & Mirijamdotter, 2011).

A great number of various systems may be involved in the design, both as enablers and constrainers. Operational software systems in use in the business processes often set critical constraints to the design. The new solutions may be implemented as changes in the old software or new software may be built or obtained. The design concerns all phases of the document life cycle and therefore diverse expertise is needed in the design.

Design of XML document management solutions is a continuous process typically including a great number of different tasks after the first implementations, for example, the design of XML transformations for multiple-channel publishing and planning schema, style sheet, or ontology updates. XML schemas evolve over time and this evolution should be taken into account already in the original schema design (Genevès, Layaïda, & Quint, 2011).

Content production

In content production the main purpose is to create, maintain, and store the content units that are used as components in XML documents. In some environments the content is entered directly to an XML document. At the content production stage content items are regarded as editable data, be they XML documents, files in various formats, or data in a database. Important information repositories for XML documents are databases that are created and maintained in various operational systems. In some cases large amounts of data are collected by migration from legacy content repositories, for example, from

documents in pdf, SGML, HTML, or proprietary formats (e.g. Reuben, 2003). The transformation into XML format is a semi-automated process where a human expert of the content solves the problems that cannot be handled automatically.

The creation and editing of a content unit may be performed using any kind of authoring system like, for example, an office system or a syntax-directed editor. In the last case, the validity of the content against the schema is checked at the time of content authoring. A workflow system designed especially for collaborative authoring of XML documents may enable, for example, adding multiple digital signatures to the document as fragment signatures (Brooke, Paige, & Power, 2010). In some cases the content units are stored simply in a file system. A great number of more advanced systems are however available. They can be roughly divided into three categories: content management systems, XML-enabled database systems, and native XML database systems. Characteristic features of these three kinds of systems have been described in (Salminen & Tompa, 2011). Research on version management of XML data has produced a number of articles during the last years (e.g. Mella, Ferrari, Bertino, & Koglin, 2006; Rönnau & Borghoff, 2009; several articles published in the ACM Document Engineering symposiums during 2005-2011).

Capture and Dissemination

At capture documents become under systematic records management. They are associated with metadata that provides evidence of the creation of the records and enables the access of the documents, information in them, understanding their content in the context of their creation, and their retention. If the content of a record is not stored in the form of an XML document at the time of content authoring, then the XML document is

first assembled from units available in the content repository, validated, and then captured. The ISO standard 15489 for records management defines the necessary functions included in the capture of a record (ISO 15489-1, 2001; ISO 15489-2; 2001). The capture may be implemented in the same content management system where the content is produced. Alternatively the documents are registered and stored in a separate system designed especially for records management.

There are various ways to capture documents as records in an XML document management environment. One possibility is that the records management operations as defined in records management standards concern only the publishing format of the XML document, for example, pdf. In that case, however, the benefits of XML for dissemination, use, and retention are missed. One of the benefits is that XML format enables selective dissemination of document content, according to the stated access control policies (Bertino & Ferrari, 2002). On the other hand, XML as an open standard format is also suitable for the publication of documents as *open data*, available for software processing in commercial services.

MoReq2010 (DLM Forum Foundation, 2011) specification is a *de facto* industry standard for software systems that manage records. In the MoReq terminology, records are organized in *classes*. A record consists of one or more *components* and is associated with metadata, event history, and access control list. In case of XML documents, documents of the same type might be organized as records in a class. The metadata associated with a document of the class would include schema and style sheet information. The components might consist of the external entities of the document. If

different style sheets are in use for different documents of the class, then the style sheets might be stored as components.

MoReq2010 defines for every record, independently of its format, an XML export format. The export in the specified format is mandatory to all MoReq2010-compliant systems. Content in non-XML data formats can be either embedded in the XML export data format, or linked by an URI to the XML data.

Use

The use of XML documents includes location, retrieval, presentation, interpretation, and reuse of the data stored in documents and in the associated metadata. The data access is performed either by human users or by software systems, inside or outside the context where the content items and documents were created. In commercial enterprises the use of documents is usually strictly controlled by access rights and tracking. Some of the documents created in an organization can however be publicly accessible, as part of the continuingly growing ecosystem of heterogeneous Web data. The use of documents often involves the reuse of their parts in other documents.

A great number of methods to improve the retrieval effectiveness from XML documents have been developed, as shown, for example, by the surveys of Liu, McMahon, and Culley, (2008) and Luk, Leong, Dillon, Chan, Croft, and Allan (2002). The retrieval may concern whole documents or their parts, or data is gathered from several documents, or by integrating data from different data repositories (Pérez, Berlanga, Aramburu, & Pedersen, 2008). In many situations the diversity of repositories potentially containing useful data is great, including data both in XML and non-XML format, institutional repositories and open-access repositories, on local, regional, national, or

international level. It is important that the user has capabilities to check the trustworthiness of the data. The Online Computer Library Center and The Center for Research Libraries have jointly developed and published criteria and checklist for the audit and certification of trustworthy repositories (OCLC and CRL, 2007). No special criteria are expressed for XML repositories but the checklist still provides a tool for evaluation also for those repositories.

Retention

Retention includes activities for maintaining the usability, integrity, and authenticity of documents. Usable documents can be located, retrieved, presented, and interpreted, in spite of the possible changes in the technological and organizational environment. Metadata plays a critical role in retention. The retention policies developed for corporate document repositories must include rules concerning the length of retention. Unless the retention has been defined permanent, the retention activities include disposal according to the disposal schedule.

In XML document management environments there often is a need to change XML schemas. The documents conforming to the old schemas and stored in the document archive do not necessarily conform to the new schemas without changes. In these situations maintaining the accessibility of information possibly requires the transformation of old documents so that they conform to the new schemas and preserve information. Methods for transformations needed by schema evolution have been developed earlier for databases and later for XML documents (e.g. Kwietniewski, Gryz, Hazlewood, & Van Run, 2010). Another alternative is to maintain different schema versions.

The retention activities in an environment may include the transfer of non-XML content items into XML format. The National Archives of Australia (Heslop, Davis, & Wilson, 2002) has described a preservation process where items in any format are transferred into a normalized XML format. In the normalization the core idea is to preserve the *essence* of the source document, to enable the recreation of the essential performance over time. Thus the preservation policies have to include the specification of the essential features of the source documents. Several other proposals have been made for migration strategies where the obsolescence of file formats has been solved by migration to XML (e.g. van Horik & Roorda, 2011). The source documents being originally XML documents, the preservation policies in some environment might determine schemas belonging to the essence of the documents, in some other environment they might be seen non-essential from the point of view of long-term access. Similar differences may occur concerning style sheets. In some environment the archival of different XML document versions is important for enabling historical queries concerning the evolution of documents and their contents (Wang & Zaniolo, 2008).

Demonstration in Two Cases

In this section two very different XML document management cases are described, both from Finland. We have been involved in both cases mostly as researchers, partly as consultants and teachers. The first case describes a part of a complex information and content management environment having importance at the national level. The other case concerns content management in a Faculty Office. In both cases one central document

type of the environment has been chosen for the life cycle description. Our case descriptions are divided into four parts:

Data gathering methods. This part explains the information sources used for the case description.

XML document management environment. This part identifies the core components of the environment where the life cycle activities of the case take place.

Life cycle description. The most important activities, actors, systems, and content items in each of the five phases are covered. The description of the design phase includes some history background, the descriptions of the other phases concern the current situation.

Impact analysis. In this part the idea is to assess the consequences of the deployment of XML in the environment.

The Case of the Finnish State Budget Proposal

The State Budget Proposal of the Finnish Government is prepared in a complex XML-based environment supporting the budgetary process of the Government. The schedule and deadlines for both content management and decision-making activities are tight. Annually one State Budget Proposal and typically 1-5 Supplementary State Budget Proposals are created.

Data gathering methods

Data for the case description was gathered from literal sources, by expert interviews, and by observations done by one of the authors when working as an ECM consultant for the

Finnish Parliament. The Internet service for the State Budget Proposal (http://budjetti.vm.fi/) was also utilized.

XML document management environment

The major information producers and users during the life of a State Budget Proposal are illustrated in Figure 5. Next we briefly describe the three kinds of information resources: systems, actors, and content items. More about the role of the resources in the information management is explained in the context of the life cycle description.

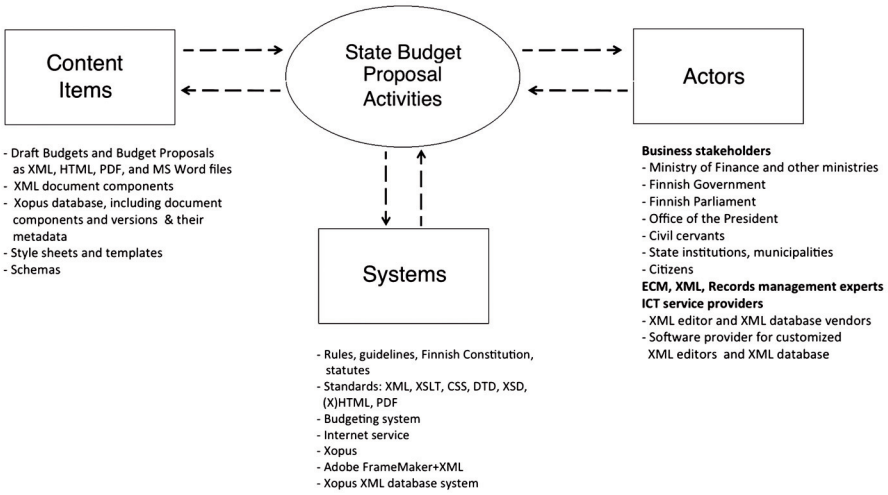

FIG. 5. XML document management environment for the State Budget Proposal.

Systems. The creation of the Budget Proposal is regulated by the Finnish Constitution and Acts, and many of the regulations are implemented in software systems. Besides XML, important XML-related standards that have been adopted in the environment. The budgeting system provides access to the financial data and spending

limits of the budget. Xopus XML editor is a tool for producing XML content and Adobe FrameMaker+XML for editing the Budget Proposal for printing.

Actors. The most important business stakeholders in the creation of the Budget Proposal are the Finnish Government and Parliament, and the decision-makers there. In the Government, Ministry of Finance has the central role. In the decision-making process the most important user of the Budget Proposal is the Finnish Parliament using it as the basis to create and accept the State Budget.

Contents items. The Budget Proposal consists of a number of separately produced content items. The State Budget Proposal as a whole typically consists of about 900 pages. The XML content is stored in the Xopus XML database, according to the component structure defined in the schema.

The life cycle of the State Budget Proposal

Design. The current solution is based on three major design cycles, the first starting in 1995. The analysis of the content management environment and requirements for the new solution was done in a research project where the first versions of RASKE methods were developed and tested. The project was participated by researchers, representatives from the Finnish Parliament, Ministry of Finance together with a few other ministries, and a software company. Preliminary schemas as SGML DTDs were designed for the major document types of the budgetary process by the researchers of the project. Later selected ICT service providers designed and implemented final SGML DTDs and customized software products for document authoring, visualization, and storage. The first solution concerned mainly the capture and dissemination phase of the document life cycle. The content production of the Draft Budgets remained unstructured, paper copies and text

editor (Word 6) files were used for distribution. Transformation into SGML format was done manually in the Ministry of Finance. The shift to structured content production took place in 1998, using Adobe FrameMaker+SGML software.

A few years later, problems in content authoring and managing motivated to start the second design cycle. Important objectives in the redesign included better support for last-minute changes, automated multi-channel publishing, and switch from SGML to XML. In the redesign project, an ICT service provider transformed SGML DTD into XML DTD and customized new software products: Microsoft Word with X4O extension for content production and XHive XML database for capture, dissemination and use. Ministry of Finance continued using Adobe FrameMaker for content authoring. Content production in the renewed environment started in 2003. The earlier produced SGML documents were transformed into the new XML format.

The third design cycle started at the time of the renewal of the budgeting system in 2008. Thus far major financial calculations for the budget had been done in a separate system and the results were copied manually into documents. The aim in the renewal was to integrate budget calculations, content production, and document capture into a single, centralized system. The new system was intended to support collaborative content production in all ministries and in the Office of the President. Furthermore, a goal was to provide new support for the Parliament for using the Budget Proposal. A selected ICT provider redesigned the XML schema and customized new software products: Xopus XML editor for content production and Xopus database for capture, dissemination and use. Adobe FrameMaker+XML was re-customized for preparing the print format.

Content production in the redesigned environment started in 2012. The documents produced during 1998-2002 were transformed to conform to the new schema.

As a result of the three design iterations, several XML editors and XML databases have been customized and many transformations have been implemented. From the users' point of view, the most important result is the Internet service. The design has been done in deep collaboration of ECM, XML, records management and business process experts. The foundation of the design work is the XML schema describing the structure of the State Budget Proposal.

Content production. Annually each of the 12 ministries, Office of the President, and the Finnish Parliament create their Draft Budgets entering the data to the budgeting system with the Xopus XML editor. The system allows parallel editing of different parts of the same document by different users. The structure of the document and related metadata is controlled by XML schemas. The content of the document is stored in the Xopus XML database. The related metadata includes information about the actors, document type and status of the content. Metadata is used, for example, to control the access to the content. The Draft Budgets are not available for other ministries until the Ministry of Finance has published the first version of the Budget Proposal.

Capture and dissemination. The Draft Budgets are assembled into a draft version of the State Budget Proposal using the XML content of the Xopus database. The final version is a result of negotiations between the ministries and decisions done in the Cabinet Finance Committee and Government sessions. The versions handled in the budgeting negotiations

of the ministries are transformed from the budgetary system into Microsoft Word XML or pdf formats. The final State Budget Proposal is published from the budgeting system on the Internet service in HTML and pdf formats. To the Parliament the document is submitted in the XML format and as printed books. The printed version is edited with Adobe FrameMaker+XML application. The metadata required for the creation of HTML pages and for supporting advanced searches is included in the document already in the content production phase.

Use. A new State Budget Proposal is of interest to state institutions, municipalities and citizens, because it regulates the government funding, taxes and loans for the next year. The Finnish Parliament creates and accepts the State Budget and thus is the main user of the State Budget Proposal. The Parliament captures the Proposal into its own Parliamentary System. Members of the Parliament use the printed book. For processing and reuse, the content of the Proposal is available in the budgeting system. Typically Members of the Parliament accept some part of the Proposal as it is, but propose some changes to the content. These changes are assembled in the Finance Committee into a Finance Committee Report. The content of the Committee Report is partly created by reusing the data of the Proposal. The content of the Proposal is reused again when preparing the final State Budget. The content of the Proposal is later used in the ministries when preparing Supplementary Budget Proposals or a new State Budget Proposal. Advanced search capabilities for these reuse needs have been implemented to the budgetary system and to the Internet service.

Retention. The Budget Proposal is archived permanently in paper format into external diary systems. All Budget Proposals that have been produced in structured format since 2002 are available on the Internet service. The three design iterations have resulted in three different schemas and correspondingly three classes of structured documents, each class originally conforming to its own schema. The selected retention strategy in the case has however been to transform the documents to conform to the latest schema. The transformations have concerned 13 Budget Proposals and about 50 Supplementary Budget Proposals. Between the major design iterations small changes have been done to schemas: some structures have been added while others have been removed.

Impact analysis

The implementation of the structured approach for the State Budget Proposal has improved effectiveness of the administration: the budget data management, financial calculations and content management activities are integrated into a single budgeting system. The technical content management process for authoring, publishing and delivering of the content is automated. Thus the last minute changes in the decision-making are better managed and copying mistakes in content production are avoided. Delivery of the documents between organizations is done in digital format, which is fast and lowers printing costs. Better availability of the Budget Proposals for citizens and interest groups on the Internet has enhanced transparency of public administration processes. The XML structure and associated metadata enables advanced search capabilities. Furthermore, the XML format can be offered as open data for processing by software systems.

The transfer from SGML to XML has enhanced the utilization of open standards and offered more choices when selecting suitable tools. However, lack of maturity of the tools has caused problems. Implementation of the structured approach in the environment has not been fast and easy: several iterations have been done during last ten years and in each increment more complexity has been added to the environment. The latest design took 3,5 years and the cost for the design and implementation of the unified budgetary system was 1,5 million euros.

The Case of the Faculty Council Meeting Agenda

The Faculty Council meetings of the Faculty of Information Technology at the University of Jyväskylä are called by Meeting Agendas 11-14 times each year. The agendas are created at the Faculty Office in an XML-based environment.

Data gathering methods

Data for the case description was collected by one of the authors. As a student she had participated in the development project were the XML-based agenda production started. Later she participated as a consultant in the update of the document production system. By participating in the projects she had gathered detailed knowledge of the environment and the solution. During these projects, data was collected by analyzing earlier agendas and meeting minutes, by interviewing the office personnel of the faculty, and by prototyping. After implementations information about new solutions was available in project reports.

XML document management environment

The major information producers and users during the life cycle of a Faculty Council Meeting Agenda are illustrated in Figure 6. Next we briefly describe them.

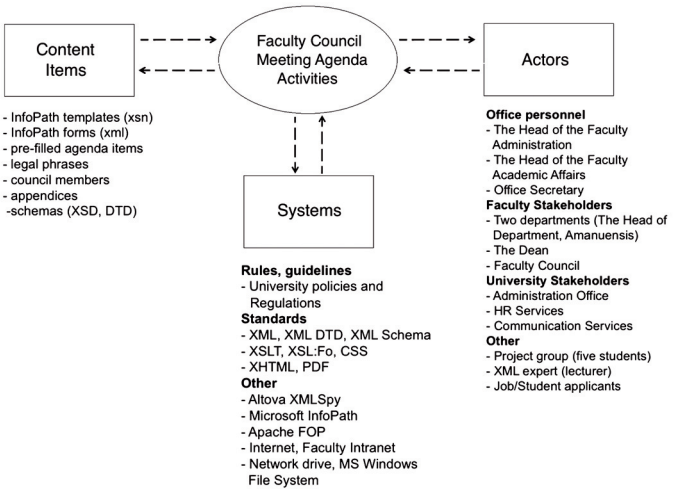

FIG. 6. XML document management environment for the Meeting Agenda.

Systems. The creation of Meeting Agendas is regulated by the rules concerning the decision-making process in the faculties and departments of the university. Besides XML, important standards that have been adopted in the environment are XML DTD, XML Schema, XSLT, CSS, XSL:FO, XHTML, and pdf. From the software systems Altova XMLSpy has been the tool for schema design and Microsoft InfoPath for authoring agendas.

Actors. Student groups and their supervisors have participated in the development of the XML-based solutions. The most important organizational business stakeholders involved in the creation and use of the Meeting Agendas are the two departments of the Faculty, the Faculty Office, the Faculty Council, and the University Administration

Office. In the Faculty Office mainly the Head of the Faculty Administration and the Head of the Faculty Academic Affairs prepare the issues for handling.

Content items. In the content repository the schemas, InfoPath templates, InfoPath forms, XSLT transformations, as well as the XSL and CSS style sheets represent class metadata. The agendas and related meeting minutes are stored in the network drive in XML format as InfoPath forms. The folders also include the meeting specific documents transformed from XML into HTML and pdf formats for publishing purposes. No separate instance metadata items are stored in the content item repository. Instance metadata related to a meeting agenda is embedded in the agenda. Also properties attached to files and folders, their names in particular, carry instance metadata.

The life cycle of the Faculty Council Meeting Agenda

Design. Problems in document production, publishing, and delivery were the main motivations for starting the design of an XML-based solution for Faculty Council Meeting Agendas in 2004. Since there was knowledge and experience about the use of the RASKE methodology at the faculty, the methodology was adopted for analyzing and describing the document management environment. No suitable predefined schema was publicly available at the time, therefore a decision was made to design custom schemas for the agenda front page, list of items page, and agenda items. In the design an important objective was to produce minimal changes to content authoring in the Faculty Office environment where MS Word had been the main authoring tool for a long time. Therefore the schemas were designed to be used with MS InfoPath.

The content authoring using InfoPath was facilitated by InfoPath form templates, associated with schemas. Several pre-filled InfoPath forms were created to support

content authoring. XSLT transformation definitions and CSS style sheets were designed for HTML and pdf outputs.

The XML-based document production started in 2005. For the first time in the history of the IT Faculty the entire agenda had a coherent layout. In the beginning of 2010, the schemas and InfoPath form templates were modified due to the university re-form. The update required only minor changes in the schemas and templates. In spite of that, the earlier pre-filled InfoPath forms conforming to the earlier template and schema versions could no longer be opened and used with InfoPath. Therefore new pre-filled forms had to be created. The latest upgrade of the solution was carried out in June 2012 together with the update of the MS Office Suite to the 2010 version. InfoPath forms receiving XML data from another form required a full trust setting. Forms that run with a full trust setting are digitally signed with a certificate. This allows a form to access data from external files. In the case of agendas, InfoPath forms required a full trust setting in order to reuse XML content properly between InfoPath forms as well as to authenticate the code that runs document assembly of each agenda.

Content production. The two departments of the faculty and the Faculty Office deliver the documents needed for handling in the next meeting. Typically the documents end up in the appendices of the Meeting Agenda as such. Some appendices may be received from external sources. If paper documents are received, they are scanned and saved as pdf documents. The Head of the Faculty Administration and the Head of the Faculty Academic Affairs, possibly by support of other Faculty Office staff, enter content to the agenda gradually using InfoPath forms. Typically there is a draft version of each agenda

item. All forms for items to be included in the agenda of an upcoming meeting are saved in the same folder on a network drive. Some of the text content may be copied from the published minutes of an earlier meeting available on the intranet. The agenda authors use the HTML versions of the archived documents for the needs of copying pieces of content. Legal phrases reoccurring in agenda items are retrieved from an external XML document, which can be modified in InfoPath when needed.

Capture and dissemination. Once the agenda items have been finalized, the separate document files are assembled into one agenda document. The document author carries out this task with the *list of items* form. The content of the form ends up as the second page of the agenda. A click of a button on the form runs required scripts to assemble the chosen XML-based InfoPath forms into one XML document file. The assembly includes also the execution of the format conversion, realized with XSL transformation files, to create HTML and pdf outputs of the assembled agenda. The document assembly software also adds reoccurring information on agenda pages automatically without any input from the document author. For example, the name of the document type, date of the meeting, and page numbers are added to each agenda item header as they are on the *list of items* form. The file names for the assembled documents in their different formats are generated automatically. For example, the pdf format of the agenda created on April 8th 2011 is named as "agenda.8.4.2011.pdf". The HTML and pdf formats of the agenda are published to the Faculty Council on the intranet, and the list of agenda items is delivered via email to the whole personnel of the faculty. Most of the agenda metadata is available on the agenda front page. It includes organizational and meeting-specific information. Each

agenda item comes with a header section including, for example, the name of the person who has prepared the item for the meeting.

Use. The published agenda is used by the Faculty Council members to support the decision-making in the meeting. After the meeting the Head of the Faculty Administration and the Head of the Faculty Academic Affairs use the agenda to create Meeting Minutes. The XML documents available as InfoPath forms for the agenda are complemented with new content, such as decisions made in the meeting and the names of the faculty council members present in the meeting. The minutes serve as evidence of and information of the decisions made in the meeting. Unlike the agenda, the minutes are public documents and therefore viewable for everyone. Access to the minutes appendices, however, is restricted.

Retention. Long-term archival is guided by the regulations of the university. The official archival records of the Faculty Council meetings are the meeting minutes. The minutes of all meetings during a year are collected in a printed book archived in the Faculty Office. The number of people involved in the management of agendas and related documents in the Faculty Office is small. Therefore simple rules concerning file and folder names, together with the rules concerning the metadata embedded in documents have been sufficient. No explicit preservation policies have been defined for the digital content repository. Instead of the earlier repository consisting mostly of MS Word files, the new content repository includes different kinds of documents in different formats. All document instances related to the agenda of a meeting are stored on a network drive in a

folder. The folder includes the XML documents (InfoPath forms) as well as assembled HTML and pdf output documents, both for the agenda and the corresponding minutes. None of the documents created for the Faculty Council meetings have ever been destroyed, following the current archival policy at the university.

The InfoPath form templates, XSL transformation files, CSS style sheets, and custom XML schemas are archived in their own folders on a network drive. Only the latest versions of the class metadata files are stored. The office personnel creating agendas has no need to access these folders for agenda content authoring.

Impact analysis

The design of the XML document management was activated by problems related to laborious document preparation and publishing process, and frequently occurring errors in Faculty Council Meeting Agendas. The main concern was in solving these problems. Therefore improvements in the work of people at the Faculty Office and decrease in the number of errors in the agendas were regarded as the main objectives to the development project. In the new environment the work is supported by the adaptation of the authoring environment and by the support of automation. Members of the office staff were briefly trained before the deployment of the new system. The transfer did not cause major problems or complains as the users were closely involved in the development process. This ensured commitment to the use of the new solution. In the content production phase the need for manual typing and copy pasting has decreased and the automated creation of the HTML format for online publishing has replaced the earlier laborious manual gathering of content to the HTML format. No systematic evaluation of the number of

errors in meeting agendas has been carried out, but most obviously the form-based data input with constraints controlled by the schema has decreased the number of errors.

In the case the focus in the development was in the work of document authors. The management of documents as records and their long-term archival was left outside the development project. The metadata embedded in documents in the front page and in page headings has followed the rules created already before the adoption of XML. Some new rules for document naming had to be decided for the new kinds of files in the content repository.

The daily work at the Faculty Office does not require XML knowledge. For occasional changes needed to XML schemas, InfoPath form templates, transformation files, or style sheets, an XML expert has been requested. So far such expertise has been available at the faculty but some concerns have been stated about the lack of policies for ensuring the needed XML knowledge.

Evaluation

Our main objective was to develop a model for analyzing and describing XML document management throughout document life cycle. Our development process was iterative. We first designed a preliminary model and then gathered data from the two cases. After several discussions and test descriptions we ended up to the life cycle consisting of the five phases as shown in Figure 4. It provided, together with the XML document management environment model (Figure 3), a good basis to analyze and describe the cases in a uniform way. One of the principles related to the artifact created in design science concerns the level of abstraction (Österle et al., 2011). According to this

principle the artifact should be applicable to a class of problems. Analysing parallel two different cases during the model development was an important means to avoid developing a model suitable just for one case.

In the model we want to emphasize the explicit inclusion of design and retention in the phases of the life cycle. The retention of business documents has become an important topic especially after the Sarbanes-Oxley Act of 2002 of the United States. The law mandates the retention of electronic documents and criminalizes the altering or destroying electronic records (Volonino, 2003; Stephens, 2005). Organizations are required to facilitate *e-discovery*, meaning the process of gathering electronic information for legal, regulatory, or administrative actions (Volonino et al., 2007). The retention of electronic records is important also in public sector. In Finland, for example, the law about electronic public sector services (24.1.2003/13) requires the archival of electronic documents so that their authenticity and integrity can be later demonstrated. Lack of understanding of the special features of XML document management environments may cause problems for retention and e-discovery. One of the special features of XML document management environments is their dynamics. It is typical to XML document management environments that the design is an ongoing process. Therefore we see it important to show the design phase explicitly in the model as the first phase.

We believe that our framework provides a tool both for researchers and practitioners to compare different cases. Our example cases have some similarities but are contrasting in many respects. The design of the document production was in both cases an iterative process involving schema updates. The storage and retention solutions in the two cases are quite unlike. In the Faculty case the XML data of the meeting agendas is stored in the

file system at the intranet server while the State Budget Proposal is stored in an XML database. No special retention strategies have been designed for the digital format of Faculty Council Agendas and no changes have been done to the old documents at the time of the schema update. For the Budget Proposals the strategy in schema updates has been to transform the old documents to conform to the latest schema.

The feedback from the case environments showed that the case descriptions were seen as useful tools in the environments to analyze the solutions and also as tools to record the core characteristics of the environments.

Conclusion

In the paper we first introduced the concepts related to XML document management and described methodologies that have earlier been used to analyze XML document management in organizations. We argued that the methodologies lack support for the analysis of XML document management throughout their life, from the design to the retention. To fill the gap, we developed a model for the purpose, as an extension of the RASKE methodology, following the design science approach.

During the development of the model we tested its applicability only in cases where the deployment of XML had already been implemented. Further research is needed to show how the model could support the design of new solutions for XML document management. Basically there are two different kinds of support. On the one hand, earlier case descriptions can be used as a means to learn about XML document management and earlier implementation cases. On the other hand, the concepts and models of the

framework may be used as a tool to describe the new XML document management solutions.

Both in public and private sector organizations there is a strong tendency to shift from paper archival to digital archival. At the same time, new legislation is created to mandate the preservation of digital information produced in business processes. Especially in case of legal auctions organizations are required to discover all relevant documents and demonstrate their reliability and authenticity. E-discovery is a complex and often costly procedure. An interesting area of future research is to consider XML document management especially from the point of view of e-discovery.

As mentioned earlier, the RASKE methodology has provided models and methods for holistic analysis of document management environments. It has some special support for analyzing structured documents, but its use is not restricted to XML-encoded documents. The same concerns also the life cycle model introduced in this paper. The five phases of Figure 4 are not restricted to XML documents. Therefore the model is applicable in analyzing electronic document management also more generally.

References

Aversano, L., Canfora, G., de Lucia, A., & Gallucci, P. (2002). Integrating document and workflow management tools using XML and Web technologies: A case study. In T. Gyimóthy, & F. Brito e Abreu (Eds.), Proceedings of the Sixth European Conference on Software Maintenance and Reengineering (CSMR'02). Washington, DC: IEEE.

Barker, R.M., Cobb, A.T., & Karcher, J. (2009). The legal implications of electronic document retention: Changing the rules. Business Horizons 52(2), 177-186.

Bernard, R. (2007). Information lifecycle security risk assessment: A tool for closing security gaps. Computers & Security 26(1), 26-30.

Bernstein, P.A., & Haas, L.A. (2008). Information integration in the enterprise. Communications of the ACM 51(9), 72-79.

Bertino, E., & Ferrari, E. (2002). Secure and selective dissemination of XML documents. ACM Transactions on Information and System Security 5(3), 290-331.

Bertino, E., Ferrari, E., Paci, F., & Provenza, L.P. (2007). A system for securing push-based distribution of XML documents. International Journal of Information Security 6(4), 255-284.

Bhatti, R., Ghafoor, A., Bertno, E., & Joshi, J.B.D. (2005). X-GTRBAC: an XML-based policy specification framework and architecture for enterprise-wide access control. ACM Transactions on Information and System Security 8(2), 187-227.

Borglund, E.A.M. (2008). Design for recordkeeping: areas of improvement. PhD Thesis, Department of Natural Sciences, Mid Sweden University Doctoral Thesis 52. Retrieved April 18, 2013, from http://miun.diva-portal.org/smash/get/diva2:1952/FULLTEXT01

Bray, T., Paoli, J., & Sperberg-McQueen, C. M., (Eds.) (1998). Extensible Markup Language (XML) 1.0, W3C Recommendation, 10 February 1998, W3C Consortium. Retrieved April 18, 2013, from http://www.w3.org/TR/1998/REC-xml-19980210.

Broberg, M. (2004). A successful documentation management system using XML. Technical Communication 51 (4), 537-546.

Brooke, P.J. Paige, R.F., & Power, C. (2010). Document-centric XML workflows with fragment digital signatures. Software – Practice and Experience 40(8), 655-672.

Brooks, T.A. (2001). Where is meaning when form is gone? Knowledge representation on the Web. Information Research 6 (2).

Cerri, D., & Fuggetta, A. (2007). Open standards, open formats, and open source. Journal of Systems and Software 80(11), 1930-1937.

Chen, M. (2003). Factors affecting the adoption and diffusion of XML and Web services standards for E-business systems. Human-Computer Studies, 58(3), 259-279.

Clark, J., & Murata, M. (Eds.) (2001). RELAX NG Specification, Committee Specification 3 December 2001, OASIS. Retrieved April 18, 2013, from http://www.oasis-open.org/committees/relax-ng/spec-20011203.html.

DLM Forum Foundation (2011). MoReq2010®: Modular Requirements for Records Systems – Volume 1: Core Services & Plug-in Modules. Retrieved April 18, 2013, from http://moreq2010.eu/

Dublin Core Metadata Initiative (2010). Dublin Core Metadata Element Set, Version 1.1. 2010-10-11. Retrieved April 18, 2013, April 18, 2013, http://dublincore.org/documents/dces/

Genevès, P., Layaïda, N., & Quint, V. (2011). Impact of XML schema evolution. ACM Transactions on Internet Technology 11 (1), 4:1-4:27.

Gilliland, A.J. (2008). Setting the Stage. In T. Gill, A.J. Gilliland, Whalen, M., & Woodley, M.S., Introduction to Metadata, Online Edition, Version 3.0. Los Angeles, CA: Getty Publications. Retrieved April 18, 2013, from http://getty.edu/research/publications/electronic_publications/intrometadata/setting.html

Glushko, R.J., & McGrath, T. (2005). Document Engineering: Analysing and Designing Documents for Business Informatics and Web Services. Cambridge, MA: MIT Press.

Goldfarb, C. F. (1990). The SGML Handbook. Oxford: Oxford University Press.

Greenberg, J. (2010). Metadata and digital information. In M.J. Bates & M. Niles Maack (eds.), Encyclopedia of Library and Information Science, Third Edition, 1:1 (pp. 3610-3623). New York: Taylor & Francis.

Gregor, S., & Jones, D. (2007). The anatomy of a design theory. Journal of the Association for Information Systems 8(5), 312-335.

Heslop, H., Davis, S., & Wilson, A. (2002). An approach to the preservation of digital records. national Archives of Australia. Retrieved April 18, 2013, from http://www.aa.gov.au/Images/An-approach-Green-Paper_tcm16-47161.pdf

Hevner, A.R., March, S.T., & Park, J. (2004). Design research in information systems research. MIS Quarterly 28(1), 75-105.

Horik, van, R., & Roorda, D. (2011). Migration to intermediate XML for electronic data (MIXED): Repository for durable file format conversions. The International Journal of Digital Curation 2(6), 245-252.

ISO 15489-1 (2001). Information and documentation – Records management. Part 1: General.

ISO/TR 15489-2 (2001). Information and documentation – Records management. Part 2: Guidelines.

ISO/IEC 26300 (2006). Information technology – Open Document Format for Office Applications (OpenDocument) v1.0.

ISO/IEC 29500-1 (2008). Information technology – Document description and processing languages – Office Open XML File Formats – Part 1: Fundamentals and Markup Language Reference.

Järvenpää, M., Virtanen, M., & Salminen, A. (2006). Semantic portal for legislative information. In M. Wimmer, H.J. Scholl, Å. Grönlund, K. Viborg Andersen (Eds.), Proceedings of the Fifth International Conference on Electronic Government (EGOV 2006). Lecture Notes in Computer Science 4084 (pp. 219-230). Berlin: Springer Verlag.

Kundu, A., & Bertino, E. (2008). A new model for secure dissemination of XML content. IEEE Transactions on Systems, Man, and Cybernetics 3 (3), 292-301.

Kwietniewski, M., Gryz, J., Hazlewood, S., Van Run, P. (2010). Transforming XML documents as schemas evolve. Proceedings of the VLDB Endowment 3(1-2), 1577-1580.

Liu, H., Wang, X., & Quan, Q. (2009). Research on the enterprise' model of information lifecycle management based on enterprise architecture. Proceedings of the Ninth International Conference on Hybrid Intelligent Systems (pp. 165-169). New York, NY: IEEE Computer Society.

Liu, S., McMahon, C.A., & Culley, S.J. (2008). A review of structured document retrieval (SDR) technology to improve information access performance in engineering document management. Computers in Industry 59(1), 3-16.

Luk, R.W.P., Leong, H.V., Dillon, T.S., Chan, A.T.S., Croft, W.B., & Allan, J. (2002). A survey in indexing and searching XML documents. Journal of the American Society for Information Science and Technology 53(6), 415-437.

March, S.T. & Smith, G.F. (1995). Design and natural science research on information technology. Decision Support Systems 15 (4), 251-266.

Maler, E., & El Andaloussi, J. (1996). Developing SGML DTDs. From text to model to markup. Englewood Cliffs, NJ: Prentice Hall.

Mella, G., Ferrari, E., Bertino, E., & Koglin, Y. (2006) Controlled and cooperative updates of XML documents in byzantine and failure-prone distributed systems. ACM Transactions on Information and System Security 9(4), 421-460.

Ministry of Government Administration, Reform and Church Affairs (2007). Open document standards to be obligatory for state information. Press release. Retrieved April 18, 2013, from http://www.regjeringen.no/en/dep/fad/pressesenter/pressemeldinger/2007/Open-document-standards-to-be-obligatory.html?id=494810

Noy, N.F., & Hafner, C.D. (1997). The state of the art in ontology design. A survey and comparative review. AI Magazine 18(3), 53-74.

Nunamaker, J.F., Chen, M., & Purdin, T.D.M. (1990). Systems development in information systems. Journal of Management Information Systems 7(3), 89-106.

Nurmeksela, R., Jauhiainen, E., Salminen, A., & Honkaranta, A. (2007). XML document implementation: Experiences from three cases. In Y. Badr, R. Chbeir, & P. Pichappan (Eds.), Proceedings of the Second International Conference on Digial Information Management (pp. 224-229). Los Alamitos, CA: IEEE.

OCLC and CRL (2007). Trustworthy Repositories Audit & Certification: Criteria and Checklist. Version 1.0. Retrieved April 18, 2013, from http://www.crl.edu/sites/default/files/attachments/pages/trac_0.pdf

Peffers, K., Tuunanen, T., Rothenberger, M.A., & Chatterjee, S. (2008). A design science research methodology for information systems research. Journal of Management Information Systems 24(3), 45-77.

Peled, A. (2011). When transparency and collaboration collide: The USA Open data Program. Journal of the American Society for Information Science and Technology 62(11), 2085-2094.

Pérez, J.M., Berlanga, R., Aramburu, M.J., & Pedersen, T.B. (2008). Integrating data warehouses with Web data: A survey. IEEE Transactions on Knowledge and Data Engineering 20(7), 940-955.

Powell, A., & Johnston, P. (2003). Guidelines for implementing Dublin Core in XML. Dublin Core Metadata Initiative. Retrieved April 18, 2013, from http://dublincore.org/documents/dc-xml-guidelines/

Reuben, E. (2003). Migrating records from proprietary software to RTF, HTML, and XML. Computers in Libraries 23(6), 30-33.

Rockley, A., Kostur, P., & Manning, S. (2003). Managing Enterprise Content: A Unified Content Strategy. Indianapolis, IN: New Riders.

Runardotter, M., Mörtberg, C., Mirijamdotter, A. (2011). The changing nature of archives: whose responsibility? Electronic Journal of e-Government 9(1), 68-78.

Rönnau, S., & Borghoff, U.W. (2009). Versioning XML-based office documents. An efficient, format-independent, merge-capable approach. Multimedia Tools and Applications 43(3), 253-274.

Salminen, A. (2005). Building digital government by XML. In R.H. Sprague, Jr. (Ed.), Proceedings of the Thirty-Eighth Hawaii International Conference on System Sciences (HICSS-38). Los Alamitos, CA: IEEE Computer Society.

Salminen, A. (2010). Modelling documents in their context. In M.J. Bates & M. Niles Maack (Eds.), Encyclopedia of Library and Information Sciences, Third Edition. New York: Taylor & Francis. DOI: 10.1081/E-ELIS3-120044399.

Salminen, A., Kauppinen, K., & Lehtovaara, M. (1997). Towards a methodology for document analysis. Journal of the American Society for Information Science 48(7), 644-655.

Salminen, A., Lyytikäinen, V., & Tiitinen, P. (2000). Putting documents into their work context in document analysis. Information Processing & Management 36(4), 623-641.

Salminen, A., Lyytikäinen, V., Tiitinen, P., & Mustajärvi, O. (2004). Implementing digital government in the Finnish Parliament. In W. Huang, K. Siau, & K.K. Wei (Eds.), Electronic Government Strategies and Implementation (pp. 242-259). Hersley, PA: IDEA Group Publishing.

Salminen, A., Nurmeksela, R., Lehtinen, A., Lyytikäinen, V., & Mustajärvi, O. (2008). Content production strategies for e-Government. In A.-V. Anttiroiko (Ed.), Electronic Government: Concepts, Methodologies, Tools, and Applications. Hersley, PA: Information Science Reference.

Salminen, A., & Tompa, F.W. (2001). Requirements for XML document database systems. In E.V. Munson (Ed.), Proceedings of the ACM Symposium on Document Engineering (DocEng '01) (pp. 85-94). New York: ACM Press.

Salminen, A. & Tompa, F. (2011). Communicating with XML New York: Springer-Verlag New York Inc.

Sartor, G., Palmirani, M., Francesconi, E., & Biasiotti, M. A. (Eds.) (2011). Legislative XML for the Semantic Web. Law, Governance and Technology Series 4. Dordrecht: Springer.

Shah, R., Kesa, J., & Kennis, A. (2008). Implementing open standards: A case study of the Massachusetts open formats policy. In Proceedings of the 2008 International Conference on Digital Government Research (pp. 262-271). Los Angeles, CA: Digital Government Society of NorthAmerica.

Smith, H.A., & McKeen, J.D. (2003). Developments in practice viii: enterprise content management. Communications of AIS 11(33), 1-26.

Sohn, W.-S., Ko, S.-K., Lee, K.-H., Kim, S.-H., Lim, S.-B., & Choy, Y.-C. (2002). Standardization of eBook documents in the Korean industry. Computer Standards & Interfaces 24(1), 45-60.

Sprague Jr., R.H. (1995). Electronic document management: Challenges and opportunities for information systems managers. MIS Quarterly 19(1), 29-49.

Stanescu, A. (2005). Assessing the durability of formats in a digital preservation environment: The INFORM methodology. OCLC Systems & Services, 21 (1), 61 – 81.

State Records Authority of New South Wales (2007). Strategies for documenting government business: The Dirks Manual. Retrieved April 18, 2013, from http://www.records.nsw.gov.au/recordkeeping/dirks-manual

Stephens, D.O. (2005). The Sarbanes-Oxley Act: Records management implications. Records management Journal 15(2), 98-103.

Tiitinen, P., Lyytikäinen, V., Päivärinta, T., & Salminen, A. (2000). User needs for electronic document management in public administration: a study of two cases. In H.R. Hansen, M. Bichler, & H. Mahrer (Eds.), Proceedings of ECIS 2000, European Conference on Information Systems, Volume 2 (pp. 1144-1151). Wien: Wirtschaftsuniversität Wien.

Volonino, L. (2003). Electronic evidence and computer forensics. Communications of the Association for Information Systems 12 (Article 27), 457-468.

Volonino, L., Sipior, J.C., & Ward, B.T. (2007). Managing the lifecycle of electronically stored information. Information Systems Management 24(3), 231-238.

Wang, F., & Zaniolo, C. (2008). Temporal queries and version management in XML-based document archives. Data & Knowledge Engineering 65(2), 304-324.

Österle, H., Becker, J., Frank, U., Hess, T., Karagiannis, D., Krcmar, H., …, & Sinz, E.J. (2011). Memorandumdesign-oriented on information systems research. European Journal of Information Systems 20 (1), 7-10.