Constructing tailored SGML documents

Helena Ahonen
Barbara Heikkinen
Oskari Heinonen
Jani Jaakkola
Pekka Kilpeläinen
Greger Lindén
Heikki Mannila
Department of Computer Science, University of Helsinki
P.O.Box 26
Teollisuuskatu 23
FIN-00014 UNIVERSITY OF HELSINKI
Finland
kilpelai@cs.helsinki.fi
Phone: +358 9 708 51
Fax: +358 9 708 44441
http://www.cs.helsinki.fi/research/rati/

August 20th, 1996

A tailored document corresponds to the need of a certain user group or user task. Tailored documents may be constructed through document assembly from a pool of documents. An intelligent document contains information that supports this assembly. We suggest different kinds of information that may be associated with an intelligent document, and used in the assembly process. We also study the assembly process itself, and the transformations needed to form the tailored document from fragments of documents. We report on three case studies, where intelligent document assembly methods are being prototyped on commercially used document material. As a basis for the project we consider documents marked up with SGML (Standard Generalized Markup Language).

Keywords: document assembly, intelligent documents, SGML.

1 Introduction

The role and even the concept of documents is undergoing a tremendous change. A document is not only created by a user but also collected, combined and customized from reusable documents [Lev93]. The multi-purpose reuse of documents is facilitated by SGML [ISO86], which has, over a ten-year period, slowly become the predominant document markup standard in the electronic publishing area. The standard does not, however, provide a solution for managing or processing large document collections. Common problems are finding the right version of the right document, and keeping related documents compatible with each other, or synchronizing documents with the latest data. Other problems include reusing appropriate existing document fragments, and coordinating multiple access and update of documents [Ang93].

One of the basic problems is to provide on-demand generation of individualised documents through dynamic document assembly. A *document assembly* composes new documents from a collection of documents. Such reuse includes finding the relevant document fragments, modifying them as needed, and stitching the fragments together to a new document. An assembled document can be a base for new assemblies. Document assembly can be fully automatic or *automated*, or it may be performed *on demand* by the user. Document assembly is *intelligent* when it uses application-domain-specific information about the document in addition to the contents and the structure. Intelligent assembly can be implemented by constructing agents which know how to use supplementary information to retrieve document fragments. Document assembly is *dynamic* when the result depends on the user's demands or on the state of the environment. An assembled document may be *individualised* or *tailored* satisfying the needs of a certain user group.

In an on-going research and development project called *Structured and Intelligent Documents* (SID (The Structured and Intelligent Documents Project)) we study intelligent document assembly in its different aspects. More precisely, the goals of the SID project are:

- to define the supplementary information needed for building active and intelligent documents, and for their intelligent assembly,
- to develop tools for intelligent assembly,
- and to develop a methodology for introducing supplementary information into a document collection.

Especially, we want the assembled document to be a finished product to be delivered either electronically or on paper. As a basis for the project we consider structured documents marked up using SGML. The project is funded by the Finnish Technology Development Centre and seven Finnish enterprises (The Aamulehti Group, Oy Edita Ab, The National Board of Education, WSOY, Helsinki Media, Lingsoft, Inc., and MTV3).

In the rest of this paper we analyse the assembly process and some examples of documents that contain supplementary information for intelligent document assembly.

2 Intelligent document assembly

We define the document assembly to consist of three phases. In the first phase, the user must express the requirements: what kind of a document (s)he wants, what the result should contain, how large it should be, etc. For an on-demand assembly, the user gives a set of conditions. In an automated assembly, the conditions have already been specified before-hand. One approach for expressing the conditions is to extend the notion of a document, and see it as a user interface for assembling documents. The search or assembly mechanism is then merged into the documentation. For example, WWW

(World Wide Web) servers with associated CGI (Common Gateway Interface) scripts or Java programs can support the construction of such document interfaces.

The second phase performs the assembly. Appropriate document fragments are found and returned. A document fragment is returned if it either fulfils the user's demands, or if it is heuristically deemed relevant. For example, in addition to a paragraph returned from a textbook, we could also return some fragments that form its context. Suitable additional fragments might include section titles and their opening paragraphs, or paragraphs containing definitions for the terms used in the chosen fragments.

The third phase merges the document fragments by transforming them into a single uniform assembled document. This result can then, e.g., be displayed on the screen or printed on paper. If the assembled document is to be used as an SGML document, a new DTD may need to be constructed. The assembled document may then further need to be transformed to conform to this new DTD.

Assembling a document is done in an iterative process, where the user might want to reject some intermediate assembly parts and set new, more precise query conditions. The final document consists of the information the user has collected during one or more assembly processes.

The assembly is performed by different *software agents* [CACM95] working on the document and its supplementary information. A general agent assists in searching and reading documents. A specialized agent handles application-specific information. Agents are implemented in such a general and open way that they easily can be used on different platforms and systems. Implementations can be made using, e.g., script languages, or, in the Java language [GM95] which provides a safe device-independent way to express portable actions. Agents also provide connections to commercial transformation languages and converters. To address the problem of document transformation, which is inherent in the assembly process, the project is also implementing a new powerful SGML transformation language called TranSID [JKL96].

3 Case studies

Document assembly in practice is being prototyped and developed in several case studies. The case studies focus on intelligent assembly of law collections, tailored textbooks and newspaper articles. All cases are based on material that is produced and used in commercial circumstances, for producing CD-ROMs, books or newspapers. The case studies apply different methods to the intelligent assembly of the structured document material, but eventually we will try to merge the best techniques together.

3.1 Semantic clustering of statutes

In a law book application we are dealing with a collection of about 800 Finnish statutes marked up using SGML. This kind of material is quite fertile as a source of document assembly, because of its large number of independent, yet strongly related units of textual information. We are developing an assembly environment that allows the user of the material to browse the contents of the collection, and to pick the appropriate pieces to be included in the final product, e.g., a collection of statutes relevant to some specific area of commerce. The browsing tool employs the Scatter/Gather clustering techniques developed by Cutting et al. [CPKT92][CKP93].

The documents are broken off into proper subdocuments in the preprocessing phase. This is done according to the structure of the documents; for example, parts and chapters of the statutes are eligible retrieval units. We add *supplementary information* to the documents by computing keyword lists for the subdocuments. The keyword lists are computed with the assistance of the `fintwol` parser, recording only base forms of nouns and their frequencies and removing stop words (frequent unimportant words). The keyword lists are then used as a basis for the clustering process.

The user-controlled iterative clustering process forms a classification of the material, and gives the user an extracted view of the documents. This is especially invaluable when the user is dealing with an unfamiliar collection of documents; traditional query models do not help the user to achieve a simple overview. The retrieved clusters, i.e., a classification, is presented to the user in a form. The user can choose the appropriate clusters for reclustering, and in this way narrow down the retrieval. Similar methods for browsing large collections of text documents based on their contents have been recently developed also in the WEBSOM interface [KHLK96]. The Scatter/Gather approach is different, however, since it allows the user to recluster and browse large scattered portions of the material at once, instead of choosing a single branch in a precomputed classification.

The document fragments found and chosen by browsing (or larger elements containing the fragments) can then be retrieved for the final assembly process.

The set of documents intended for the clustering process need not be the entire document collection. The documents can be selected by a query for which the matching elements of the collection are returned. Here a Finnish dictionary of general indexing terms [Kar94] is used as a search-aid thesaurus. In other words, the terms given by the user are expanded by related terms listed in the thesaurus, which, naturally, broadens the selection of initial documents.

3.2 Tailored educational textbooks

The second case study develops educational material for use at technical schools, colleges, and universities. The overall goal is to create a material pool that contains textbooks, articles, exercise collections, and simulations to be used in multiple ways. The publisher may print customized textbooks for various branches and levels. Additionally, teachers can form tailored compositions of CD-ROM delivered material - e.g., summaries, slides or tests - for their own class. Students could also use the pool directly for problem-oriented learning, as a data bank, or as an active textbook with computer-assisted learning features.

The first component of the material pool is a text book on control engineering. The book, originally prepared using Word for Windows, has been converted into SGML. The basic structure based on the DTD for books in the ISO 12083 standard [ISO94] extended with element types for exercises, answers, examples, formulas, definitions and clauses.

The book is enriched with internal supplementary information. Manual seeding, based on information from the authors, was necessary to classify the fragments of the material into appropriate categories. Each text paragraph is classified according to level (vocational, college, university), branch of specialization (metal industry, process industry, data communications, automation, electric power, civil engineering) and didactic form of the contents (introductory, base material, advanced, applied, summary, auxiliary, examination). Keyword elements are being added automatically to the paragraphs containing occurrences of the terms that have a definition (normally recognizable as boldface words or phrases) in the book.

The categorization is represented in the text paragraphs and their superstructures as SGML attributes. Attribute values are propagated from the upper levels of the structure, if not overridden in the element. The program below is an example of the supplementary information (in boldface) attached to the markup of the material.

<CHAPTER id="c4" **author="aJS" level="b"**>

<NO>4</NO><TITLE>SÄÄTÖLOHKOT TAAJUUSTASOSSA</TITLE>

<SECTION id="c4.1" **level="b" content="B"**>

<NO>4.1</NO><TITLE>Impedanssi-kompleksinen vastus</TITLE>

<EXAMPLE id="c4ex1" level="c" content="C">

<TITLE>Esim.**</TITLE>**

<FIGGRP><FIG id="c4fi2" name="c4f2" origfile="graf0001.wmf">

<NO>4.2</NO><TITLE>Impedanssin sisältämä virtapiiri.</TITLE>

</FIGGRP>

<P>**<KEYWORD freq=1>kokonaisimpedanssi</KEYWORD>**

Laske sarjakytkennän kokonaisimpedanssi U = 5 V</P>

<ANSWER>

<P>**<KEYWORD freq=1>kokonaisimpedanssi</KEYWORD>**

Kokonaisimpedanssi on sarjaankytkettyjen impedanssien summa,
samoin kuin tasavirtavastusten summa on osavastusten summa.</

<DFORMULA id="c4df13" notation="tex">

<DFIG name="c4d13">

<DFBODY>$$\bar Z=\bar Z_R+\barZ_C$$**</DFBODY>**

</DFORMULA></P>

...

This chapter has been authored by Jari Savolainen (author=aJS). It is directed at students at the college level (level=b). The contents of the section 4.1 is base material (content=B). The example of the section is directed at students at the university level (level=c) and it contains advanced material (content=C). Keywords and their frequencies are listed explicitly. Formulas are both in TeX format (element `dfbody`) and in picture files (entity `name` of element `dfig`). We can easily extract, e.g., base material for college level students, or exercises for university level students. Search features of existing commercial SGML browsers support this kind of querying. We have implemented a prototype assembly architecture, where the sgrep text search program is used for retrieving appropriate fragments of the textbook [JK96].

Individualised documents are assembled based on their contents, structure, attributes and keywords. In order to achieve sensible results, the assembly has to utilize heuristics which guarantee the inclusion of all necessary fragments. We have designed a simple form-based assembly interface for the users, although it may be useful to offer some pre-selected assemblies as well. The user will be able to modify assembled views by including or removing fragments.

3.3 Summarizing accounts reports

In the third case study, we have designed and implemented a prototype environment for assembly of newspaper articles. The articles chosen for the prototype are annual accounts reports of enterprises. The articles were originally created using the QuarkXPress page layout software. Each of these articles contains a summary of the financial results of an enterprise in a table form. The tables of related enterprises are naturally attractive for comparison.

To start with, the articles are converted to a structured form using tools developed in the project. The conversion adds supplementary information that can be found in an external data source. This information includes the current date and some extra information about the enterprise described in the article. Currently this information is picked from small SGML files created by hand for each article, but we are planning to develop automated user-assisted methods for seeding intelligence into the articles. The article structure is based on the Universal Text Format, which is an SGML DTD developed for the interchange of news [Bec94].

The assembly environment consists of the structured articles with their supplementary information, of scripts which perform the actual assembly, and of a user interface for expressing the assembly conditions and viewing the results. The user describes the result of the assembly by specifying the relevant articles, the elements of the chosen articles to be included in the assembly, and by expressing how the tables in the articles should be combined for the result. The selection of the articles can be done either manually from a list, or by giving a search condition on phrases occurring in specified elements of the articles. For example, one could choose the articles whose keyword elements contain phrases like *sugar* or *domestic goods*.

The assembly tool combines the summary tables of the chosen articles, which is useful for comparing the reported results of the enterprises of interest. The user can choose the enterprises and the data columns of the original tables that are to be included in the summary tables.

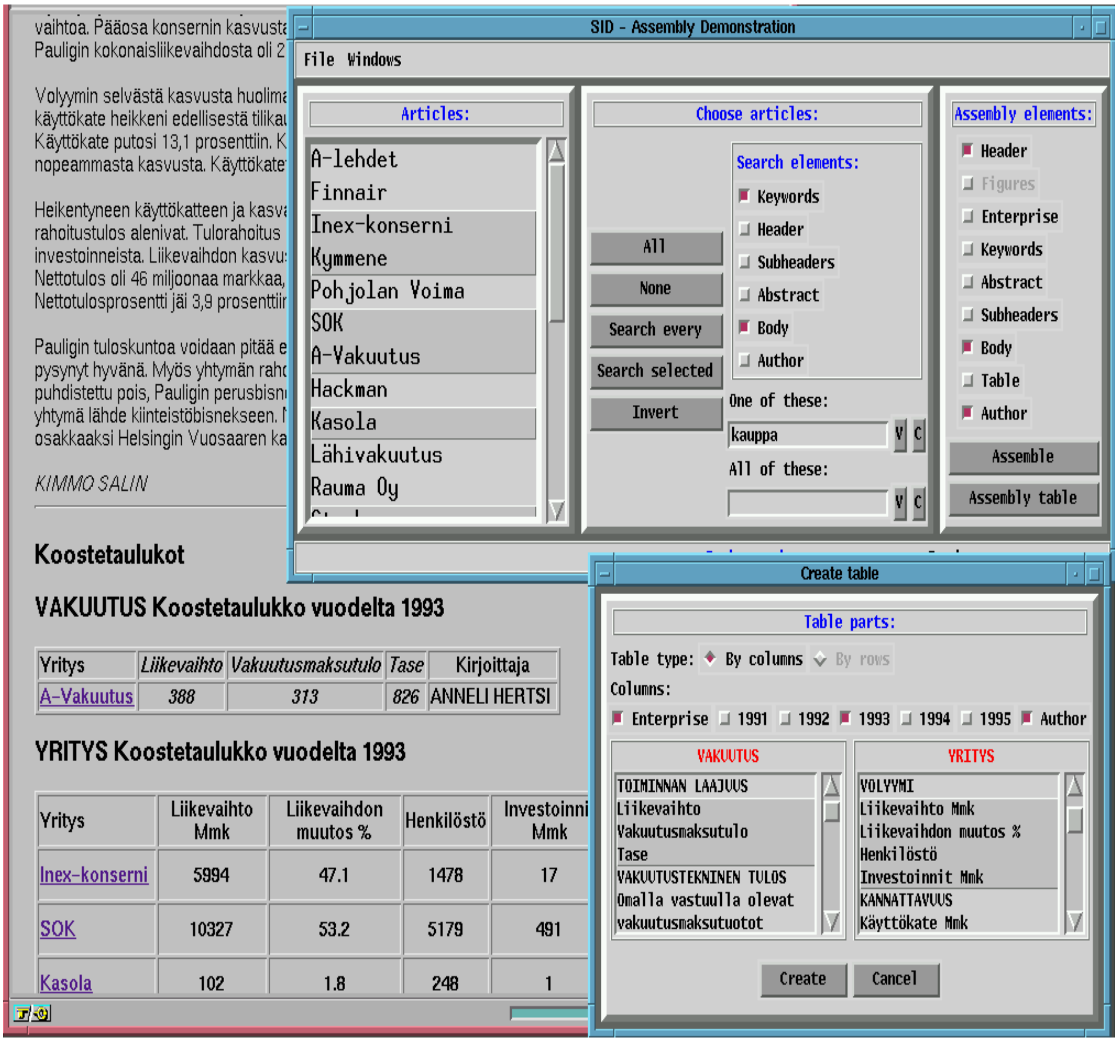

Figure 3.1 The user interface of the assembly tool

Figure assembly gives an example of the use of the system. Here the user has chosen the articles whose `Keywords` or `Body` elements include the word `kauppa` (*trade* or *commerce*). Of these articles the user has asked the system to show their `Header`, `Body` and `Author` elements, and to combine the tables by their column for the year 1993. The resulting document can be seen behind the two control windows. For each of the enterprises described in the chosen articles (A-Vakuutus, Inex-konserni, SOK, etc.), there is a row in the summary table. In fact, this assembly produces two separate tables, since the summary data of insurance companies, like A-Vakuutus, differs from that of the ordinary enterprises.

The selection of articles is implemented using a structured text search tool developed by the authors [JK96]. Element selection, conversion and gathering is implemented using MetaMorphosis scripts, which take the assembly parameters from the graphical user interface. For visualization purposes the assembled articles are converted to HTML (Hypertext Markup Language), which makes them instantly ready for on-line publishing on the Web.

4 Conclusion

We have described several ways of seeding documents with intelligence. Using either inherent or supplementary information associated with the document, allows us to perform intelligent document assembly to produce individualised documents on demand. Inherent information is based either on the text contents, the semantics or the structure of the document. Supplementary information can be added in the form of references to external information. A number of case studies have been implemented for testing and further developing the principles of intelligent document assembly.

One of the open problems in structured document assembly is the computing of a new DTD for the assembled document. In the general case, the new document will not follow any of the DTDs of the document fragments that are a result from the document query. An instance-specific DTD can always be found for the assembled document. If a more general DTD is needed, it could be computed from the DTDs of the underlying documents and the assembly query. Another open question is the copyright of the assembled document. In the general case, the document may include fragments written by different authors and even copyrighted by different companies. This is, of course a problem that occurs in other reuse as well.

5 Trademarks

Java is a trademark of Sun Microsystems, Inc. FINTWOL is a product of Lingsoft, Inc. Word for Windows is a trademark of Microsoft, Inc. QuarkXPress is a trademark of Quark, Inc. MetaMorphosis is a product of the MID/Information Logistics Group GmbH.

References

[Ang93] User requirements for SGML data management. Paula Angerstein. In Michael Popham, editor, The SGML '93 Conference Report. 1993.

[Bec94] UTF [Universal Text Format] - An SGML standard for the news distribution industry. Dave Becker. Technical report, Mead Data Central. August 1994.

[CACM95] Special issue on intelligent agents. Communications of the ACM, 37(7). July, 1995.

[CPKT92] Scatter/Gather: A cluster-based approach to browsing large document collections. Douglas Cutting, Jan Pedersen, David Karger and John Tukey. In Proceedings of the 15th ACM SIGIR Conference, pages 318-329. June 1992.

[CKP93] Constant interaction-time Scatter/Gather browsing of very large document collections. Douglas Cutting, David Karger and Jan Pedersen. In Proceedings of the 16th ACM SIGIR Conference, pages 126-134. June 1993.

[GM95] The Java Language Environment: A White Paper. James Gosling and Henry McGilton. Sun Microsystems Computer Company. May 1995.

[ISO86] Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML), ISO 8879. ISO. 1986.

[ISO94] Information and documentation - Electronic manuscript preparation and markup, ISO 12083. ISO. 1994.

[ISO96] Information technology - Processing Languages - Document Style Semantics and Specification Language (DSSSL) ISO/IEC DIS 10179. ISO. 1996.

[JK96] Using sgrep for querying search structured text. Jani Jaakkola and Pekka Kilpeläinen. In this volume. October, 1996.

[JKL96] TranSID - A language for transforming SGML documents. Jani Jaakkola, Pekka Kilpeläinen and Greger Lindén. Internal report, Department of Computer Science, University of Helsinki. June, 1996.

[KHLK96] Creating an order in digital libraries with self-organizing maps. Samuel Kaski, Timo Honkela, Krista Lagus and Teuvo Kohonen. Submitted to WCNN'96, the World Congress on Neural Networks, September 15-18, 1996, San Diego. June 1994.

[Kar94] Yleinen suomalainen asiasanasto (General Finnish Thesaurus). Eeva Kärki, editor. Kirjastopalvelu Oy, Helsinki. 1994.

[Lev93] Document reuse and document systems. David Levy. Electronic Publishing - Origination, Dissemination and Design 6(4):339-348. December 1993.

The Document Management (DocMan, http://www.cs.helsinki.fi/research/rati/) Research Group at the Department of Computer Science at the University of Helsinki studies the theory and applications of structured documents. In a research project called Structured and Intelligent Documents (SID), the group studies and develops methods for attaching intelligent features to structured documents. The group includes the following researchers: Prof. Heikki Mannila, project leader Dr. Pekka Kilpeläinen, Ph.L. Helena Ahonen, M.Sc. Barbara Heikkinen, M.Sc. Oskari Heinonen, B.Sc. Jani Jaakkola, and Ph.L. Greger Lindén.