University of Helsinki
Department of Computer Science
Series of Publications C, No. C-1996-40

Intelligent Assembly of Structured Documents

Helena Ahonen, Barbara Heikkinen, Oskari Heinonen, Jani Jaakkola, Pekka Kilpeläinen, Greger Lindén, and Heikki Mannila

Helsinki, June 1996

Report C–1996–40

University of Helsinki
Department of Computer Science
P.O.Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, FINLAND

Intelligent Assembly of Structured Documents

Helena Ahonen, Barbara Heikkinen, Oskari Heinonen, Jani Jaakkola, Pekka Kilpeläinen, Greger Lindén, Heikki Mannila

Department of Computer Science, University of Helsinki
Report C-1996-40
June 1996
12 pages

Abstract

An intelligent document contains information about its structure, its contents and its environment. This information supports intelligent document assembly, i.e., the effective reuse of existing documentation to produce new documents adapted to specific needs. 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 assembled document from fragments of documents. We report on three case studies, where intelligent document assembly methods are being prototyped on commercially used document material.

CR Categories and Subject Descriptors:

I.7.2 [Text processing]: Document Preparation.
J.7 [Computers in other systems]: Publishing.

General Terms: Design

Keywords: document assembly, structured documents, SGML

Contents

1 Introduction **1**

2 Adding intelligence to documents **2**

2.1 Inherent information 2

2.2 Supplementary information 3

2.3 Placement of supplementary information 4

3 Automated document assembly **4**

4 Case Studies **5**

4.1 Intelligent law collections 5

4.2 Intelligent textbooks 6

4.3 Intelligent newspapers 8

4.4 Transformation of structured documents 9

5 Conclusion **10**

References **10**

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 of 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. An *individualised* or *tailored* document is a specialised document that satisfies the needs of a certain user group. 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 *intelligent document* contains information about itself and its environment. It is no longer a passive linear version of a text, but may contain references to other documents and information sources. The document knows how to assemble subparts of itself and of other documents to form a tailored new document. *Active documents* [Spi88, TB90, ET94, QV94, HHR94] are one form of intelligent documents. An active document includes *activities* that are performed as a result of the user opening, scrolling, editing, or closing a document. Even if the activities are triggered by the document editor, they are associated with the document and preserved when the documents are filed, copied, or even electronically mailed.

Documents can be considered as user interfaces [BG90][1]. World Wide Web (WWW) servers with associated Common Gateway Interface (CGI) programs offer very popular active-document interfaces. However, CGI programs tend to form incomplete, unfinished, list-like documents as a result of their activities. The result is ready at hand, but seldom suitable for commercial publishing or even printing. Active documents often use proprietary formats for either activities or underlying knowledge. In contrast to this, the Java[2] language is a new tool for the development of device independent code that WWW clients interpret and execute by using the corresponding document as a user interface.

In an on-going research and development project called 'Structured and Intelligent Documents

[1] Complementarily, Fuchs has suggested *implementing* portable user interfaces *as* SGML documents [Fuc96].
[2] Java is a trademark of Sun Microsystems, Inc.

(SID)'[3] 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 with SGML [ISO86].

The rest of this paper is organized as follows. In the next section we define the inherent and supplementary information needed to make a document intelligent. We also discuss the representation and the placement of the information. In Section 3 we analyze the assembly process, which we have divided into the phases of assembly query, assembly, and merging of subresults. In Section 4, we present some examples of applications that use intelligent document assembly. Finally, we give a short conclusion of our work.

2 Adding intelligence to documents

An intelligent document needs to use both information based on the contents, and information about the application and the world. Adding intelligence is called *seeding* [ET94]. Seeding can be performed both automatically or manually. Manual seeding gives the author better possibilities to fine-tune the supplementary information. However, for a large documentation we also need automatic seeding methods. In the following, we distinguish between inherent information and supplementary information.

2.1 Inherent information

Some of the information required for an intelligent document is available in the document itself. We call this information *inherent* if it is present in the document or automatically derivable from the document. Such information is based on the contents of the document, e.g., words or phrases, on the structure of the document, on a combination of both, or it may capture more higher-level semantics of the document.

The simplest way to utilize inherent information is *full-text search*. We may determine that a document fragment is appropriate for our assembly if it contains certain words or phrases. *Indexing*, on the other hand, tends to discard common words like *the* and *is*, and only include relevant terms and phrases. In many languages, like in English, indexing is easy as words take very few inflected forms. Only in few cases (e.g., *ate* from *eat*) the base form might not be automatically recognizable from the inflected form. Such cases can, however, be processed separately because they are few.

In morphologically rich languages, like in Finnish, a word can take many, even thousands of

[3]The project is funded by the Finnish Technology Development Center and seven Finnish enterprises (The Aamulehti Group, Oy Edita Ab, the National Board of Education, WSOY, Helsinki Media, Lingsoft, Inc., and MTV3).

different inflected forms [Kar83]. The base forms are often not recognizable from the inflected form. For example, we have the word *poika* ('boy', 'son') which takes for example the inflected forms *pojan*, *poikaa* and *pojille*. In such cases a morphological analysis is needed to collate the indexing terms. Unfortunately current natural language processing systems are, in general, not capable of choosing the correct meaning for homographs, i.e., words written in the same way but with different senses.

Another way to use or compute information directly from the document is based on the document structure. Ideally, we describe the logical structure of a document with SGML. In addition to marking sections and paragraphs (which are often recognizable even in plain text files), we can define much richer document structures. The document structure on its own may not tell us much, but combined with the contents we can use a very powerful search approach to find relevant document fragments. On a simple level, we can ask for all documents that were written by a certain author (assuming that the author's name has been properly marked up). On a more complicated level, we can index only those words that have been marked as keywords in the document, or cluster the documents based only on their abstracts.

The most difficult inherent information to use and to deduce is the semantics of the documents. We would like our assembly mechanism to understand our documents. For example, the assembly mechanism should know when the document talks about movie stars and not about stars in the sky. It should be able to understand phrases like 'The pen is in the box' versus 'The box is in the pen' as has been suggested by Lenat [Len95]. However, it still remains an open question of how much semantic understanding we are able to include in the assembly.

2.2 Supplementary information

In addition to the inherent information of a document, we may add *supplementary information* associated with the document. Such supplementary information are, e.g., *references* to other documents. Typical examples are references stored in a separate document. A reference in a document might also indicate that another document fragment should be included in the result of the assembly. An example is the inclusion of addresses in marketing letters from an address database.

Other supplementary information is based on *common sense knowledge*, e.g., in a separate knowledge base [GL94, Len95]. Common sense tells about the world and the application area. It could include information about geography saying that Finland is one of the Nordic countries, which are a part of Europe. If we demand documents about the Nordic countries or Europe, our assembly mechanism will then provide documents about Finland but not about non-European countries. Common sense knowledge also appears in *thesauri*, containing, e.g., synonyms, hypernyms (supernames), or hyponyms (subnames)[4] of a certain fact. A thesaurus can also handle application areas telling about specialized terminology.

Supplementary information can be described in *dependencies*, *conceptual hierarchies*, or other *descriptions of the application area*. We could describe how different statutes depend on each

[4] ... and meronyms (part-name) and holonyms (whole-name) and troponyms (manner-name) and entailments ... [Mil95].

other in a collection of laws, or the structure of a product described by its documentation.

There are several other sorts of supplementary information that could be added. *Problem descriptions* and *solution models* automatically control the execution of a process. *Environmental information* shows what the user has done, what choices have been made, and what states have been reached. Even *style sheets* describing different formatted appearances for the documents can be considered supplementary information pertaining to the representation of the assembled documents for different media.

User profiles are handy to include as supplementary information. The profile can contain information about experience, access rights, roles, etc. It can also be based on the user's previous behaviour in similar situations, or on the monitoring of expert users. Novice users are offered more help and simpler documents than expert users, and in an educational environment parts of the documents may be chosen based on the previous actions of the reader.

2.3 Placement of supplementary information

Both inherent and supplementary information can be placed either in the SGML document (*internal information*), or outside in files, databases or the like (*external information*).

Information can internally be placed in the SGML markup, in SGML attributes or in the SGML document type description (DTD). For example, we can choose to explicitly mark all the paragraphs of a document with tags telling the degree of difficulty a supposed reader will have in understanding the text. Just as well we can include an attribute in already existing markup expressing the same thing. In the DTD we could mark by default all first paragraphs of new sections as 'introductory' and all remaining as 'advanced'.

External information can be placed, as we have seen above, in databases or rule collections. It can also be placed in the query which will assemble the document. We note that references to documents and other data sources do not have to be placed within the intelligent document. The HyTime standard [ISO92] presents a way to store references outside the document as 'ilinks'.

3 Automated document assembly

Document assembly consists of three phases. First, 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. Assembly conditions are based on the internal and external information listed above. One approach 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.

The second phase performs the assembly. Appropriate document fragments are found and returned. A document fragment is returned if it either fulfills 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 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.

The assembly is performed by different *software agents* [CAC94] working on the document and its supplementary information. A general agent will assist 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. Just as in the case of the supplementary information, agents can be modified or combined. Implementations can be made using script languages, or, e.g., in the new Java language [GM95] which provides a safe device-independent way to express portable actions. Agents can also be bound to a certain environment, especially for in-house needs where the locally available programming languages and platforms are sufficient.

4 Case Studies

The techniques and methods mentioned in the previous sections are 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 and 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.

4.1 Intelligent law collections

In this case study 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 intelligent assembly, because of its large number of independent, yet strongly related units of textual information. We are developing an assembly environment that allows the utilizer 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[5]. We add *inherent information* to the documents by computing keyword lists for the subdocuments. The keyword lists are computed with the assistance of

[5] For a discussion on subdocument granularity in information retrieval, see [VAB95].

the FINTWOL[6] 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. classification, is presented to the user in a form. Thus, 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 [Kär94] is used as a search-aid thesaurus. That is, the terms given by the user are expanded by related terms listed in the thesaurus. This, naturally, broadens the selection of initial documents.

4.2 Intelligent textbooks

The second case study develops intelligent educational material for use at technical schools. The overall goal is to create a material pool that contains textbooks, articles, exercise collections, etc. The pool can be used in multiple ways. The publisher can print customized textbooks for various branches and levels. Additionally, teachers can form tailored compositions of CD-ROM delivered material — e.g., summaries, slides and 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[7], contains many equations and technical pictures. Creating a structured SGML version required laborious conversion steps using tools such as MathType[8], DynaTag[9] OmniMark[10], and some graphics programs. The basic structure for the book is the BOOK DTD of 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 infor-

[6] FINTWOL is a product of Lingsoft, Inc.
[7] Word for Windows is a trademark of Microsoft, Inc.
[8] MathType is a trademark of Design Science, Inc.
[9] DynaTag is a trademark of Electronic Book Technologies, Inc.
[10] OmniMark is a trademark of Exoterica Corporation.

```
<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.</P>
     <P><DFORMULA id="c4df13" name="c4d13" notation="tex">
         <DFBODY>$$\bar Z=\bar Z_R+\barZ_C$$ </DFBODY>
       </DFORMULA></P>
      ...
```

Figure 1: An example of internal supplementary information.

mation 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. Figure 1 is an example of the supplementary information (in boldface) attached to the markup of the material. 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 'dformula'). 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 such as DynaText[11] or Panorama[12] support this kind of querying.

Individualised documents are assembled based on their contents, structure, attributes and key-

[11] DynaText is a trademark of Electronic Book Technologies, Inc.
[12] Panorama is a trademark of SoftQuad, Inc.

words. In order to achieve sensible results, the assembly has to utilize heuristics which guarantee the inclusion of all necessary fragments. We plan to offer a simple query language for the users, although it may be necessary to offer some pre-selected assemblies as well. The user will be able to modify assembled views by including or removing fragments.

4.3 Intelligent newspapers

In the third case study, we have designed and implemented a prototype environment for intelligent assembly of newspaper articles. The articles chosen for the prototype are annual accounts reports of enterprises. The articles were originally created using the QuarkXPress[13] 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 2 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 tables. 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.

[13] QuarkXPress is a trademark of Quark, Inc.

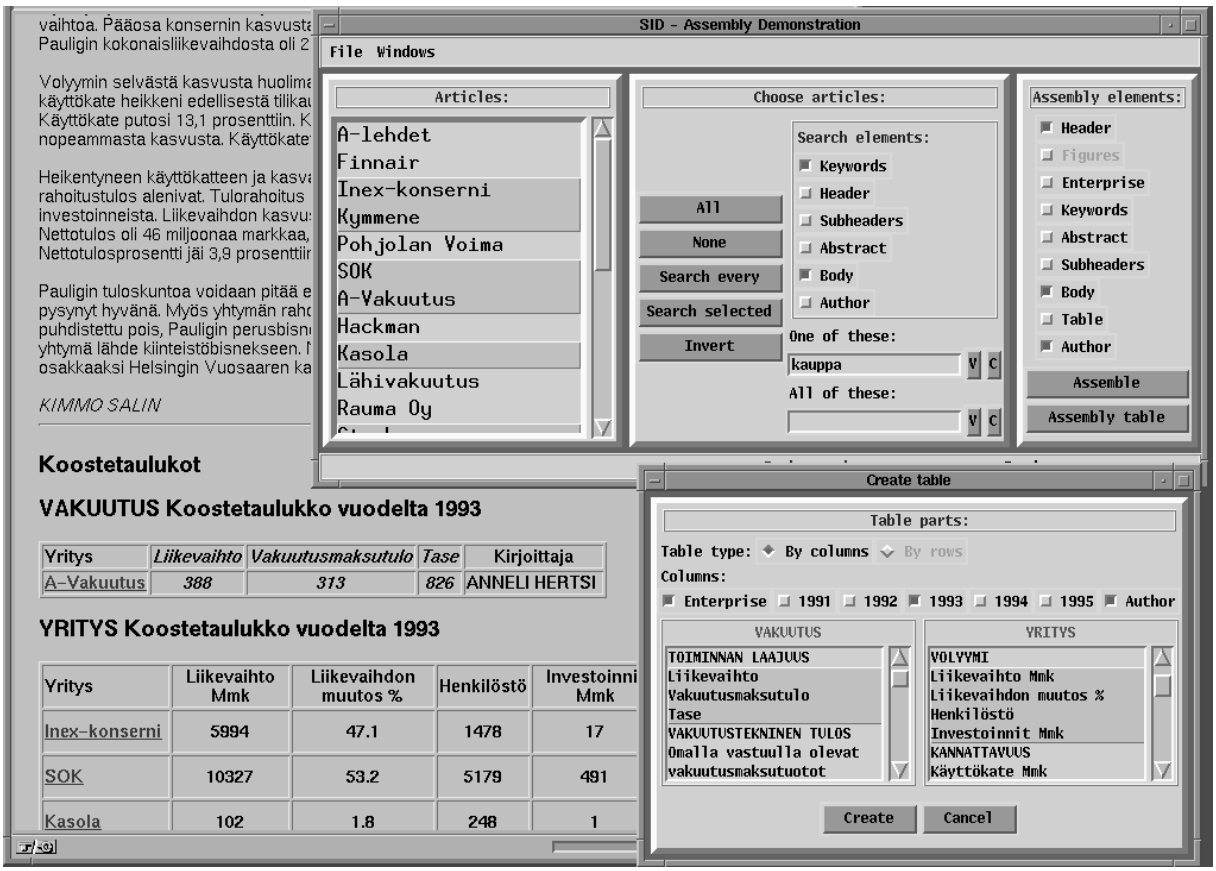

Figure 2: The user interface of the assembly tool.

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[14] scripts, which take the assembly parameters from the graphical user interface. For visualization purposes the assembled articles are converted to HTML, which makes them instantly ready for on-line publishing on the Web.

4.4 Transformation of structured documents

In all the cases mentioned above, there is a need for the transformation of structured document fragments. Sometimes fragments must be transformed to conform with a certain DTD; sometimes the assembly itself requires that the retrieved document is pruned of certain parts, augmented with others, or that components pulled out from different parts are combined together. In order to satisfy these needs, we have designed a new declarative, simple and powerful transformation language for SGML documents, called TranSID.

The TranSID language gives the user access to the entire SGML document tree, not only to a

[14] MetaMorphosis is a product of MID/Information Logistics Group GmbH.

stream of start and end events of structural components, which is common in most other SGML transformation languages. The use of the language is based on describing the substructures of the SGML instance that are to be replaced by other substructures. The transformation process corresponds to the tree transformation process of the forthcoming DSSSL [ISO96] standard, but the transformation description is given on a higher level of abstraction.

5 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 are under way to test and to develop the principles of intelligent document assembly further.

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 can 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.

References

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

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

[BG90] Eric A. Bier and Aaron Goodisman. Documents as user interfaces. In *Proceedings of the International Conference on Electronic Publishing, Document Manipulation & Typography*, pages 249–262, September 1990.

[CAC94] Special issue on intelligent agents. *Communications of the ACM*, 37(7), July 1994.

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

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

[ET94] Paul M. English and Raman Tenneti. Interleaf active documents. *Electronic Publishing – Origination, Dissemination and Design*, 7(2):75–87, June 1994.

[Fuc96] Matthew Fuchs. The user interface as document: SGML and distributed applications. *Computer Standards & Interfaces*, 18(1):79–92, January 1996.

[GL94] R. V. Guha and Douglas B. Lenat. Enabling agents to work together. *Communications of the ACM*, 37(7):127–142, July 1994.

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

[HHR94] Anja Haake, Christoph Hüser, and Klaus Reichenberger. The individualized electronic newspaper: an example of an active publication. *Electronic Publishing – Origination, Dissemination and Design*, 7(2):89–111, June 1994.

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

[ISO92] ISO and IEC. *Information technology – Hypermedia – Time-based Structuring Language (HyTime), ISO/IEC DIS 10744*, 1992.

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

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

[JK96] Jani Jaakkola and Pekka Kilpeläinen. Sgrep – a tool to search structured text. In preparation, 1996.

[Kar83] Fred Karlsson. *Finnish grammar*. WSOY, Helsinki, 1983.

[Kär94] Eeva Kärki, editor. *Yleinen suomalainen asiasanasto (General Finnish Thesaurus)*. Kirjastopalvelu Oy, Helsinki, 1994.

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

[Len95] Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, November 1995.

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

[Mil95] George A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, November 1995.

[QV94] Vincent Quint and Irène Vatton. Making structured documents active. *Electronic Publishing – Origination, Dissemination and Design*, 7(2):55–74, June 1994.

[Spi88] Robert Spinrad. Dynamic documents. *Harvard University Information Technology Quarterly*, VII(1):15–18, Spring 1988.

[TB90] Douglas B. Terry and Donald G. Baker. Active Tioga documents: an exploration of two paradigms. *Electronic Publishing – Origination, Dissemination and Design*, 3(2):105–122, May 1990.

[VAB95] Marc Volz, Karl Aberer, and Klemens Böhm. A flexible approach to combine IR semantics and database technology and its application to structured document handling. GMD Technical Report No. 891, German National Research Center for Computer Science, January 1995.