Proceedings of DETC'04
ASME 2004 Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
September 28-October 2, 2004, Salt Lake City, Utah, USA

DETC2004-57677

AN APPROACH TO EXTRACTING KNOWLEDGE FROM LEGACY DOCUMENTS

Richard Crowder and Yee-Wie Sim
Intelligence, Agents, Multimedia Group
School of Electronics and Computer Science
University of Southampton
Southampton
{rmc,yws00r}@ecs.soton.ac.uk

ABSTRACT

Organisations are increasingly information intensive; hence providing access to data that is trapped in various proprietary forms including catalogues, databases, human resource systems and internally generated documents is now becoming a significant and challenging task. The authors have undertaken research into approaches to capture relevant knowledge from legacy documents. This is achieved by converting the legacy documents to XML, (eXtensible Markup Language), documents where the output is semantically tagged. Once in an XML form, the data can be easily transformed. This paper describes the development of tools to automate the process of converting legacy documents to XML documents. The purpose of this work is improve the efficiency and reliability of Expertise Finder suitable for use within an engineering design environment. We will also show that by querying the resultant XML versions of legacy documents provides better results than a basic text search over the identical documents when applied used within an Expertise Finder.

Keyword: Knowledge management, Expertise finders, XML.

INTRODUCTION

Organisations have been sharing information and knowledge for many years. But only recently, organisational interest in knowledge management has increased dramatically, [1]. The collaboration space now is more virtual then physical. As global reach of business expands and companies become larger and more geographically dispersed, it becomes increasingly difficult for them to know where their best knowledge is and even more difficult for them to *know who knows what*. The goal of knowledge management is to develop and use computer technology to improve the performance of companies. More specifically, organisations aim to acquire knowledge from valued individuals and to analyse users' activities to learn from successes and failures.

To solve many problems people need to have access to specific documentation. In many cases this expertise can be captured from a individual and used to populate a specific database which works very well if the problem is restricted to a very specific domain, for example robot maintenance, [2] or to key personnel (managers, senior employees, information concierges), [3], facilitating the contacts. With the advent of knowledge management systems it is possible to automate the process through Recommender Systems or Expertise Finders (EF). A recommender system that suggests people who have some expertise with a problem holds the promise to provide, in a small way, a service similar to these key personnel. Expertise recommender systems can also reduce the load on people in these roles and provide alternative recommendations when these people are unavailable. A number of these systems have been reported in the literature, [4–8].

Valuable data trapped in various propriety forms can be mined for use to construct profile in expertise finding system. Organisational environments are increasingly information intensive, hence providing access to data that is trapped in various proprietary forms including catalogues, databases, human resource systems and documents is a challenging task.Our approach to EF contrasts with a number of reported systems where web based information is used to provide the recommendation, [9–11]. Answer Garden 2, [12], has an explicit expertise-location engine and provided computer-mediated communications mechanisms to find others with a range of expertise, though the mechanisms were not very elaborate. A different approach was taken by McDonnald, [3] who used software developed by employees to identify their expertise in various aspects of software development. In the recommendations provided by the EF, trust is important, this can be achieved by showing why people were not recommended or why a document was not considered so important. A document might seem relevant based on a full text search but is actually twenty years old, an important factor in some situations, but not in others. As discussed previously the provision of evidence for an EF decisions in the form of a list of documents and other data is considered a key EF output, [13].

This papers details an approach to capture knowledge from legacy documents. This was achieved by converting legacy documents to equivalent XML (eXtensible Markup Language) documents. Once converted to XML, the resultant documents can be used for a number of applications including being searched as part of an Expertise Finder.

CONVERSION RATIONALE

Converting a document to XML requires parsing followed by tagging with suitable descriptors. For example within a Microsoft Word document, text and hyperlinks are already tagged by their formatting. Most documents contain multiple structural elements, such as headings, footnotes, and quotations. A wide range of formatting can be applied to indicate what the elements mean. For example, most headings are not the same size, weight, or even font as paragraph level text. Within a document, you alter text by one of two methods: by applying a style or by applying formatting manually. A style in documents is nothing more than a named set of specific instructions describing the formatting to apply. When a style is applied to text, that text is basically being tagged as something: for example a heading, a subheading, or some other document element. Very often when formatting is applied manually, that text is being tagged as “something special”, but that “something” is normally not defined.

If the document is parsed by formatting, information about the appearance of text in that document can be found, but what the text’s style means is not known. However, if formatting is applied using styles, when the document is parsed, not only how the text appears in the document is known, but the style names applied also carried information for describing what the text is. Creating a document in this manner requires that formatting representation to be defined. Instead of making text bold for emphasis, a style is applied that not only bolds the text but is descriptive of why the text is bold to begin with. For example, a document is created and includes a quotation. Rather than applying italic formatting to the quotation to highlight it, a style called *quotation* that includes italicized formatting should be defined.

Querying

If documents are authored using styles and then converted into XML documents, it becomes a queryable data source. A folder of XML documents are essentially a database. Once in XML form, the data can be easily transformed into more sophisticated controls or behaviours. The records become easier to manipulate using stabdard programming language such as Visual Basic, Java or C++.

Consider the following example; a software development company creates technical documents using Microsoft Word. One day, the company decides to find out how many of their products are web-enabled. An indexing program could be employed to perform the required task, but it need to parse the Word binary format. Then what should it look for? Would the mere presence of the words ”Internet” or ”Web” indicates which parts of the products had Internet capability? Now suppose the documents were stored as XML. Elements describing each part of a document can be selected by using XML Document Object Model (DOM) scripts or style sheet linking techniques, hence providing more robust searching capability. Also the descriptive tags labelling each part of the documents allow them to be displayed in variety of ways, i.e. using sort or filter techniques.

Reusing Documents

It is widely recognised that the best document is one that can be used multiple times in multiple ways. As XML is a markup language written in plain text, it is easy to exchange between instances. Using text means the task for decoding complex, propriety data formats can be avoided. Imagine the overhead that would be incurred in trying to convert a binary Word document into another format. Working with text makes it easy to transform one XML structure to another. Different documents can be created from the original document saved in XML. One possible technique is data binding using scripting language, like JavaScript or Perl, allowing pieces of data in the XML documents linked to a display document. Possible uses for this approach are to create synopses of articles or a code library from developer articles, or retrieve all of the references in an article set. This approach has been demonstrated in a reconfigurable equipment manual, [14].

Reducing Workload

In many applications knowledge needs be captured directly from work process without extra effort by the users. Asking the users to author documents in both word processing format and

XML format is too much of a burden to them. The conversion solution discussed in this paper requires no programming skills to apply in an organisation, and minimal training in XML processes is required.

XML CONVERSION TOOLS

In this section we reviewed four approaches to XML conversion, in all cases we considered the starting point to be a conventional *Word* document:

1. Standalone Word to XML Converters
2. Integrated Word to XML Editors and Converters
3. Word 2000 / 2002 as XML Converter
4. Custom Built Converter

Standalone Word to XML Converters

Tools, which take the strategy of converting Microsoft Word into XML, provide a standalone conversion application into which one or more Word documents are entered, and one or more XML documents emerge. Very often these tools convert Rich Text Format (RTF) versions of a document, rather than the native Word format. This is useful if documents have been created by a number of different word-processing tools, but an extra conversion step if not.

Many of the standalone converters define a fixed XML Document Type Definition (DTD) into which all input documents are converted. This usually means that a further processing step must be carried out to convert the raw XML output to comply with the DTD required a particular organisations requirements. However, because the input for this step is well-formed XML, XSLT (Extensible Stylesheet Language Transformation) can be used as a very efficient method for carrying out this processing.

Standalone converters are useful in a number of situations. If there is a large repository of legacy information to be converted, a batch-oriented approach is efficient. If you wish to protect authors from the details of XML, they can send their documents to a central administrator who can take responsibility for the conversion stage. Such converters are also usually either inexpensive or free. An examples of a standalone converters is *UpCast*, from Infinity Loop [15], this any similar tools originated from the *Rainbow* [16] converter for RTF to SGML conversion.

Two of the major drawbacks of standalone converters are that the conversion process is not integrated with the authoring environment, and can take a number of manual steps (conversion from Microsoft Word to RTF, RTF to raw XML, raw XML to desired XML). The conversion process is one-way only, so that the information is maintained in Word format only. If changes are made to the XML version, these must be re-keyed in the Word version if the document is being maintained over a long period. Finally, considerable programming skills are often required to configure an XSLT script to convert the raw XML from the converter into the final desired result.

Integrated Word to XML Editors and Converters

The second major strategy for turning Word into XML is to integrate the conversion process into the Word environment. With this approach, Word is converted into a structured syntax-directed authoring environment, similar to a structured XML editor like SoftQuad XmetaL [17]. Usually this is achieved by creating a mapping between named styles in Word and elements in the XML target DTD. Authors are constrained in two ways. Firstly they can use only the styles named in the document, and may not add their own new styles. Secondly, they may only be able to use particular styles in particular locations in the document, for example a ‘Country’ style is allowed only after a ‘City’ style in an address. Usually, tools taking this approach not only convert Word into XML, but also read XML files into Word for editing.

This approach is useful when full two-way conversion between Word and XML is required. Edits can be made either in an XML editor or in Word, so if the information is subject to ongoing change, there is no requirement to maintain separate Word and XML versions. When a DTD is very complex, the ability to provide direction to authors as to what mark-up is allowed at a particular point is also useful. Examples of integrated Word XML editing environments are S4/Text from *i4i* [18] and *Worx SE* from HyperVision [19].

The major drawbacks of integrated Word XML editors are that the software is expensive to buy, and complex to configure for particular DTD’s. This type of tool also makes major modifications to the default Word authoring environment, which can be disorienting for authors, and may have a significant impact on performance. Usually the underlying rationale behind using Word as the XML editing tool is to offer authors a familiar environment. Changing that environment to a significant degree begs the question of why bother with Word at all? Using a full What You See Is What You Get (WYSIWYG) XML editor, such as XMetaL, would be a more suitable choice compares to Word XML editor. But the users need training in using the new authoring tools. This will cost organisations time and money.

Word 2000 / 2002 as XML Converter

Word processing software, such as Microsoft Word, typically saves documents as binary data. While other editing applications may be able to convert the file, it requires sophisticated code and detailed knowledge of the format. Often,the conversion is flawed, and some degree of formatting is lost.

Some progress was made when word processors were able to save documents as HTML. This increased portability, but the limitations of HTML resulted in a loss of control over presentation details, and certain standard document features, such as embedded comments, do not have a natural counterpart in HTML.

Microsoft, with the release of the Office 2000 suite, made significant enhancements to how documents were saved as HTML by the use of CSS (Cascading Style Sheets) and embedded islands of XML data. The result is that a Word document saved as HTML will appear almost exactly in a browser as it does in Word. Though it should be noted that the enhanced HTML is only understood by Internet Explorer 4 or above.

If you look at the generated HTML, it appears to be XML, which would certainly be a step towards application independence. The content is readable in any text editor, and conversion to another format could be performed by XSLT. However, although it looks like XML, there are flaws. When Word creates the HTML, it often inserts incorrectly nested tags, and fails to quote all attributes.

Another problem with the HTML is the immense amount of formatting information included in the document. A simple document one line in length can produce an HTML file of over 200 lines, because of the inclusion of style definitions that are not even used. There are tools available, such as *tidy* [20], that will go through an HTML document and attempt to convert it to XML. However, such programs will need to make assumptions about what is the best way to correct formatting problems, and the result may not always be what is wanted. For example, if an element is missing a closing tag, the program may not realize this until the end of the document, and decide to append it there.

Custom Built Converter

The approaches described above have their advantages and drawbacks. Another category of solution exists for converting information in Word format into XML, is by using custom scripts written in WordBasic, Visual Basic for Application, Omnimark or Perl.

Custom scripts approach was chosen for the conversion process because the application design is for specific niche needs instead for general-purpose intention. VBA is the appropriate candidate for the job. It is a high-level programming language that underlies several important Microsoft Windows applications, such as Microsoft Office including Microsoft Word, Excel, Access and Power Point. Microsoft has done a considerable job in developing the suite of Office applications as scriptable objects.

Coding of the conversion tool will rely heavily on the usage of the extensive properties and methods exposed by the Word Document Object Model. The key technique is to use Word Styles (user-defined formatting properties) as markers for document structures (paragraphs, headings, font weights etc.). Then using scripts to generate markup based on the styling information.

Using the conversion tool, it is possible for users to create feature-rich XML content with Word. The tool allows authors to extract information from Word format and then converted into XML. Organisations can realise the benefits of XML with no disruption to users because they can continue to use the tools they already familiar, i.e. Microsoft Word. Finally the conversion tool will save organisations time and money associated with training in using new authoring tools.

However there are no magic bullets that will convert all Word documents to semantically tagged XML. Although the specialised tools in the survey are capable of converting Word documents to XML, but the output from even the best of these tools often needs to be cleaned up by hand. How much clean-up work you need to do generally depends on how structured the Word document is to start with. Problems such as, missing fields or a field that is supposed to contain numerical value has non-numerical value in it, has to be accounted. Hence the author has chosen the Custom Built Converter solution to develop conversion tools.

DEVELOPED CONVERTER

The authors developed a conversion tool for transforming Microsoft Word documents into XML. XML is an ideal format for data storage because it is data centric. It allows separation of structure, content and presentation and it is designed specifically to work on the interactive and tool-rich Web environment. XML can be accessed using programming language, such as Visual Basic, Java and C++, making it easy for manipulation and display. When used to described database, XML has two advantages over propriety format (such as Microsoft Access); XML is human readable, and it is based on a public, open standard.

Overview of approach

The goal of the conversion solution is to convert a Word document into a well-formed XML document. The most logical way to do this is to tag data in a Word document by using styles. Formatting using style in documents give information about the meaning of the styled text. An overly simplified description of how this solution works is that it parses through the document paragraph by paragraph and identifies text with styles applied, and then tags that text.

Microsoft Word exposes an object model that can be scripted using Visual Basic. Extensive properties and methods available in the Word object model form the basis for developing the conversion tool. Two objects named Document object and Range object play a major role in the coding. They are called frequently to execute commands such as open Word document or find text.

The functions developed can be employed to convert most well structured documents, i.e. one that make use of styles in authoring document. The *GetField* and *GetNextField* functions are the core of the conversion code. Suppose a document with structure shown in Figure 1 where the field content is followed closely after the field title in the same table cell. *GetField* function is used to extract the field content with the field title supplied as the search text. Field title is first selected and then the starting point of selection is moved to the start of field content, and the end point of selection is relocated to the end of table cell or beginning of next field title. The text selected will be store at the end of execution of function.

If the document with a different structure as shown in Figure 2, where the field title is in one table cell and field content is in the next following table cell. The *GetNextField* method is employed to locate the field title and then the next table cell will be selected, which is the field content. Then the selected field content will be stored.

The intricacies and limitation of the Word object model made coding a very challenging exercise. A major area of concern was associated with graphics in Word documents. Graphics in Word documents are contained in two types of objects, namely *Shape* and *InlineShape* objects. Also each Word document has two layers, the text layer and drawing layer. A Shape object is an object that is placed in the drawing, i.e. floating over text of a document. On the other hand, an *InlineShape* object is an object that is placed on the text layer. It is therefore treated like a character, in the sense it moves with the surrounding text. When converting a Shape object to text describing the path where it is stored, anchor information of the object is lost. The Shape objects may not appear in the right places in the XML output. Also the method for saving embedded *InlineShape* or *Shape* objects is not available in the Word object model. The author solved the problem by using *PictureBox Control* and *Clipboard Object* in Visual Basic. Each *InlineShape* or *Shape* object is cycled through and placed them in PictureBox Control. This solution enabled the embedded graphics in the Word documents to be saved at specific paths in either bitmaps or JPEG formats.

Although certain features in Word document are harder to convert than others, but with some restriction most documents can be transformed into XML.

CONVERTER APPLICATIONS

The converter was developed to support two research project being undertaken at University of Southampton, namely *Electrical and Electronic Engineering Assessment Network*, e3an, [21] and *Knowledge Capture, Sharing and Reuse in the Design Process*, KCSR, [22]. In both application legacy documents were converted to XML for either presentations reasons or for subsequent analysis:

e3an This project developed a very large database to support the teaching of Electrical and Electronic Engineering to students at UK Universities. Twenty authors prepared over 3000 questions using a predefined template. The completed questions were then processed by the converter to allow the questions to be distributed to the Question and Test Interoperability (QTI) Specification as used by a number of computer aided assessment packages. With this large user group, and the volume of documentation it was clear that our solution needs to be robust – this was achieved. During the project the converter proved to be highly reliable, with most problems being associated with early versions of Word or Windows.

KCSR The Knowledge Capture, Sharing and Reuse in the Design Process (KCSR)project involved the Engineering Design Centre at the University of Cambridge, the Institute of Work Psychology at the University of Sheffield and the Intelligence Agent Multimedia Group at the University of Southampton, together with two major industrial organisations. The stated objectives of KCSR were to develop an understanding of the human aspects of sharing and reusing knowledge through an integrated theory of knowledge structure. The ideas would be demonstrated through a prototype of appropriate technical system. One key feature of KCSR was taking a sociotechnical approach to knowledge management in the context of the engineering design environment. One key objective was the development of an expertise finder, the initial work associated these developments has been reported in the literature, [13, 23]. As part of this work the conversion techniques have been used to analyse and convert 300000 documents into a XML database that then can be mined. Our initial assessment is that this gives a more accurate method of locating experts over the more conventional approach.

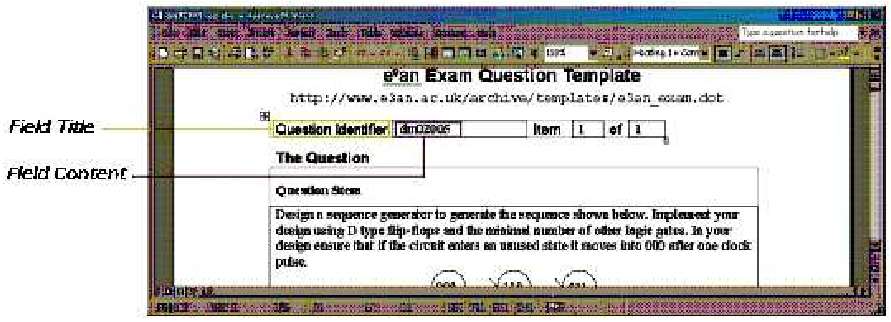
CONCLUDING COMMENT

Expert finding systems uses various evidences as indicators of expertise. In general, these evidences can be grouped as explicit and implicit evidence. Example of explicit evidences is expertise data entered manually in expertise database. But focus of expert finders is on employing implicit evidences since they do not impose burdens on the experts. Some of the implicit evidence considered as indicators of expertise are document authorship, name occurrence in non-authored documents, use of information sources, queries sent to an information retrieval system and the departments or projects experts work in.

The work discussed in this paper focus on capturing implicit evidences from legacy documents. Documents in organisation are potential sources of expertise evidence since they are created by or about somebody. In order to adequately exploit the information space as a source of expertise indicators, expert finders need to handle the diversity of the information space.

One aspect that needs to be addressed is the diversity of sources in reflecting expertise. How well expertise indicators like terms and phrases reflect expertise is mainly a factor of how the source in which these indicators occur relates to the expert.

Expert finder systems [3–8, 24] reviewed by the author employ raw documents indexes technique to capture documents' concept. But since this technique only capture concept-to-



e3an Exam Question Template

http://www.e3an.ac.uk/archive/templates/e3an_exam.dot

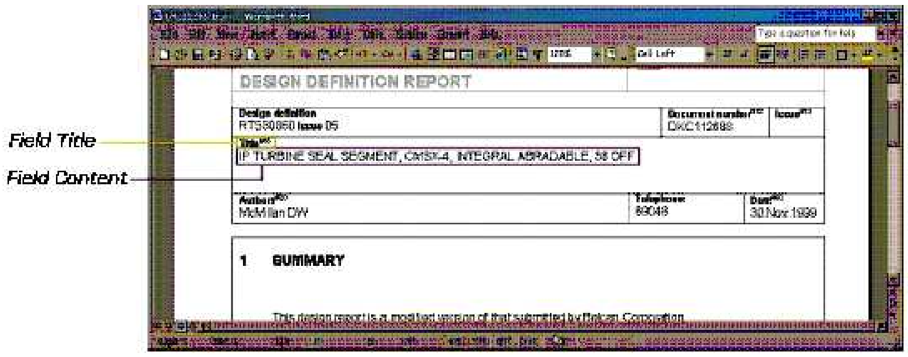
Question Identifier	dm02005	Item	1	of	1

The Question

Question Stem

Design a sequence generator to generate the sequence shown below. Implement your design using D type flip-flops and the minimal number of other logic gates. In your design ensure that if the circuit enters an unused state it moves into 000 after one clock pulse.

Figure 1. Document with field title and field content in the same cell.



DESIGN DEFINITION REPORT

Design definition	Document number	Issue
RTS90850 Issue 05	DKC112688	

Title
IP TURBINE SEAL SEGMENT, CMSX-4, INTEGRAL ABRADABLE, 58 OFF

Authors	Telephone	Date
McMillan DW	69048	30 Nov 1999

1 SUMMARY

This design report is a modified version of that submitted by Rolcan Corporation

Figure 2. Document with field title and field content in different cell.

document relationship (all concepts are associated with the document that contains them), a different approach is required. This is because in expert finder systems, the overriding aim is to find concept-to-expert relationship.

The concern with the documents focuses more on how they relate to expert rather than how they relate to the concepts they contain. Therefore, expert finding applications entail their own requirement of interpreting concept-to-document relation beyond the container contained one. Terms found in different types of documents indicate expertise differently irrespective of their statistical traits. For example, the occurrence of the term 'adaptive hypermedia' in Dr X's resume and its occurrence in one of his publications may not weight the same. Moreover, the occurrence of this term in the title of the document shows different distance to his actual expertise, compared to its occurrence anywhere in the body. Therefore, the relationship of expert-to-document needed to be determined before extracting terms from the document.

Organisations can benefits from author's XML conversion solution since no disruption to users while carrying out their daily routine task. This is because they can use the tools they already familiar, i.e. Microsoft Word. Also the tool will save the organisations time and money associated with training in using new authoring tools.

Use of knowledge management in industrial can be beneficial for developing a system that is both easy to use and completely comprehensive in manufacturing environment. Some of the most difficult problems encountered in implementing the system relate to the possible inertia of the management culture within the organisation. To make real progress with knowledge management requires changing work practices, mind-sets and reward structures.

Acknowledgements

The authors wish to thanks the School of Electronics and Computer Science, University of Southampton, for providing a research scholarship for Y W Sim. The Knowledge Capture, Sharing and Reuse in the Design Process (KCSR)project was funded by the UK Engineering and Physical Science Research Council under grant number GR/M83582/01. The Electrical and Electronic Engineering Assessment Network (e3an) was funded through the UK Higher Education Funding Council for England Fund for Development in Teaching and Learning (FDTL) Phase 3, as project 53/99.

REFERENCES

[1] R Smith and A Farquhar. The road ahead for knowledge management: An ai perspective. *AI Magazine*, 21(4):17–40, 2000.

[2] E Auriol, R M Crowder, R J McKendrick, R Rowe, and T Knudsen. Integrating case-based reasoning and hypermedia documentation: An application for the diagnosis of a welding robot at Odense steel shipyard. In *Proceeding of the International Conference on Case-Based Reasoning (ICCBR'99)*. 1999.

[3] D. McDonald and M. Ackerman. Just talk to me: a field study of expertise location. In *Proceedings of ACM 1998 Conference on Computer Supported Cooperative Work, CSCW 98*, pages 315–24. ACM, New York, 1998.

[4] D Mattox, M Maybury, and D Morey. Enterprise expert and knowledge discovery. In *Proceedings of the 8th International Conference on Human-Computer Interaction (HCI International'99)*, pages 303–7, Munich, Germany, 1999.

[5] A Kanfer, J Sweet, and A Schlosser. Humanizing the net: Social navigation with a 'know-who' email agent. In *Third Conference on Human Factors and The Web*, Denver, CO, 1997. http://www.optavia.com/hfweb/history.htm.

[6] H Kautz and B Selman. Agent amplified communication. In *Proceeding of Thirteen National Conference on Artificial Intelligence (AAAI-96)*, pages 3–9, Portland, OR, 1996.

[7] A. Vivacqua. Agents for expertise location. In *Proceedings of the AAAI Spring Symposium on Intelligent Agents in Cyberspace*, pages 9–13, Standford, CA, 1999.

[8] A Cohen, P Maglio, and R Barett. The expertise browser: How to leverage distributed organizational knowledge. In *The Workshop on Collaborative and Cooperative Information Seeking in Digital Information Environments at CSCW'98*, Seattle, WA, 1998.

[9] I Becerra-Fernandez. The role of artificial intelligence technologies in the implementation of people-finder knowledge management systems. In *Proceedings of the 2000 American Association for Artificial Intelligence Spring Workshop*. AAAI, March 2000.

[10] K. Bollacker, S. Lawrence, and C. Giles. Citeseer: an autonomous web agent for automatic retrieval and identification of interesting publications. In K. Sycara and M. Wooldridge, editors, *Proceedings of the Second International Conference on Autonomous Agents*, pages 116–23. ACM, 1998.

[11] P Chandrasekaran and A Joshi. An expertise recommender using web mining. In *Proceedings of AAAI 14th Annual International Florida Artificial Intelligence Research Symposium*. American Association for Artificial Intelligence, 2001.

[12] M. Ackerman and D. McDonald. Answer garden 2: Merging organisational memory with collaberative help. In *Proceedings of ACM 1998 Conference on Computer Supported Cooperative Work, CSCW 96*, pages 97–105. ACM, New York, 1996.

[13] G Hughes and R Crowder. Experiences in designing highly adaptable expertise finder systems. In *Proceedings of Design Engineering Technical Conferences and Computers and Information in Engineering*, Chicago, 2003. Proceedings on CDROM.

[14] R. Crowder, Y. Sim, G. Wills, and R. Greenough. A review of the benefits of using hypermedia manuals. In *Proceedings of the twelfth ACM conferences on Hypertext and Hypermedia*, pages 245–6, Arhus, Denmark, 2001.

[15] Upcast, 2003. Information avaiable at www.infinity-loop.de.

[16] Rainbow, 2003. see http://www.w3.org/Tools/Word_proc_filters.html.

[17] Xmetal, 2003. Information at http://www.sq.com/products/xmetal/.

[18] S4/text, 2003. Information at http://www.i4i.com.

[19] worx, 2003. Information at http://www.hvltd.com.

[20] tidy, 2003. Information at http://www.w3.org/People/Raggett/tidy/.

[21] e3an: Electrical and electronic engineering assessment network, 2003. Information at http:// www.e3an.co.uk.

[22] KCSR: Knowledge capture, storage and reuse, 2003. Information at http://www.iam.ecs.soton.ac.uk/projects/kcsr/.

[23] R Crowder, G Hughes, and W Hall. An agent based approach to finding expertise. In D Karagiannis and U Reimer, editors, *Proceedings Fourth International Conference on Practical Aspects of Knowledge Management*, pages 179–88. Vienna, Austria, December 2002.

[24] L Streeter and K Lochbaum. An expert/expert locating system based on automatic representation of semantic structure. In *Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications*, pages 345–9, San Diego, CA, 1988.