A Technical Architecture Proposal for the Knowledge Exchange Model for Scholarly Publishing

Introduction

This is a living document which proposes a technical architecture for the Knowledge Exchange Model for Scholarly Publishing. The term “living document” refers to the evolving nature of the document. The document will continue to change as the project moves forward. The purposes of the document include the following:

- To provide information on the current state and content of the technical architecture of the project.
- To communicate the information contained within, and to promote ongoing discussion of the content.
- To document the technical architecture process.

By way of background, the Public Knowledge Project is a federally-funded research project that is currently investigating the feasibility of a Knowledge Exchange Model (KEM) for scholarly publishing. We are examining the economic, technical and social feasibility of shifting publication directly to the research libraries, not only as a response to the oft-cited "crisis in scholarly publishing," but with an eye to improving the scholarly and public value of university research. With this model, the research libraries would devote their serials' budget to operating a distributed database system that supports the reviewing, editing, indexing, and archiving of research, while linking it to outside resources and making it globally and freely available through a common knowledge portal to scholars and the public. This model is described in detail at http://cie.ed.asu.edu/volume3/number6/ where it is accompanied by an ongoing discussion.

The first section of this document outlines the architectural principles that will guide the work. The following sections move from a very high overview of the document flow, through a high level functional specification of the system, and finally to a technical architecture. The portions of the document that precede the technical architecture are necessary in order to create a context for the technical architecture.

Any comments or feedback on the contents of this document should be addressed to fullerta@interchange.ubc.ca.

Architectural Principles

This section contains principles that will be applied in the definition of the technical architecture. That is not to say that these guidelines will be carved in stone, but rather that these principles will be kept in mind in decisions where there is latitude in design choices.

Wherever possible design choices should not limit subsequent options. For example, when interfacing a database with a web server a major design choice is whether to go with a proprietary interface or to use a generic Common Gateway Interface (CGI) written in a scripting language such as Perl or php. The proprietary interface will typically be faster and more efficient, but subsequent work is then tied to the database product. The CGI script can be written to work with most of the major databases on the market, and the Perl and php language will run on most if not all server operating systems.

The application should be modular in design as opposed to monolithic, and the modules should interface through a well documented Application Program Interface (API). The advantage of this approach is that the development can be more easily run in parallel, and individual components can be worked on without having to deal with the entire architecture.

Where feasible, components will be adopted from commercial or public sources rather than built from scratch. There is no desire to reinvent the wheel, no matter how much fun that could be.

High Level Business Process

The information system being designed will support a number of business processes. The business processes all revolve around the production of “on line” scholarly “journals”. In essence the information system will support the authoring and submission of papers, the peer review process, and the publication of the papers on line.

In the current world there are a large number of journals that are being made available as electronic editions over the internet. Some of these are traditional journals that are making forays into the world of electronic publishing such as the Elsevier journals, http://www.elsevier.nl. Other journals are published solely in the electronic domain, such as The International Review of Research in Open and Distance Learning, http://www.irrodl.org/. In either case the journals usually retain most of the trappings of a paper journal. In most cases they have collections of papers that are grouped as "issues", and the issues are released to the public as coherent packages.

In a broader sense the continued publication of "journals" reflects the business and processes of paper publication. The ties to paper publications cascade further than the authoring and publication processes. References to a paper refer to the name of the journal, and the issue number. Evaluation of the paper, and in a broader sense of the author, is largely based on the reputation of the journals that the author publishes in. (An example of how journals get ranked based on the number of citations that they receive can be seen in http://www.stern.nyu.edu/~wstarbuc/cites.html.) What is proposed here is a clean slate, where publication is independent of journals, while retaining peer review and improving ease of access.

The proposed system for "publication" of papers will need to support more than the process of submission, review, and dissemination of papers. It will also have to support a means of quickly evaluating the "worth" of the paper, perhaps by providing information on the number of times that the paper has been cited, or by supporting the ad-hoc "annotation" of the paper by readers. (This harkens back to the origins of scholarly publication, when copies of manuscripts were passed between scholars and they added comments in the margins. This "marginalia" was a useful measure of the papers worth.)

The proposed process is initiated by an author writing a paper, and submitting it over the internet to the system. An editor receives the paper and co-ordinates the peer review process. If the paper is to be made available before peer review then it is released as a "pre-print". Once the paper has passed peer review it is released as a "published" paper to the general public.

Some fields of study allow public access to papers as pre-prints before they have completed the peer review cycle. There is an open question as to whether the decision to allow pre-prints is made by the editor, the author, or by some other body. The system must be flexible enough to allow dissemination of pre-prints, and to clearly distinguish them from reviewed and published material.

A key feature to the system is an efficient indexing and retrieval system to facilitate access to the papers. Authors today provide some of the indexing information such as the papers title, the authors names, the authors affiliations, and keywords. A complete set of indexing information must be determined, and its collection and use must be supported.

The public can either access a paper through a search engine, or it may be sent to a reader identified through a previously registered interest in the author, or topic.

A high-level design of the document flow can be seen in Figure 1.

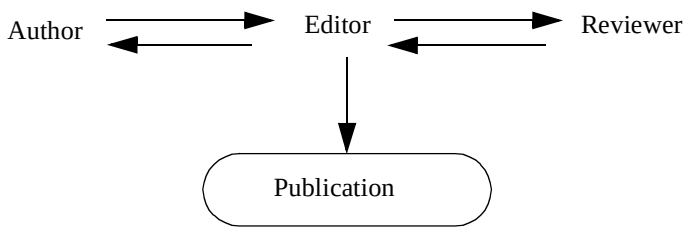

Figure 1: High Level Document Flow

High Level Functional Specification

This section of the document details how the system will function. This should address the business processes that were identified in a previous section, and will establish the high level functionality that the Technical Architecture will need to support.

Authoring a Paper

The process of authoring a paper will be substantially the same as it currently is. The only major change will be the need to structure the document to make the indexing meta-data available to the system. Software tools will be needed to assist the author in writing a paper that conforms to the specified structure, and that contains the appropriate meta-data. In order to be accepted by the user community these authoring tools must not be excessively complex or expensive. Ideally they would function as an "add in" to existing authoring tools such as Microsoft Word, or Wordperfect.

Once the paper is complete, there needs to be a mechanism for submitting it to the system. This will be accomplished by sending it as an e-mail attachment. The address that the e-mail was sent to would identify the body of knowledge, editor, or virtual journal that the paper is being submitted to.

Receiving a New Paper

When a paper is received by e-mail, it will be validated to ensure that it conforms to the required document structure. An e-mail will be sent to the appropriate editor, indicating that the paper has been received.

Peer Review

The editor will review the paper and either accept it for peer review, or reject it.

If the paper is accepted for peer review, then e-mail is sent to appropriate reviewers asking if they can review it. (Appropriate reviewers may be nominated by the system based on keywords, or perhaps on reference citations.)

At this stage the peer review process starts. The system should provide the editor with the necessary tools to track the papers currently under review. This should include the paper name and author, the names of the reviewers, when the paper was sent for review, the reviewer's summary opinion when it is available, and links to detailed notes. This entire tool could be built on the internet and made available through any web browser.

The same tool, (or a close variant on it), could be used by the reviewers to enter their comments on line.

Based on the peer review, the editor will decide to either sent the paper back to the author for revisions, accept it as written, or reject the paper.

Revision Cycle

If a paper is returned to the editor with reviewer comments that the editor feels should be addressed, then the editor notifies the first author of the requirement. The authors revise the paper and re-submit it to the editor. The editor then decides to either publish it, reject it, or send it out for more peer review.

This cycle of peer review and revision continues until either the editor decides to publish the paper, or until either the editor or the author decide to remove the paper from consideration.

The editor's toolkit mentioned in the Peer Review section above should also provide support to track the paper through the revision cycle.

Publication

Once the status flag of the paper is updated to indicate that the paper is accepted for publication, the index information is replicated across the distributed system. The status flag would also make the paper available to search tools.

Access on Demand

Users of the system will be provided with a search tool to identify papers of interest. They will then have the option of reading the paper as HTML through their web browser, or to read the paper as an Adobe Acrobat document. This type of access is often referred to "Pull" access, since the user pulls the document off of the server.

Access by Subscription

The alternative means of accessing the system will be through "subscription". In this model the user identifies the types of papers they are interested in through a web page. Their subscription is kept in the database, and when an appropriate paper is "published" on-line a copy is e-mailed to them. This type of access is often referred to as "Push" access, since the content is pushed from the server out to the user.

Core Technologies

A number of technologies are fundamental to the design of the system. It is useful to identify these core technologies at this point, as it will allow us to provide a more concise technical architecture model in a subsequent section.

XML

Back when UNIX was a relatively new invention, there were no graphical user interfaces. In order to have word processing that would work across a broad spectrum of terminals, documents were typed using plain text. Document elements such as bold text, italics, and paragraph breaks were noted in the text document by inserting text "tags". These tags were delimited by a special combination of characters that were unlikely to be encountered in the normal course of events. Before the document was printed, it was run through software that had special knowledge about the particular printer or terminal, and which translated the tags into the carriage returns, tabs, spaces, and special control sequences that would cause the printer or terminal to switch to bold or italic type. This use of "tags" to embed "metadata" in a document for later use in processing the document has continued to be used over time. HTML is an example of how this use of tags is still in use today.

Extensible Markup Language (XML) is similar to HTML which most people are somewhat familiar with. Whereas HTML uses "tags" to indicate how information should be presented, XML uses "tags" to indicate what the data content of the information is. This extends the model to not only include formatting metadata but to also include metadata about the document itself. For example, an XML tag may identify the name of the author. This may be used later to control how the authors name is formatted, and it may also be used to convey information programmatically to software that processes the document. This software may use the author information to index the document for later retrieval for example. The "Extensible" part of XML is that the number and specification of tags is not fixed in the specification, but can be extended by the document designer.

An XML document conforms to a template that is provided through the means of either a Document Type Definition (DTD) document or an XML Schema document. These documents identify what tags are valid, and the structure that the documents must conform to. For example we can create a DTD document that defines what a paper is, and have that document specify that a valid paper must have exactly one title, followed by at least one author, zero or more keywords, etc.. We can then write a simple program to extract out the information that we want to use to index the paper.

XML and its associated technologies are in the early stages of development and deployment. Products are being introduced, companies are being bought and merged, and some companies will inevitably not survive. It is difficult to predict exactly how some of the emerging standards will end up, and which companies products will prove to be dominant. We can identify some of the technologies will prove useful however.

Parsers are available to validate XML documents against a DTD. Similarly there are authoring tools that make use of a DTD to guide the writing of the document.

Extensible Stylesheet Language (XSL) supports the translation of an XML document into a formatted document suitable for use. The same XML document could be rendered by different XSL programs into HTML for viewing through a web browser, into Adobe PDF format for secure distribution, or into some other format.

In summary, the embedded XML tags in a paper will be used to render the paper into a presentable format, it will be used to implement indexes to facilitate later retrieval, and it may be used to provide linkages to other related documents. Although XML is "Extensible", we are able to control what tags are permissible in a given application, and how the document is structured, through the use of a template document called a DTD, or a XML Schema document. Because the documents are stored as tagged ascii text documents, incompatabilities with future technology is minimized. As document presentation technologies evolve, new XSL programs could be produced to render the documents using the new technologies when the documents are retrieved.

Distributed Database

As the project grows, as an increasing number of journals are brought on-line, and an increasing number of users want to access the system, there will be a serious problem of scalability. It is envisioned that participating research libraries will host nodes in a distributed architecture. Users will be able to access the system through any node. Users may choose to access a particular node because the internet topology provides a higher bandwidth connection, or because that node is less heavily used at that point in time.

Regardless of which node a user connects to, all papers must be accessible. A key component of this distributed architecture will be a database, which will allow rapid searching and retrieval of papers. The database will also be used to track usage details.

There is an open question regarding how much of the paper goes into the database. One possibility is to only store the indexing information along with identifying information that locates the paper outside of the database. (It is possible that the abstract should also be stored in the database and distributed across the system.) This has the advantage of simplifying the design, and probably improving performance. If the paper is stored within the database it would need to be stored as a large object, which complicates the database interface. A disadvantage of not storing the paper in the database would be that by separating the paper content from the indexing information, we would be limiting the advantage that could be obtained from the databases proprietary distributed architecture. In order to be able to evaluate this disadvantage, lets look at how distributed databases can be implemented.

Distributed databases fall into two broad camps. One is a design where data is held at only one location, but is accessible from all locations. The other design is one where data is replicated across all participating nodes. The relative advantages are fairly obvious, and generally entail trading off ease and speed of access against scalability. If all data is locally held then access is simpler and faster, but the size of each node does not scale well, (although user access does). If the data is held at only one location then access involves going across a network to get the document, which is more complicated and slower, but the size of each node will only grow as fast as the locally hosted data grows.

In the replicated model of distribution, the databases typically handle the replication of the database contents. The disadvantage of this is that it heavily predisposes the entire system to conform to one database software package. This contravenes the first of our architectural principles. It would also be difficult to implement since the nodes will be hosted by autonomous research libraries, which will have an existing infrastructure in place to manage their chosen database product.

What is proposed is a sort of hybrid scheme where the indexes are replicated and locally hosted, while the document content is held only at the node that "owns" the paper. (This does not rule out the use of "mirror" sites to mitigate access bottlenecks.) The replication of the indexes would not necessarily utilize a proprietary distribution mechanism. One of the major complications of replication is managing updates to the data in a way to ensure that a consistent view of data is maintained across the entire system. In our case the majority of the access will be read access, and updates will be practically non-existent. When the record is "published" locally, the associated index information will need to be replicated out to all nodes.

This also leads to the question of which site would "own" the paper. (Note that the term "owned" is only used in the sense of where the paper is kept, and does not necessarily imply any intellectual property rights.) It is proposed that the author would submit the paper to the research library affiliated with his University or institution, which would then take responsibility for "owning" the paper. Another possibility would be to have a national library, which would have a central repository for the papers, and restrict the University libraries to maintaining index databases.

There is another possibility, which entails having local data, and local indexes, with a distributed search mechanism. <Insert reference here>

Technical Architecture

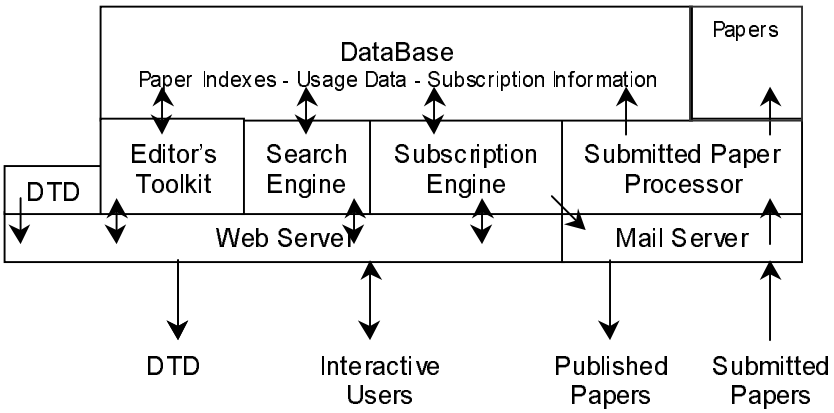

Figure 2: High Level Technical Architecture

This section of the document outlines the technical details of how the system will be built. Each major component of the architecture will be discussed separately. Discussions on how the component will be implemented, whether it will be bought or built, and alternative options will be discussed for each component. Refer to Figure 2 to see how the components fit together.

Authoring Software

Strictly speaking, the authoring process is outside of the scope of the system. It is carried out by any number of individuals, whom we will have no control of. We cannot specify what software they must use to author

papers, we can only specify what format, and to a lesser degree what content, the papers must follow. Not withstanding the preceding, without authors there will be no virtual journal. With this in mind we will look at some of the options available to authors.

Software	Price	Tested	Comments
None	Nil	No	Awkward and prone to error
MS Word & Bladerunner		No	
Adobe Framemaker		No	

Table 1: Authoring Tools

While XML documents can be written with any editor capable of saving the paper as a text file, this requires manual insertion of the XML tags. Since the placement of the tags and their content must conform to a standard which is provided as either a DTD document or an XML Schema, it is much easier to use a tool that is capable of reading the DTD document and using the information in it to assist the author in creating the tags. This is analogous to using an HTML editor to create web pages. Refer to Table 1 for a list of authoring tools that will be considered.

Interleaf was one of the two major products on the market ten years ago for authoring large complex documents. Today Interleaf has been re-born as "the e-content company", and specializes in software for authoring and using XML and SGML. One of their products is Bladerunner, which integrates with Microsoft Word to produce an XML editor. (They have a white paper available on the web, http://www.interleaf.com/products/whitepaper.htm.)

Ten years ago Interleaf's major competitor in the editing software market was Framemaker. Framemaker has since been acquired by Adobe. The current release of the software includes QuadralayWebWorks® Publisher Standard Edition software for HTML and XML capabilities.

Web Server Components

The main user interface to the system will be the web browser over the internet. The two major components that will need to be designed will be the DTD document and the search engine.

DTD Document

The purpose of the DTD document is as follows:

- To specify which XML tags are valid
- To define the structure of the document
- To specify the meta-data that the tags contain

The DTD document will be made available for ftp access over the internet by authors, and other interested parties via the web server. (Note that the authors may never need to see the DTD document. It will be required however by the authoring software as a template for the paper.)

XML Schema documents will probably supercede DTD documents in the near future since they can specify a finer degree of resolution in the document structure, and since they will conform to the XML standard themselves. At this point in time however, the prevalent mechanism is the DTD document, so that is what we will start with.

There are a number of related projects that have already defined a DTD document. Each of these will need to be evaluated, and one of these may either be used or taken as a starting point for our own work. Refer to Table 2 for a list of DTDs.

DTD	Comments
IXML	ICAAP XML – Electronic Papers http://www.icaap.org/software/ixml/ixml/ixml_1.0/ixml.dtd
RDF	Resource Description Framework – supports Dublin Core std. http://www.w3.org/RDF
ETD	Electronic Thesis and Dissertations http://etd.vt.edu/etd-ml/etddtd.txt

Table 2: DTDs Used Elsewhere

Search Engine

The ability to search for a paper is fundamental to the project. The attributes that the system will need to be able to search on will run through the system as a common thread. These attributes will be defined in the DTD document as XML tags. The authors will be required to provide these attributes in the document. The system will extract these attributes when the document is received, and will use these tags to build indexes to the document in the database. The search engine will use these indexes to locate papers of interest to users of the system.

The search engine database interface will initially be a php script. An advantage of using a php script is that it does not significantly restrict our choice of Web Server Software in later stages. It is also relatively easy to write and can be prototyped across a wide range of platforms. Performance of a php script is not as good as with a proprietary interface, and for this reason we may find that we will replace this component after the architectural proof of concept, or prototype, is built.

The search engine itself will be a functionality provided by the database. The major purpose of storing the indexing information in a database is to allow programmed and ad-hoc queries on the database to locate papers of interest.

The feasibility of having the php script dynamically generated from the DTD document should be considered. While this would add a layer of complexity, it would facilitate maintenance as the DTD document evolves in the early stages of the project.

Subscription Engine

Another php script will support the subscription process. This process will allow a user to register an interest in papers which may be published based on indexed meta-data. When a paper is published which fits the profile then the paper will be made available to the user.

Editor's Toolkit

The editor will track the status of papers in the review cycle. The editor will use it to assign papers to reviewers, to review the reviewers comments, and to track the papers progress through the review cycle. A variant of the Editors Toolkit may be used by the reviewers to register their comments.

Database Server Components

The database acts as a repository to store indexing information, and also as a central source of data for tracking usage and performance of the system. A logical database model may be seen in Figure 3. <The model needs more work. Take it as a place holder for the time being.> Refer to Table 3 for a list of database products considered for the prototype.

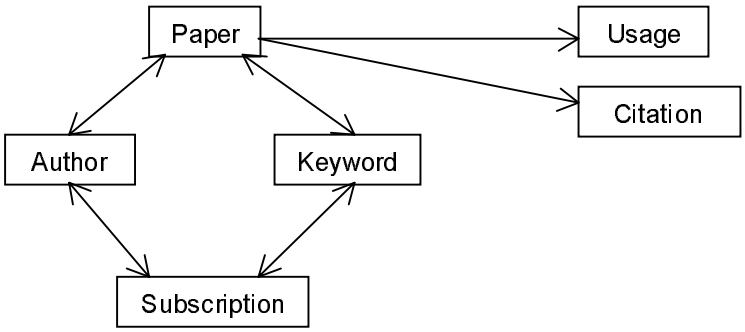

Figure 3: Logical Database Model

Paper indexes

Although the XML documents will not be stored in the database, the meta-data carried in some of the XML tags in the documents will. For example the paper title, the author(s) name(s), and the keywords will be stored in a table in the database. The table will also contain information necessary to locate and retrieve the document such as the node that is storing the document, and where the document can be found on that node.

Usage data

The system should store information regarding which papers have been accessed, and the times of access. Depending on the business model the security model for the system may also be stored in the database.

Citation data

Citation information will need to be extracted from the paper and stored in a database. This information will be used to assess papers.

Subscription information

In addition to providing access to users who are searching for papers, we will be providing access to users who register an interest in a particular author, or keyword. In these cases the paper, or an email referencing the paper, will be sent to the registered user. Information supporting this function will need to be stored in the database.

Database	**Price**	**Comments**
Oracle		Portable across multiple server architectures
Microsoft SQL Server		Requires NT server
mySQL	Free	

Table 3: Relational Databases

Mail Server Components

It is difficult to say what mail software or what hardware the nodes will use. We may have to make an assumption for the purposes of the project. It would be useful to conduct a survey to see what is currently used in research library environments.

Submitted Paper Processor

It would simplify the project if authors could be required to ftp papers onto the site, or if they could be required to cut and paste their paper into an HTML form. Reality dictates however that many authors would have difficulty with such an interface. It is safe to assume that all authors are able to make use of e-mail though, and so that will become the means for authors to submit a paper.

When a paper is received the attached paper will need to be extracted from the e-mail and processed. This processing will:

- Validate the paper against the current DTD document.
- Determine if the paper is an original submission or a revision.
- If original then the paper will have the indexing information extracted and inserted into the database. If a revision then the indexing information will be extracted and the current database record will be updated.
- The paper will be stored on a file server.

Other Components

…….