IST-2001-33127

SciX

Open, self organising repository for scientific information exchange

D9b: Architecture Synthesis

Responsible authors: Brian Clifton, Grahame Cooper
Co-authors: Ziga Turk, Gudni Gudnason, Tomo Cerovsek

Access: public
Version: 2.0
Date: May 10, 2003

Executive summary

Enabled by the Internet and the Web, new models of scientific publishing are being proposed. In the SciX project, a separate workpackage is dedicated to business process re-engineering. In its first phase it analysed the as-is processes. Towards the end of the project it will suggest the to-be processes. These will be, to a large extend, influenced by the work of, for example, Guedon[1], that suggest a separation of the main functions of the journal to allow a steady progression in the career development of academics from initial, exploratory forays into the publishing world, through a process of increasing levels of peer review and recognition, ultimately to full academic rigour within established communities through related journals.

The SciX pilots are being developed in *parallel* to the re-engineering efforts. The design goal for the architecture is to define a flexible set of components that can be combined in several different ways in order to support different current and future publication models. These components will be implemented as Web services[2]. Applications will be developed on top of these services in order to support a given publication model. In this report, architectures of applications supporting rather traditional forms of publishing are presented, however, they use modern publication media. We believe that the flexibility of the components will be sufficient to allow the consortium to create support for the re-engineered forms of publication. This open-ended flexibility is also the main difference between the SciX pilots and the monolithic digital library applications.

This document defines, how SciX will support Web based applications to publish (1) a **refereed journal**, to organise the publication process of (2) a **conference**, to set up a (3) **digital library**, that may be (3a) personal, (3b) institutional or (3c) societal and to publish (4) a **value added publication**. Even though not defined in the Technical Annex, SciX may also demonstrate a non-browser-based application and show, how one could (5) access SciX digital libraries directly from applications such as MS Office, Open Office or citation management tools such as Endnote or Procite.

The applications will provide a thin integration layer over **Web services** where the bulk of processing, searching, storage and other operations take place. The Web services required to support the above applications are (a) repository of works, (b) reviews, (c) recommendations, (d) collections, (e) discussions, (f) user management, (g) knowledge management and (h) value added services.

The services will talk to the applications and to each other using XML language over HTTP protocol. The repository services will talk to other archives using the OAI-PMH protocol. The repository service will also offer an interface to its publications using this protocol.

The prototypes are under development. First the core functionality of the services and applications is being created and tested in real-life applications running on the Web. In the second phase, Web interfaces to the services will be provided and the method of communication changed from a proprietary calling mechanism to open XML protocols.

---

[1] Guedon, J.C. (2001). In Oldenburg's Long Shadow: Librarians, Research Scientists, Publishers, and the Control of Scientific Publishing, proceedings, Association of Research Libraries, Proceedings of the 138th Annual Meeting, Toronto, Ontario, May 2001 - http://www.arl.org/arl/proceedings/138/guedon.html

[2] Version 2.0 of this document uses the term "service" strictly in the W3C sense. Services are never used by the end users directly but are accessed by other services and by applications. End users interact with applications. For the same reason, the "value added services" was renamed to "value added publications".

Release history of this deliverable (D9):

date	changes
22.7.2002	Outline document for D9.
23.9.2002	First draft of D9
6.11.2002	Version 1.0 of D9 released
20.1.2002	Draft of D9a
31.1.2003	Released D9a v1.0
3.4.2003	Review delivered to the consortium
17.4.2003	15 slides explaining the architecture released
10.5.2003	Version 2.0 of D9a released, called d09b.pdf
2.11.2003	D9a version 2.0 renamed to D9b; title changed accordingly.

Table of Contents:

1. ARCHITECTURAL PRINCIPLES

This section defines key *technical* requirements underpinning the architecture. The initial *functional* requirements have been addressed in Deliverable 8, Section 9.

1.1 Requirements

The main technical requirements for the architecture are:

- Provide a **flexible architecture**, that can be used to support different models of publication and use cases including ones defined in Deliverable 1 and Deliverable 8 in SciX but also future ones, defined as the result of the business process re-engineering and address the trend in electronic publishing that is in the separation of the main functions of publishing.
- Provide an **open architecture**. The architecture should allow to be extended in several ways, by more or less knowledgeable application developers and programmers. Also the data managed by the pilots should be available to 3rd party harvesting.
- Provide a **modular architecture**. There are a number of monolithic digital library applications on the market. SciX is not competing with them but is trying to demonstrate that by defining a set of modules implemented as Web services, different application can be created very rapidly for changing requirements.

1.2 Requirements discussion

The electronic publishing area is extremely fast moving, developing at a significant pace, particularly in the areas of inter-working. Therefore, a modular architecture is proposed for the SciX system, allowing functionality to be included, left out, added, or replaced relatively easily in any particular application.

A widely accepted principle, particularly promoted in this arena by the OAI[3] (Open Archives Initiative) and SPARC[4] (the Scholarly Publishing and Academic Resources Coalition), is the **separation of data storage from service provision**. From an academic viewpoint an even more important is the separation of the roles that a journal performs:

- certification: the primary purpose of a journal is to validate, or certify, the research quality of a research publication in relation to its usefulness to a particular scientific community (or communities). This certification is presented in the form of a journal "brand", and is based on the credibility of its editorial board.
- access: journals distribute information and provide an information exchange platform
- establishment of priority - through the management of submission and publication dates, journals maintain a log book of who thought of something first
- archiving. Through submission to major libraries, journals are archived for long periods of time.

[3] See http://www.openarchives.org/
[4] See http://www.arl.org/sparc/home/index.asp?page=0

- research assessment. Through bibliometric studies based on published papers in journals, research quality assessment is performed, by counting the number of publications and citations.

It has been pointed out[5] that improvements may be made in the scientific publishing process if these logical elements were separated from one another.

An attractive publishing model, which is being promoted quite widely, is one in which authors submit their papers to an institutional "archive" (i.e. repository) that acts as the primary location for the article. This article may then be published in a journal or at a conference. In this situation, it will be necessary for a journal to link to an institutional repository, or to retrieve the article and/or metadata from that repository.

If this is to be accommodated, there will need to be a separation between the journal support and repository support. In this model, it will be important to take steps to ensure the authenticity of refereed articles.

Developments in electronic publishing are taking place through a range of initiatives, including the OAI, SPARC, and various development efforts such as eprints.org, DSpace (MIT), and CDSWare (CERN Document Server). It is important to be able to interact with these and similar or related systems in the future as this will be an important factor in the success of any online application provided by SciX. This implies a separation between the repository implementation and the means of accessing the repository, with possibly alternative access interfaces available in the future.

While SciX will provide its own basic digital library repository its additional services (reviewing etc.) should be able to interact with any open third party digital library tool.

Additional "knowledge management" (KM) services may be provided on top of the repository. This includes such things as comparison of articles, searching for similar articles, summarisation, and linking between articles. These services may be provided as additional modules, separate from the repository. It will be important that such service are able to access metadata and articles from other repositories in addition to (or instead of) a local one if they are adequately to support activities such as authoring.

Whilst services may often be accessed through a simple, Web-based, thin-client interface, it is important to allow for access by other means, including more closely integrated approaches. For example, authoring tools will need to integrate quite closely with KM services and metadata services in order to provide the kinds of benefits identified in the requirements analysis. In this way, the application may present the service in a form that fits more naturally into the context in which the user is working. This implies that all services should be defined in the form of an API that may be accessed by a range of possible clients, with some user access being provided by separate, Web-based client applications where appropriate. This

[5] See for example *The SPARC White Paper*, SPARC, 2001 (http://www.arl.org/sparc/resources/whitepaper.pdf). Also *Gaining Independence through Institutional Repositories*, Alison Buckholtz, 2nd Workshop on the Open Archives Initiative (OAI), CERN, 2002 (at http://documents.cern.ch/AGE/current/fullAgenda.php?ida=a02333)

would also allow for the possibility of client applications that make use of and integrate several different services to provide a suitable working environment for a user.

2. SELECTED USE CASES

SciX originally set out to implement:

- a generic digital library provides a basis for a topic archive, an institutional archive or a personal archive,
- support for a scientific conference,
- support for a refereed journal,
- value added publications (services).

A detailed analysis of the related processes is documented in the Deliverable 1. The requirements related to the processes were described in Section 9 of the Deliverable 8. Here we briefly enumerate the most important ones and later on (Section 4.6) detail the most characteristic ones.

2.1 Digital library use cases

- Enter an article set
- Destroy an article set
- Add an article to a set
- Remove an article from a set
- Carry out a literature search
  - Browse author list, keyword list, index
  - Browse articles by e.g. author, paper title or keyword
  - Search articles (by keywords, authors, etc.)
  - Build search results list
- Generate an article reference
- Manage reading lists
  - Enter a reading list
  - View and browse a reading list
  - Maintain a reading list
- Set and change status of article
- Set and change version of a article
- Retrieve an article – download
- View article full text
- User profiling
  - Maintain a user profile
  - Store search profile
  - Initiate a stored search profile
  - Notify user of updates per profile
  - Add article reference of favorites
- Discussions
  - Initiate a discussion on an article
  - Contribute to a discussion
- Add a comment to article

2.2 Scientific conference use cases

- Carry out an article search

  - Browse articles by conference series, conference or session
  - Browse articles by author, paper title or keyword
  - Search for articles e.g. in: conference series, conference, by: author, keyword
- View abstract, full paper
- Enter and maintain a conference series
- Enter and maintain a conference in a conference series
- Enter and maintain sessions in a conference
- Register a conference participant - add to user database
- Maintain registered participants
- Assign a participant to conference
- Submit an abstract or paper to a conference
- Withdraw submission
- Assign a paper to a conference session
- Enter and maintain a board of reviewers for a conference
- Assign abstracts, papers to review to a reviewer
- Review abstract or paper e.g. comment, reject, accept etc.
- Notify reviewers
- Monitor and chase review progress e.g find late reviews
- Notify paper authors of review
- Publish conference proceedings, page-, author- index, list of papers
- Publish conference, session schedule of speakers

2.3 Journal use cases

- Create, set-up and maintain a journal
- Add a new journal volume
- Author submits an article for review
- Publish a journal volume (make it available to public)
- Archive a journal volume
- Retrieve journal volume from archive
- Add and maintain reviewers (add to user database)
- Create and maintain an editorial board for a journal
- Add and maintain members on the editorial board
- Editor processes a submitted article
  - Assign reviewers to an article
  - Notify reviewers of article to be reviewed
- Track review status of articles and notification of events
- Review article and submit review
- Process a reviewed article
  - Accept an article for publication in a journal volume
  - Reject an article for publication in a journal volume
  - Add an article to a journal volume
  - Remove an article from journal volume
- Generate statistics about review process
- Maintain Journal distribution
  - Set-up and maintain a journal distribution list
  - Send a journal volume to a distribution list
- Carry out an paper search

  - Browse available journals
  - Browse by journal volume
  - Browse articles by e.g. author, title or keyword
  - Search articles (by journal, volume, keywords, authors, etc.)
- Add a discussion to an article
- Add an annotation to an article

2.4 Web service administration use cases

- Register user with service and assign roles
- Register other services and applications with the service
- Grant/revoke usage rights by services an applications
- Monitor services usage by other services and applications
- Monitor resource usage (CPU times, disk space)
- Provide raw export/input of data managed by the service
- Allow for the management of the service parameters
- Display service documentation in machine (WSDL) and human readable format

2.5 Value added publications

- Register authors, co-authors
- Manage users
- Manage author workspace
- Upload-download document (e.g. digests, summaries and reviews)
- Checkin-checkout document
- Create and maintain metadata record for document
- Create and manage control access list for document
- enter & maintain bibliographic notes
- enter & maintain citations
- create & maintain bookmarks
- generate standard citation for inclusion in article
- Search and browse SciX digital archives
- download scientific publication from SciX repository service (to local machine)
- retrieve metadata record for scientific publication from SciX repository service (to local machine)
- enter & maintain search criteria
- enter & maintain business partner
- enter & maintain syndication catalogues
- enter & maintain schedule for automatic content delivery
- accept and store content syndicated from SciX VAP
- schedule delivery of content to SciX VAP or presentation system
- notify author of use of material.

2.6 The applications

The use cases will be supported in the four main applications:

- digital library,
- scientific conference,
- scientific journal,

- value added publication,
- Web service administration (one for each service though very similar to each other).

3. THE INFORMATION SCHEMA

All these publication outlets described in Section 2 are built around about a dozen different concepts, which are combined in different ways to provide different functionality, different quality assurance levels and different levels of establishing the community or authors and readers. To implement any non-trivial application, several of these concepts are needed. The concepts are shown graphically in Figure 1. Note that the Figure describes a conceptual model of the system and does not imply a monolithic implementation.

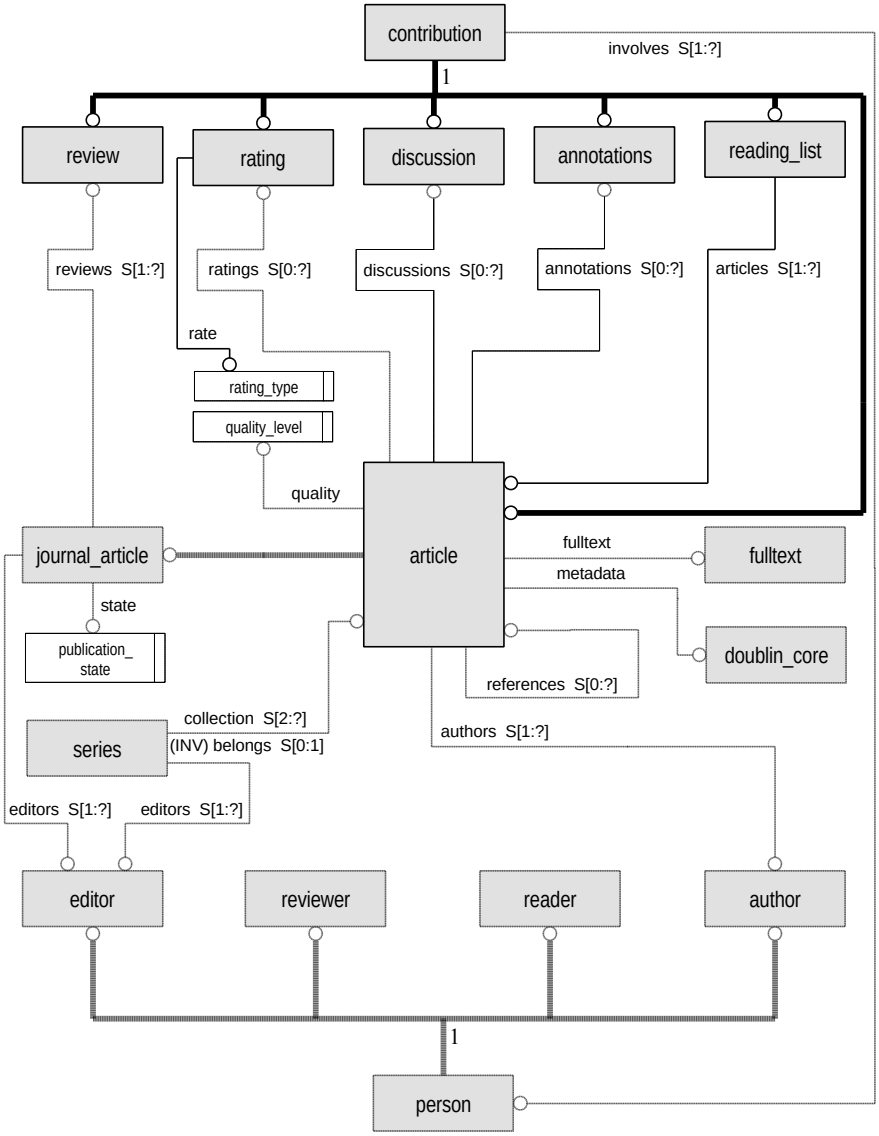

Figure 1: Main concepts to be supported by SciX pilots, drawn in ISO EXPRESS-G language.

The central concept in the diagram is an article. It has Dublin Core metadata set and full text. Article belongs to a series. It has a quality level, which is one of preprint, internal review, conference peer review, journal peer review. Article may start a discussion thread which recursively spawn more discussions. Articles may be privately or publicly annotated. Articles have references to other articles. Readers may rate the article as poor, average, good or excellent. Readers might combine articles into reading lists, for example a teacher would collect a number of articles for the students to study. Another reader might want to make a bibliography of most relevant papers on a topic. Journal and conference article is a special kind of an article. It has an editor. It has reviews. It may be in one of several publications states such as abstract submitted, article submitted, in-review, rejected, accepted, accepted for re-review, finalised, published.

The support for the "article" object is provided in quite a few digital library applications e.g. ePrints. The support for all other objects varies and is usually tightly coupled with the own "article" object. The SciX architecture is relying on the Web services technology to bridge this gap. Each of the objects is supported by a Web service, which generally provides for an administrative and an end-user GUI as well as a service-service level interface.

4. LOGICAL ARCHITECTURE

A modular architecture is proposed for the SciX system, allowing modules to be included, left out, added, or replaced relatively easily in any particular implementation or application. This is made possible because the schema shown in Figure 1 is not implemented in a monolithic relational database application but rather by a numbed of services and applications. The collaboration among them in applications is established by programmers and system integrators at runtime.

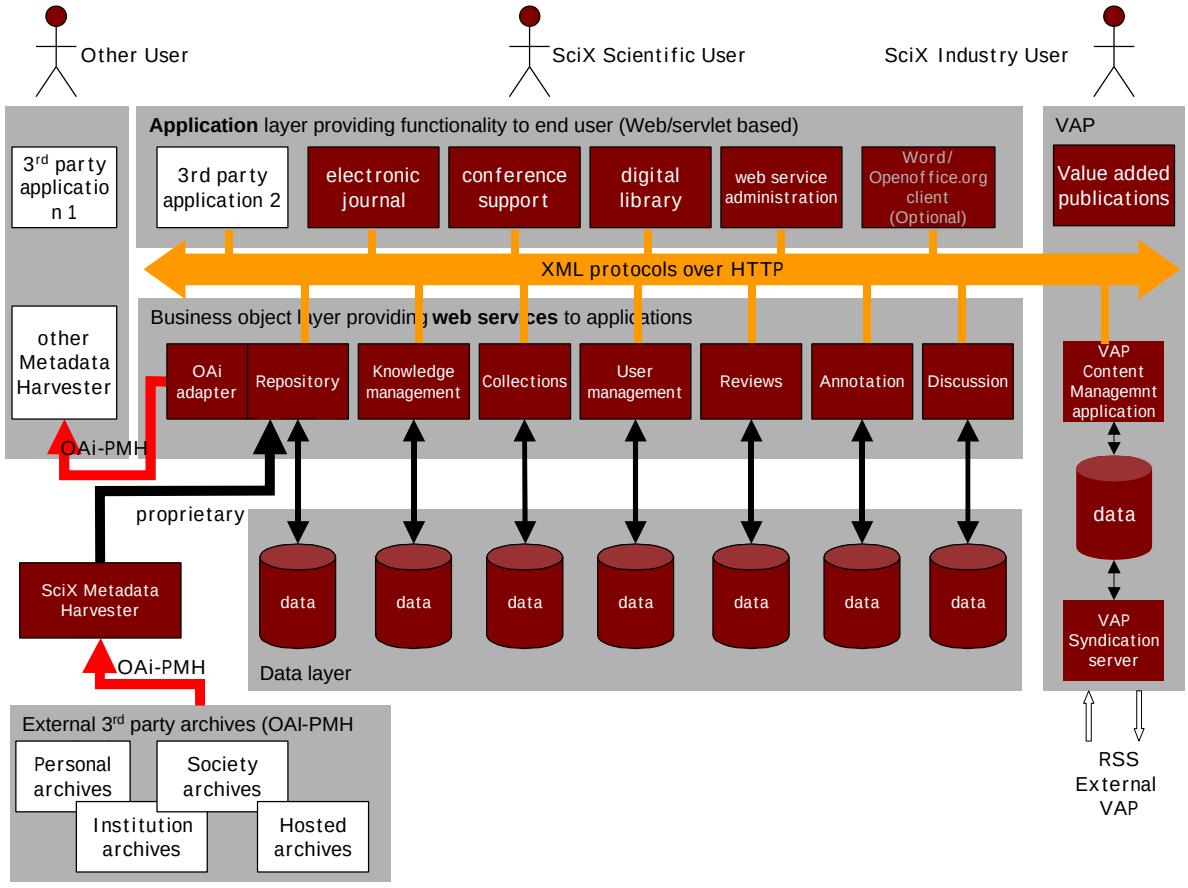

Figure 2: Logical architecture of the SciX pilot showing all major SciX and 3rd party components and ways of communication among them. SciX developments are shown as dark boxes with white text. 3rd party components are shown as white boxes with black text. XML over HTTP communication is denoted by orange, OAI-PMH by red and proprietary or other communication by black arrows.

Figure 2 depicts all major components of the SciX pilots. It is important to stress, that it does not show one system, but various components that, when properly combined, result in different applications offering different functionality to the end user. The figure is described top to bottom.

SciX applications will be used by two main classes of users (the top layer): scientific users and industry users. Each may have very different roles, such as researcher, editor, reviewer etc., as defined in Deliverable D8.

Users do different kinds of operations in the system - they take part in different use cases as discussed in Section 2. They interact with the system through applications. The applications are shown on the second layer of the diagram.

4.1 Applications

Applications are described in relation to Figure 2, left to right.

4.1.1 3rd Party Application 1

This example application's relation to a SciX based system is only through accessing the data in the SciX digital library (managed by SciX repository service) via their metadata harvester that is harvesting a SciX repository using the OAI-PMH protocol. SciX repository includes a module - the OAI Adapter that makes it possible to harvest by that protocol.

4.1.2 3rd Party Application 2

This example application is using SciX services. For example, it could use the SciX repository service but build around it some particular workflow system to support a particular publishing model. This application is talking to SciX services using an XML protocol.

4.1.3 Electronic journal

The electronic journal application provides the functionality to support the use cases defined in Section 2.3. It supports the submission, reviewing, rewriting, publishing, reading, citing, annotating, discussing etc. electronic journal papers. The application is pulling together, in a particular way, the resources offered by some of the services in the services layer. The Figure 3 shows what components of the Figure 2 are actually used by a journal application.

Journal application supports the workflow in a refereed journal publishing and include: set up & maintain journal, create new journal edition, add article to journal edition, remove article from journal edition, publish journal edition, set up & maintain journal distribution list, distribute journal edition, archive journal edition, retrieve journal edition from archive, enter & maintain reviewers, set up & maintain review board, add & maintain reviewers on review board, submit article for review, notify reviewer of article to be reviewed, receive article review, track review status of article, accept article for publication, reject article for publication, pass accepted article to publication process, review performance of reviewers, etc.

In SciX this application will be demonstrated with the case of the ITcon Journal[6].

[6] http://www.itcon.org/

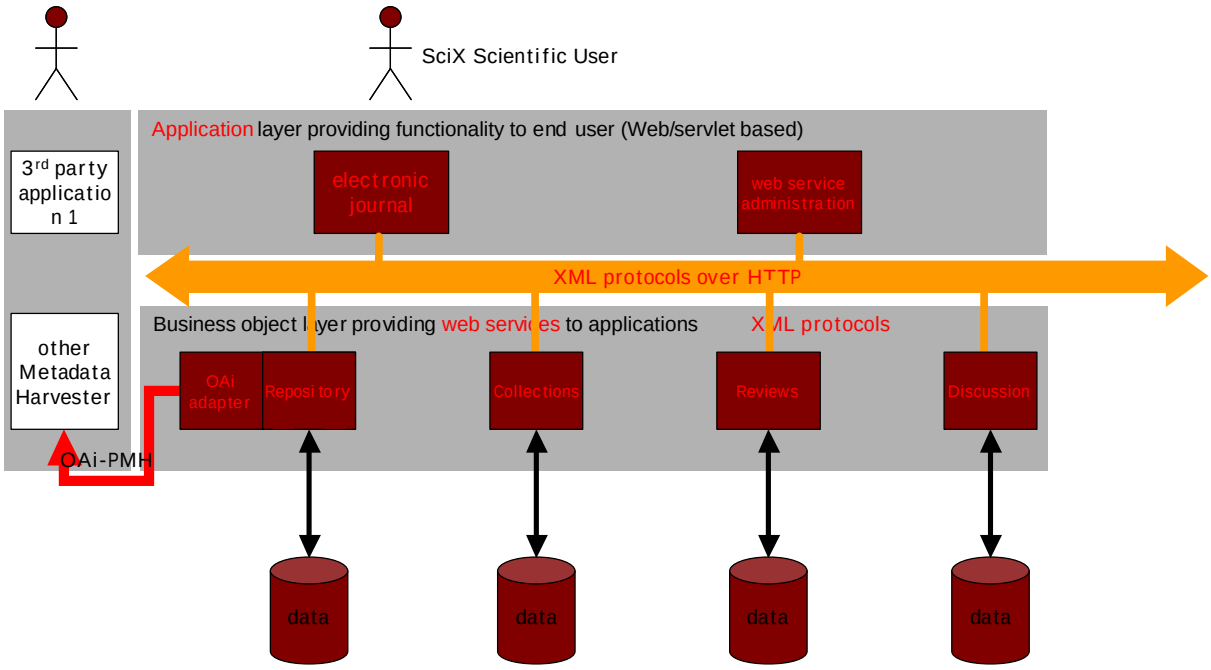

Figure 3: Configuration of components required to support and electronic journal application.

4.1.4 Conference support

The conference application provides the functionality to support the use case defined in Section 2.2. It supports the registration of participants, submission of abstracts, reviewing of abstracts, submission of papers, reviewing and publishing. The application is pulling together, in a particular way, the resources offered by some of the services on the services layer. Particularly the workflow in the conference support is different to the journal. In SciX this application will be demonstrated by supporting the ECPPM 2004 and ELPUB 2004 conferences.

4.1.5 Digital library

This is the most basic of all the applications and mainly provides for access to the repository service. Other services may or may not be included, depending on how feature-packed a digital library application is desired to be. The use cases that this application supports are defined in Section 2.1. In SciX this application will be demonstrated by digital libraries such as cumincad.scix.net, itc.scix.net and several others.

4.1.6 Web service administration

Although Web services are only used through the applications, they do need an administrative user interface that is represented by this box. The administration functions include extending or customizing the schema of the data that the service is handling, setting up access controls, monitoring use through service level log files, doing large scale data management (such as backing up data etc).

4.1.7 Value Added Publications (VAP)

VAP applications support the use case defined in Section 2.5. VAP applications provide the normal industry user with access to industry specific articles such as digests, reviews and summaries that are produced, by VAP editors and publishers, from SciX digital archives. The VAP provides the functionality for maintaining edited articles, collaborative authoring and versioning and publication of articles to Web sites and other VAPs. The application enables e.g. digest editors to use the SciX repository service to search and browse the digital archives, to read and upload papers, produce citations and bibliographic notes.

4.2 Services

The 3rd layer from the top of Figure 2 shows the SciX services. The WWW Consortium[7] defines Web service as "*a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols*".

This section of the report lists the items of functionality defined in the requirements analysis against to the modules of the system architecture, allocated according to the principles defined in Section 1. (Value Added Publications are treated separately and described in Section 5 of this report.)

4.2.1 Repository service with OAI adapter

The repository service provides basic functionality such as upload article, check for uniqueness, create unique article ID, remove article from repository, cross-reference articles, create new version of article, bulk upload articles, harvest articles from external sources, export articles in standard format, enter & maintain author details, enter & maintain institution details, retrieve an article by unique id, retrieve article by key-word search, browse repository via citations, search for similar articles etc.

The service also provides for access to other repositories. OAI-PMH are planned to be supported by the pilot but the modular architecture should allow for other access methods, such as Z39.50 as well. Functions include: upload and maintain metadata, harvest metadata from external sources, export metadata from repository etc.

4.2.2 Knowledge management

The KM service will provide enhanced abilities for categorising, classifying, browsing and searching out information and will be based on knowledge management techniques, particularly statistical text analysis and machine text learning.

[7] http://www.w3c.org/

4.2.3 Collections

A collection is defined as a bag of repository elements. It is created by a user, for example to define a reading list for her students or to pick, from the bibliography, a selection of papers on a given topic.

Related information[8]: title, owner, description, a list of works

Related functions[9]: search collections, show collection, create collection, delete collection, add work to collection, remove work from collection.

4.2.4 User management

While anonymous access is important because many people do not like to be bothered with log-ins and passwords, any active use of the services requires an identification and other related actions.

Related information: a user profile; user id, name, email address, affiliation, etc.

Related functions: enter & maintain user, list users, manage system access, authenticate users check user authorisation, create audit trail etc.

4.2.5 Reviews

The reviews service handles the information about reviews and the reviewing process.

Related information: work, reviewer, date, verdict, comments

Related functions: assign review, find reviews, find missing reviews, show review, submit review

4.2.6 Annotations

The service will allow users to add annotations to articles for their own reference purposes. Annotations may be personal or public.

Related information: work, owner, title, annotation, access.

Related functions: find annotations, show annotation, post annotation, delete annotation.

4.2.7 Discussions

The service provides support for threaded discussion about works or any other item.

Related information: work, reply-to, title, comment, owner.

---

[8] The services will be self documented and this information, including details about the types of fields, possible values etc. will be delivered by the service itself.
[9] This information as well, with calling conventions and instructions will be included in the WSDL definition of the service. All services will provide a WSDL definition of itself.

Related functions: add entry to thread, remove entry from thread, list threads, search threads, select thread.

4.3 Configurations

This section describes, how the various elements of Figure 2 can be combined to provide different kinds of applications. An "x" means that the service is essential for the application and a "?" means that it is optional. Blank cell means that the service is not needed.

columns: services rows: applications	Repository with OAI adapter	Knowledge management	Collections	User management	Reviews	Annotations	Discussions	Demonstrator
Digital Library	X	?	?	X		?	?	cumincad, itc.scix.net DLs
Conference support	X			X	X	?	?	ECPPM 2004, ELPUB 2004 conferences
Electronic Journal	X		?	X	X	?	?	ITcon journal
Value Added Publication	X			?		?	?	ReoCenter at IBRI
Web service administration	X	X	X	X	X	X	X	all

Table 1: Possible configurations of services needed to build applications. Last column: planned demonstrators.

4.4 Protocols

This section describes, how the various elements of the Figure 2 as well as actors in the diagrams in Section 4.6 communicate with each other.

- the orange "bus" in the Figure denotes the communication among the SciX Web services and applications. If two components that need to collaborate are on different machines, they would communicate using XML protocols over HTTP. Our analysis (SciX Deliverable 8) showed, that XML-RPC (http://ws.apache.org/xmlrpc) provides sufficient functionality. The use of the SOAP Web service protocol is still seriously being considered. If the two components that need to collaborate are on the same machine, a more efficient mechanism that does not include HTTP protocol overhead, but system calls will be used. The selection between the two mechanisms will be done automatically and transparently to the user who will not need to care about physical locations of the services.

- the red arrows denote the OAI-PMH protocol. This protocol is increasingly popular for the exchange of digital library metadata. Section 4.5 discusses the affected components.

- black arrows denote private and other protocols, private to one SciX service and of no interest to anyone else. SQL is an example of such a protocol for data access.

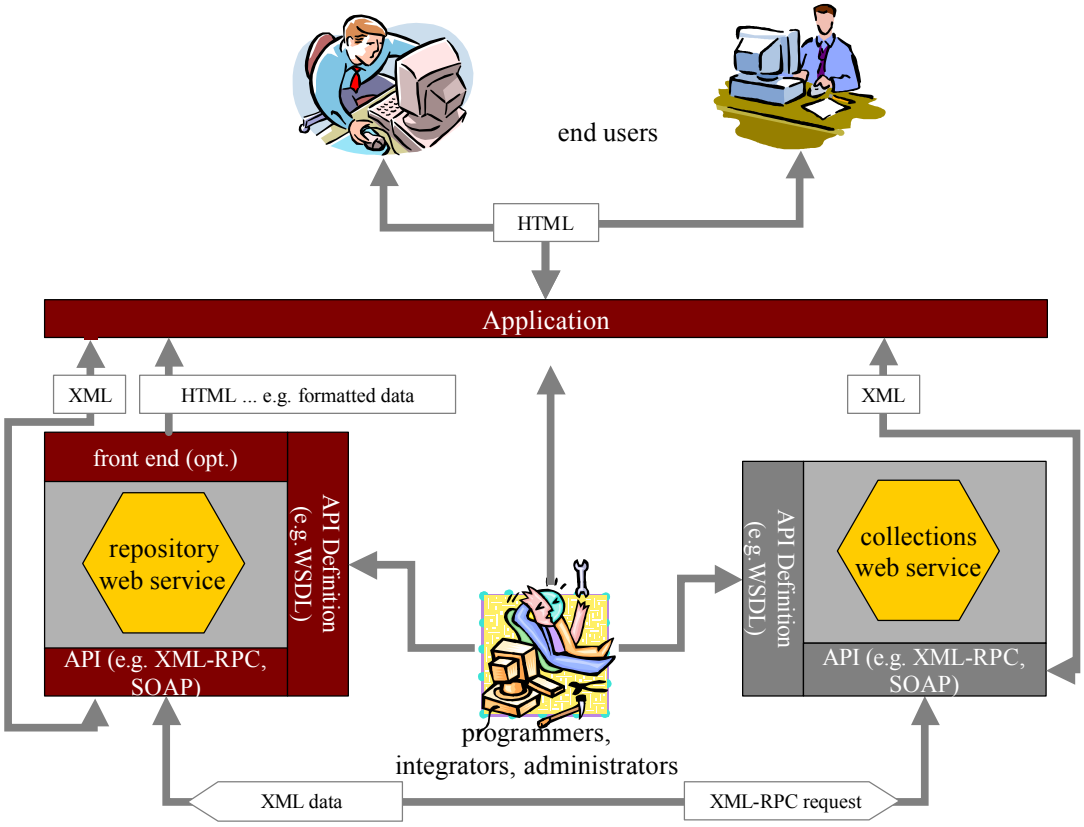

Figure 4: Detailed depiction of communication between services, applications and users.

The Figure above shows what happens in a typical scenario. Firstly, the communication is established by the programmers (centre of picture) who reads the API definitions of the two services and integrates them into, for example, a digital library application.

A digital library user sees an interesting paper in the repository and would like to see the details. The application asks the repository service to "show" the paper. It is displayed by an application, based on the data delivered (in XML or HTML) from the repository Web service. XML form is used if the application would like to do its own formatting of the data, HTML if the application is happy with the default way in which a service formats the data. The paper may be a part of a collection. To learn about this, the repository service asks the collections service using XML-RPC, if the paper in question is part of any collections. Collections service replies, in XML, with a list of collections in which the paper is present. This information is used by the repository service to generate XML or HTML data to return to the application. Finally, the application presents this to the end user inside the browser window.

4.5 Metadata harvesting

SciX architecture is compatible with the OAI-PMH. SciX will use an existing open source metadata harvesting software to feed 3rd party archives into SciX Repository service. SciX

will also provide an OAI-PMH adapter to the Repository Service, so that SciX Repositories could be harvestable by 3rd party harvesters.

4.6 Sample sequence diagrams

This section gives examples of how some typical transactions would be carried out. The original D9 report documented (to a lesser detail) a number of such cases. We define here three typical cases.

The diagrams are in the UML notation ("sequence diagrams") and show the interactions between various actors (humans or software) in columns when performing a sequence of actions (rows). Most software components mentioned are services or applications. Some are part of the system infrastructure: e.g. system mailer is smtpd (such as Sendmail or Postfix) with IMAP (or POP/3) or similar. The communication among them is shown as arrows representing method invocations (or procedure calls), or half arrows representing event propagation. Whilst these arrows represent logical interactions, independent of particular protocols and transports, in practice, these will be implemented mostly through XML over HTTP, except arrows related to humans, which are implemented through Web-browser interfaces in HTML over HTTP. See also Section 4.3.

4.6.1 Finding a paper

In this use case, a researcher wishes to find articles according to keyword and author details, etc. If required, the search may be enhanced through additional keywords supplied by the knowledge management service through a thesaurus type lookup against ontology, user's profile, paper clusters etc.

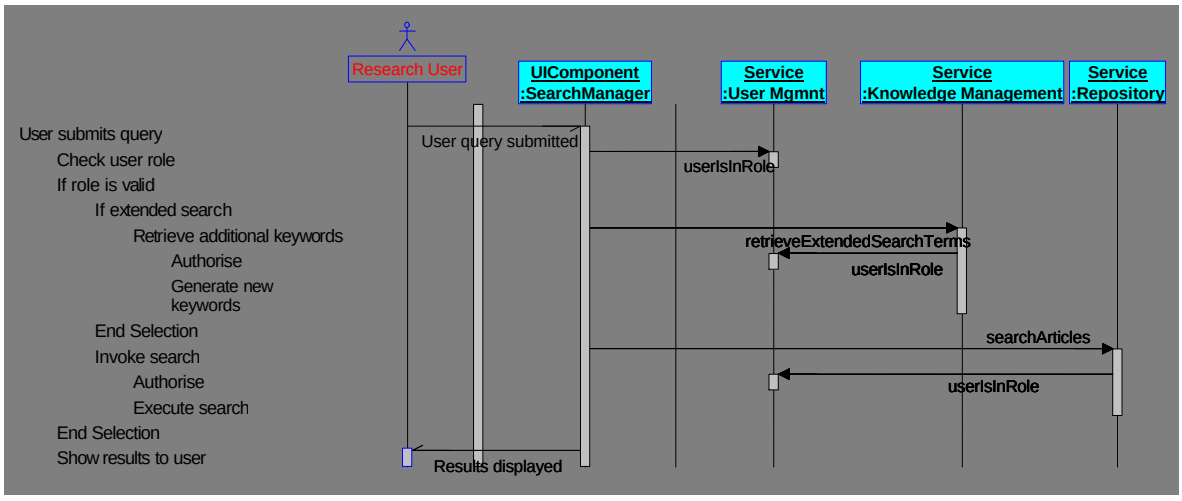

Figure 5: Finding a paper.

4.6.2 Author submits an article for review

An article is submitted by an author to a journal for review. The article is placed in the repository with appropriate state information, a new instance of the review process is started, and the editor is notified by an appropriate method (email is used in this example).

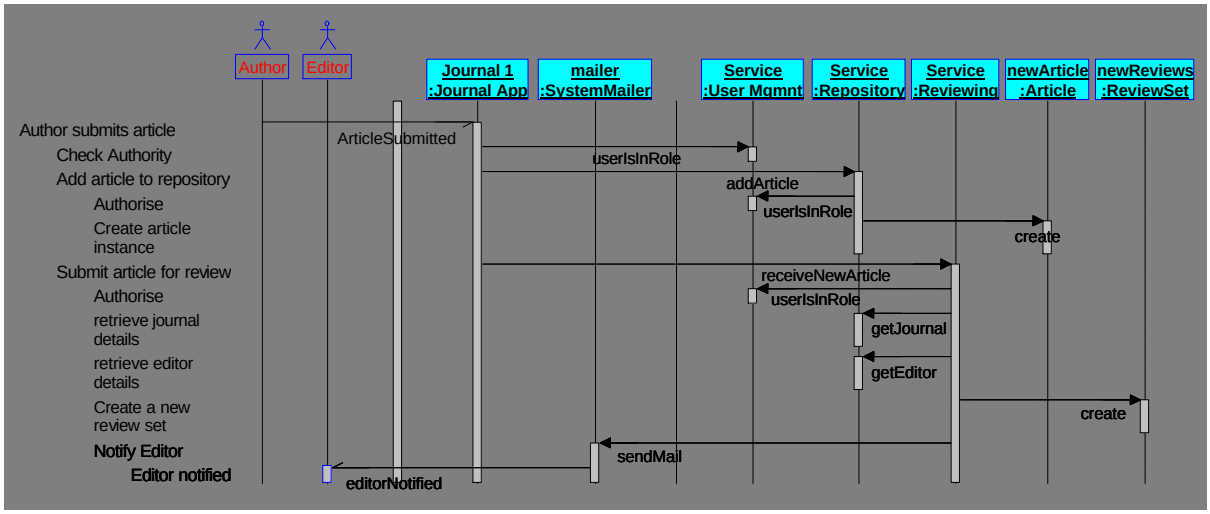

Figure 6: Submitting an article to a journal.

4.6.3 Editor processes a submitted article

The editor, on being informed of a new article for review, starts off the review process by assigning the required number of reviewers from the journal's editorial board. The reviewers are informed.

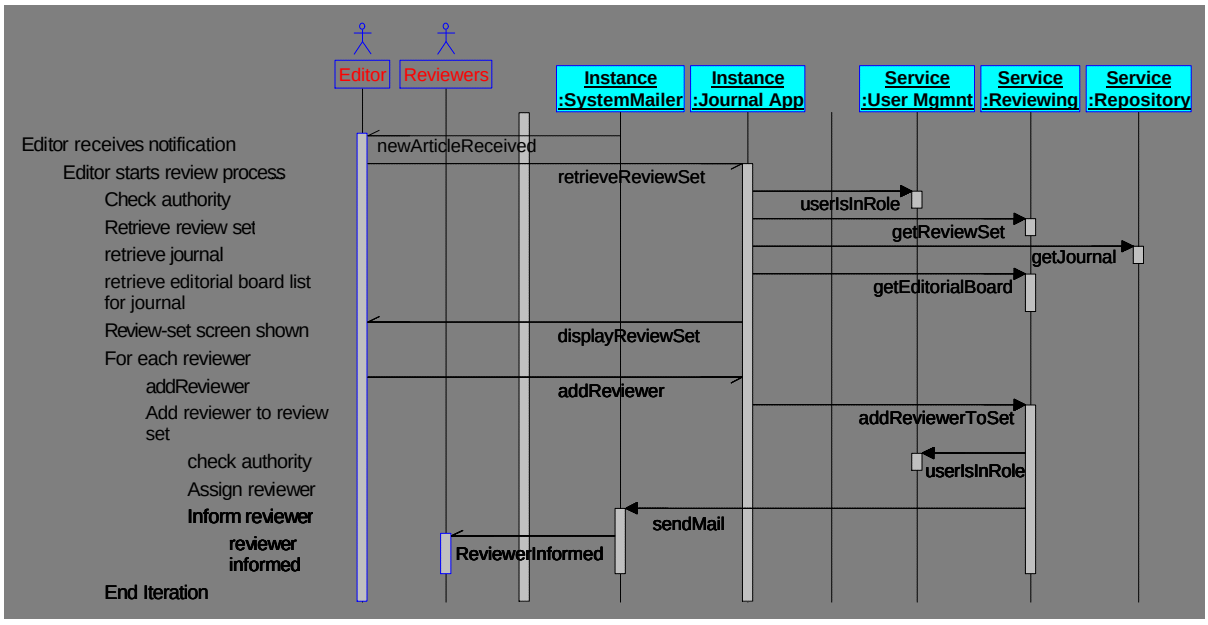

Figure 7: Processing of an article.

5. VALUE-ADDED PUBLICATIONS

Value-Added Publications (VAP[10]) provides an example, how SciX content may be used in vertical markets. The design principle for the SciX VAP stems from the gap in information requirements by the scholarly community on one hand and the Industrial community on the other. Normally scientists are funded through their research budgets to do background and literature studies as part of their normal research where as industrial practitioners have very limited resources for similar studies of scholarly work. Essentially industry practitioners require just-the-right information just in time. The business model of the SciX VAP is to create and publish articles in form of digests, reviews and summaries from the scientific content in SciX digital archives and making it available to industry practitioners to suit their specific requirements and thereby extend the reader base of scientific publications. The main difference between the VAP and other SciX applications is outlined below:

- The VAP target end-user is the industry practitioner that has different information requirements from that of researchers and scholars.
- The Operators of VAP are industry organisations, federations and associations, commercial information providers and special interests groups primarily consisting of industry people
- The VAP uses a different business model from SciX. This is because the VAPs are aimed at setting up commercial services, which will be centrally managed, and charged for (either directly or indirectly), whereas the core SciX systems will be largely self-organizing, free to use, and therefore ultimately widely distributed in order to spread costs over existing institutional infrastructures on a marginal cost basis.
- VAP moderated articles are specific to the AEC industry while SciX publications are more specific to the scientific and academic community
- VAP provide content management for collaborative auditing and versioning requiring private workspaces, locking for file checkin/checkout and editorial lifecycle management not just file storing
- VAP application can syndicate content to other VAP applications in a peer-to-peer network

5.1 Logical Architecture

The overview of the SciX VAP software architecture is shown in Figure 8 including its main modules and communication protocols. All the modules are implemented as three tier Web-applications and communicate via the HTTP protocol. For portability Java is the programming language of choice. The SciX VAP is a Java (J2EE) and XML based environment that relies on numerous open source software projects from the Apache Foundation (www.apache.org) e.g. Tomcat, Xceres, Xalan, Slide, Turbine and XML-RPC.

[10] The term VAS (value added services) used in the SciX Technical Annex and earlier reports was renamed VAP (value added publications) so that term “service” can be used only in the W3C meaning throughout this document.

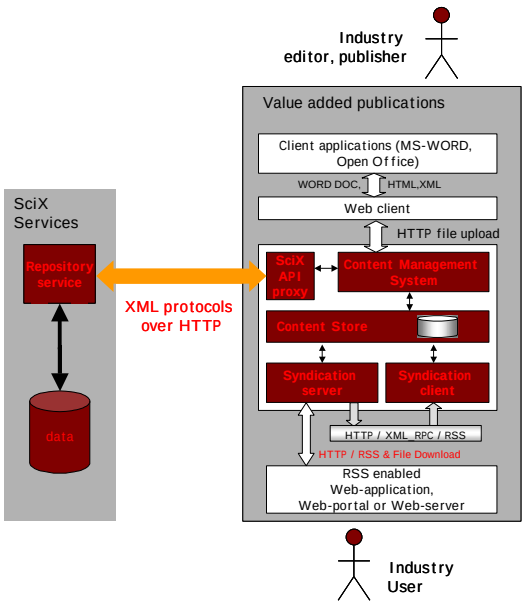

Figure 8 : The SciX VAP software architecture (may be considered a detail of the far right part of Figure 2).

5.1.1 Authoring VAP content

The Content Management System (CMS) provides the necessary service infrastructure for collaborative authoring and versioning, organising the content logically and accessing the scientific content in the SciX repository service. Also several authoring support services are available such as for bibliographic information, citation indexes and bookmarks into the SciX digital archives. The CMS provides its own user interface to interface with the SciX repository service that enable searching, browsing and downloading. Communication between the two systems will be by XML-RPC over HTTP/ and XML for metadata and bibliographic information retrieval.

5.1.2 Mechanism of syndication

The SciX VAP framework is based on a syndication model. The SciX VAP exists in a peer-to-peer network where the content delivery mechanism allows communication of content between separate SciX VAP and publication applications like Web servers. The content delivery system contains two modules, a syndication server for delivering content and a syndication client for receiving and storing syndicated content at the local host. Content delivery is subscription based and automated delivery between peers is controlled by a syndication schedule (e.g. when, what, where). Communication of content is based on the RSS syndication protocol. Servers will post RSS headlines of new and updated content to clients. Clients have then, the option to upload the actual content and store it within the CMS for publication to end-users or further processing by authors.

5.1.3 Interface between the SciX repository service and VAP

The VAP interface proxy maps the SciX repository service API and provides functionality for VAP users to:

- Do simple keyword search for scientific publications
- Do advanced search (Figure 11) i.e. to build detailed searches for scientific publications based on numerous parameters
- Store search parameters for repeated search
- Browse the digital archive (e.g. list of scientific publications, metadata etc.) by author, class and keyword
- Download scientific publication full text
- Download metadata for building citations and bibliographic notes
- Bookmark repository scientific publications locally

The interface (API) is implemented by calling the repository service exposed methods, using the XML-RPC protocol. The content (e.g. search results, metadata, etc.) wrapped in the XML-RPC response is described by a XML schema that is based on the underlying record, field description of the repository database.

5.2 Content Management System (CMS) Architecture

The content store is shared by all the software modules and is accessed via an API. It also maintains central information about users and access privileges.

Content within the SciX VAP is represented to users (the Web client) in a familiar fashion as a normal file system consisting of folders and files. Metadata is used to glue the separate information entities stored in the content store. Metadata is stored separately from the content and metadata access adapters will be explored to enable series of metadata structures to be implemented e.g. Dublin Core or MARC 21. In default configuration Dublin Core is used.

The centre of the CMS is the Apache Slide (jakarta.apache.org) CMS. Slide provides functionality to manage the central article repository, users, access permissions and locking of shared articles. This includes a logical repository structure consisting of nested collections and files similar to file systems, locking mechanism to implement chechout/checkin functionality for collaborative authoring, centrally managed access control to namespaces, collections and files. The VAP Slide store consists of two namespaces, a private shareable namespace where users store their personal articles and documents and a public namespace where syndicated content feeds and downloaded documents are stored. Both of these namespaces are physically stored in the OS file system.

The CMS API adds its own functionality on top of Slide including Version management, Workflow management and its own handling of metadata. Other author services available from the CMS API include user profiling, maintaining and organising bibliographic information, bookmarks and citation indexes.

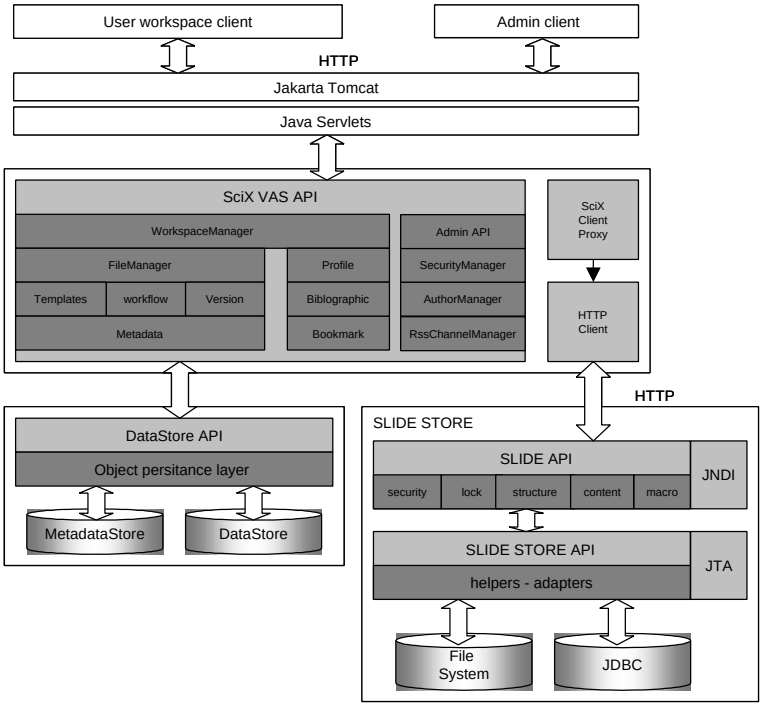

Figure 9: Content Management System Architecture

5.3 Syndication System Architecture

The overall architecture of the Syndication system is shown in figure 6b. It consists of three Web applications; the Management application, the Syndication Server application and the Syndication Client application. Servers and Clients operate in a P2P network and can schedule automatic updates of content to nodes in the network.

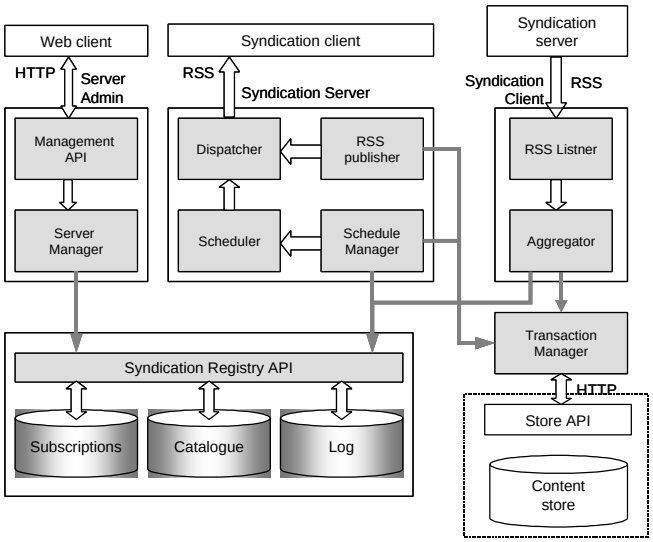

Figure 10: Syndication System Architecture

The Syndication server and Syndication client communicate by the XML-RPC request, response protocol. The Server and the Client implement an embedded in-process Web-server that accepts HTTP connections. The XML-RPC protocol handling is implemented by the XML-RPC code from the Apache XML-RPC project (xml.apache.org).

The Server Scheduler in cooperation with the Schedule Manager provides the automated mode of the Syndication process. The Schedule Manager runs periodically, at defined times and creates a job entry handle in a job queue if updates are available in the metadata content store. The Scheduler monitors the job queue and when it is not empty and time specified in the job timestamp matches, it starts a Dispatcher worker to process the job request. The Dispatcher uses the RSS publisher to retrieve the required information from the SciX VAS content store and formats the data into the requested RSS protocol version. It then connects to the syndication client and uploads the RSS channel.

On the Client side the RSS Listener, listens for XML-RPC requests from a server. On a request from the server, the client responds by calling an XML-RPC method on the server to upload the RSS channel or optionally cancel the request. The Aggregator can, besides persisting the RSS channel, automatically retrieve the actual content (e.g. digest, review etc.) being advertised by the RSS channel, is so configured, from the Syndication server and persist it in the CMS public repository namespace.

The Transaction manager is a shared component by the Server and Client. It handles all interaction with the CMS through the Content Store API.

6. GENERAL IMPLEMENTATION ISSUES

These issues relate to all applications and services and provide an overview of the generic implementation strategies to be adopted across all services and applications.

6.1 Access control

Access control will be implemented centrally within the User Management services through a role-based authorization system. Each user will be allocated a number of roles, which confer authorities on that user in relation to SciX functionality. All applications and services developed under SciX that require authorization services will be aware of the relevant roles as defined through the user management services.

Authentication services will be provided through Pluggable Authentication Modules (PAM)[11], which is the common approach used by most Unix/Linux applications, accessed for example through the PERL *Authen::PAM* module[12]. In the prototype system, users will be authenticated against a database of usernames and passwords managed by the User Management service, through a suitable PAM module such as pam-mysql[13]. However, this authentication mechanism may be replaced as required by a wide range of alternative mechanisms using the appropriate PAM modules[14]. Web-based applications will implement authentication using form-based authentication in the first instance. Authentication will be carried across a session by means of randomly generated session tokens, which will be stored using cookies for Web-based applications, and held directly by other applications. All SciX Web-service calls will be expected to receive an authentication token, which may be null in the case of non-authenticated calls.

The architecture will support:

- Application level access control. This control is user based and is subject to the requirements by different applications. If so requested, different users will have very different rights, depending on their roles (e.g. editors, people on the review board, authors and anonymous users). Application level control will restrict some functionality to some users.
- Service level access control. This control will prevent that a service would be used by just about any another service or application on the Internet, but just by services on the "white list". This list would contain identities of servers that are allowed to call services.

6.2 Validation and trust issues

It is intended that the SciX services should be self-managing as far as possible. For general services such as browsing, searching and paper submission, a self-validation scheme based on

11 See http://www.kernel.org/pub/linux/libs/pam/index.html
12 See http://search.cpan.org/author/NIKIP/Authen-PAM-0.14/d/PAM.pm
13 See http://pam-mysql.sourceforge.net/
14 For an extensive list, see http://www.kernel.org/pub/linux/libs/pam/modules.html

email addresses will be employed whereby users will register themselves and then confirm their registration by means of a token sent to their email address. This is a commonly used method for self-registration. An interface will be provided for users to re-register or recover they passwords if they have lost their passwords, etc.

For more critical functions, such as journal editorship, conference organizing, system administration, etc., manual registration will normally be required, to allow for human-based validation of users.

Facilities will be provided for digital signatures (based on X.509 certificates) to be stored in relation to articles and reviews to allow for higher levels of document authentication. These will not be compulsory, however. The implementation is expected to rely on OpenSSL[15] and the CSP (Certificate Service Provider) Perl module[16].

6.3 Error handling and fault tolerance

The pilots will not implement any proprietary fault tolerance system. This issue is orthogonal to the design of SciX itself and is therefore outside the immediate scope of the project itself. Should the publishers of the libraries, conferences or journals wish so, they could implement, on the server level, technologies like RAID disks, server clusters etc., addressed through well known Linux clustering approaches[17] or in the extreme cases (as may be envisaged in the case of advanced development of the Knowledge Management service for example) through approaches such as Beowulf clusters[18].

The applications and services, however, will use the logical information schema to handle the downtime of services on which they rely. Some of this downtime may be fatal - for example if the repository service is down, not much one can do with a digital library application. If however, the discussion service is down, basic functionality of the digital library (searching, downloading papers) will remain unaffected.

Service monitoring, using a package like Logsentry[19] will be a part of the service administration application and will be able to notify administrators about problems by email, instant messaging or SMS messaging. Since all XML communication takes place over HTTP, any commercial package for the monitoring of HTTP servers can be used.

6.4 Logging

Logging will be done at three levels:

- httpd server level; these logs are intended for administrative management of the server, log analysis will be carried out with 3rd party tools such as Webalizer[20] for statistical analysis of server logs, and Logsentry[19] for monitoring and alerting.

[15] See http://www.openssl.org/
[16] See http://devel.it.su.se/projects/CSP/
[17] See http://www.ram.org/computing/linux/linux_cluster.html
[18] See http://www.beowulf.org/
[19] See http://www.gnu.org/directory/security/misc/LogSentry.html
[20] See http://www.mrunix.net/Webalizer/

- Application level logging. The goal of this logging is to provide an integral, application specific view of how users use the application.
- Service level logging. Each service will maintain its own log file as well.

The last two log files will be created and used by the SciX code, but will be managed through that standard system logging facilities available in GNU/Linux, allowing for different levels of detail of event logging. Log file analysis will be managed through the Web Service Administration application.

6.5 Load balancing

Stress capacity testing of the prototypes show that on a 800 MHZ P4 class machine with 128MB RAM running Linux of FreeBSD, about 20.000 requests per hour can be handled without any serious degradation of response times. This is, by today's standards, outdated hardware. CUMINAD, that has a registered user base of 1000 persons, delivers up to 100.000 pages/month or 140 per hour, which is a fraction of what a server could support. Peak loads are caused by Web crawlers and robots, not by human users. They will have to be identified either by their origin and User Agent Strings[21] or through the detection of their behavior, which is very different to human behavior. We therefore do not believe that load is an issue at all. Academic digital libraries are not high volume sites such as shareware.com[22], cnn.com etc.

Should the need arise, however, the FGG has a grid of 32 machines that could take over the job. Using load-balancing software such as Zeus Load Balancer balancing can be done entirely transparently to the programmers and current architecture. Alternative solution is to simply distribute the services across more than one server which is what the architecture allows.

6.6 Search syntax

A uniform search syntax will be introduced across applications and services. It is an industry standard search syntax, first implemented by AltaVista and later by Google. The same syntax is used for both the end-user queries as well as in service-to-service or application-to-service interaction. Some examples of this syntax:

- car vehicle automobile searches for *any* of the words.
- "New York" car searches for 'New York' or 'car'.
- new +car searches records which do contain 'car' and may contain word 'new'.
- -used car vehicle searches records which contain words 'car' or 'vehicle' but not 'used'.
- n?w searches 'now' and 'new'. ? stands for any one character.
- car* searches 'car' and 'cardinal'. * stands for any number of any characters.
- *car searches 'vicar' and 'car'.
- *car* searches 'vicar', 'cardinal' and 'car'.

[21] http://www.robotstxt.org/wc/active.html
[22] members of the consortium were involved in setting up this one and necessary expertise on high load applications exists in the consortium.

- " car " or "\scar\s" searches 'car' and not 'vicar' or 'cardinal'. White space is represented by /s or a space.
- +email:yahoo searches for 'yahoo' in the email field only.
- -email:yahoo searches records which do not have word 'yahoo' in the email field.
- =lastName:Brown searches records with lastName field exactly equal to Brown.
- /lastName:Brown searches records with one line of lastName field exactly equal to Brown.
- {price} < 2000 searches for items with price field less than 2000.

The last example shows how quite complex query expressions may be built. They may be constructed with the aid of a query builder form (below):

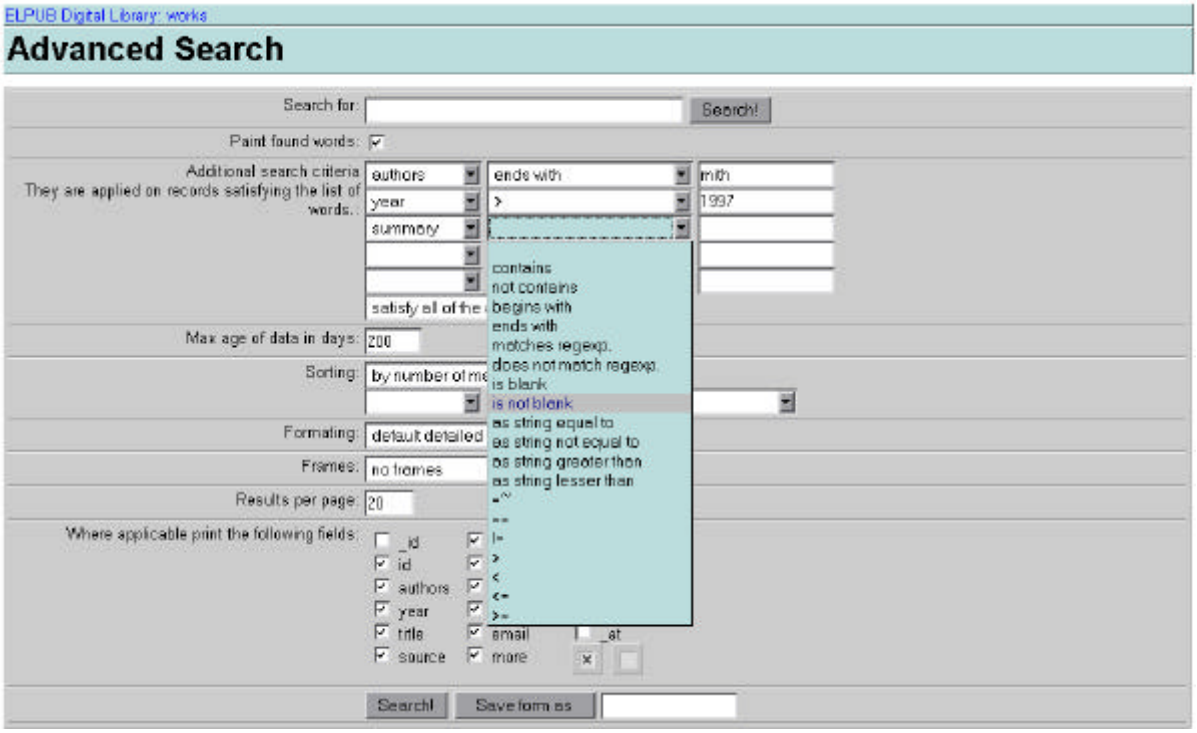

Figure 11: Query builder form.

6.7 Timeline

The Technical Annex (Gantt Chart) defines that the prototypes start to become available between month 9 and month 15 and that final implementation work should be completed by the project month 21 and in some cases project month 24.

The prototype work started on schedule, building on top of code existing before the start of the project e.g. cumincad.scix.net. This application is running on the Web with over 5000 papers and a registered user base of 1000 persons and is a monolithic, one of a kind application.

The implementation work is incremental and is taking these steps:

1. Extract re-usable parts of the existing code used to implement CUMINCAD.SciX.net.

2. Split the code into modules along the lines delimiting applications and services in the Figure 2. The result is offering little new functionality to the end user but is paving the way for future development.
3. Add new services such as discussion, collection, recommendation, rating etc. (some also based on existing code at LJU in FGGI) and use all of the above in setting up smaller, private prototypes. Develop internal logic of the services but use private interfacing mechanism, not Web services. Get user feedback.
4. Test various knowledge management tools and libraries and use them for automatic classification, clustering of papers and similarity measurement.
5. Add XML APIs to the repository service. Do initial integration with the VAP module.
6. Add XML APIs to other services. Integrate them using XML protocols.
7. Add OAI PMH adapters and integrate them with the repository service.
8. Implement any other services still missing (e.g. reviews).
9. Implement other applications (journal, conference).
10. Implement non-browser applications and interfaces to citation software if the project resources allow for it.

At the time of writing (early May 2003), this timeline has been completed up to step 4 and work on steps 5 and 6 is under way.

7. CONCLUSIONS

This document has set out a flexible architecture that can be used to support different models of publication and addresses the requirements defined in SciX Deliverables 1 and 8. It is also flexible enough to support future requirements, which may emerge as the result of the business process re-engineering, and experience of the system in use.

The architecture is open and modular, based on a three layer Web services approach, and makes use of industry open standards where possible. A number of core SciX applications are provided, and the architecture will allow additional applications to be created either by the SciX consortium, or by more or less knowledgeable third-party application developers and programmers. Also the data managed by the pilots will be available to 3rd party harvesting using the emerging OAi standard.

The modular nature of the SciX architecture will allow developments in Web service standardization to be adopted as they emerge in the future, in areas such as authorization, further developments within the OAi, and the gradual development of a global Web services infrastructure. The availability of SciX to the open-source community, coupled with the open architecture will help to ensure such further developments beyond the life of the SciX project itself.

A key aspect of the SciX architecture is to demonstrate that by defining a set of modules implemented as Web services, different applications can be created very rapidly and adapted for changing requirements. The architecture presented in the document will support such a demonstration.