IST-2001-33127

SciX

Open, self organising repository for scientific information exchange

D9a: Architecture Synthesis

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

Access: public
Version: 1.0
Date: January 31, 2003

Executive summary

This document presents a high level synthesis of the deliverable D9 "Overall Architecture Report" with a concrete architecture. It is derived from D9, the requirements and user scenarios addressed in deliverable D8 and the process diagrams of the deliverable D1. It presents main use cases, business objects, software tools, libraries and the physical architecture of the pilot that is being developed in the SciX project. In greater detail these items are described in the much longer D9 report.

The actors and use cases include the activities related to the managing of a digital library and an electronic journal and other quality assurance mechanisms that may be placed on top of it.

The business objects include the definitions of objects such as a library item (work), library item collection (series), person, discussion, journal and topic. Where appropriate, the data structures comply with the Dublin Core metadata standard.

The software tools present the tools that are being used to create the pilot running on the Internet. In addition to several free digital library tools which may be used to implement the library item database, a generator of web services Woda is used to implement all objects - the works as well as others that are needed to support the workflows briefly presented in the workflow section.

In addition Java and J2EE will be used, particularly in the development of the Industry Value added services (VAS).

The physical architecture documents how the activities performed by the system are physically supported by software. How this software will be physically distributed over the Internet and how will it work together. The proposed architecture allows for the distribution of the main objects across the Web and their implementations in different technologies, so that the functions of searching, archiving and quality assurance may be separated and may (technically) each be regarded as a separate Web service. To connect to various archives, the Open Archive Initiative's PMH protocol is used to access remote information. All other service-service communication relies on HTTP protocol and XML encoding.

Release history

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	Current version of D9a

The SciX Consortium would like to acknowledge the financial support of the European Commission under the IST programme.

Table of Contents

1. ARCHITECTURAL PRINCIPLES

This section describes a number of "architectural principles" that would provide advantages if applied to the logical architecture of the software supporting the SciX system. Some of these principles may not be satisfied immediately, but it is important that the architecture of the system is able to accommodate them in the future. In the following, the term "separation" should be taken to mean logical separation in software terms rather than necessarily physical separation.

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 functions to be included, left out, added, or replaced relatively easily in any particular implementation.

A widely accepted principle, particularly promoted in this arena by the OAi (Open Archives Initiative) and SPARC[1] (the Scholarly Publishing and Academic Resources Coalition), is the **separation of data storage from service provision**. It is also important that the repository itself (i.e. the minimum required functionality) is kept fairly simple, making it easy to put in place a repository initially to support a simple collection of articles.

The SciX architecture not only separates the data storage and service provision but is identifying a number of business objects such as annotation, review, recommendation, rating etc. which, if combined in different ways, can provide various kinds of applications on top of a digital library. One such application, for example, is an on-line academic journal.

From an academic point of view, 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. In addition, publishers of conventional journals address issues such as editorial quality (sometimes), distribution (or access), establishment of priority (date of submission and publication), storage and archiving (through submission to major libraries). It has been pointed out[1] 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. In this situation, it will be necessary for a journal to link to another 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. This may be done by taking a copy of the article, or it could be done very effectively through the use of digital signature using PKI (Public Key Infrastructure).

[1] 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)

Given the pace of change in electronic publishing, with developments 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 be 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 journal hosted on a SciX server. 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.

Various standards exist for the definition and representation of metadata (e.g. Dublin Core, MARC 21). Although there are efforts to create convertor software to move between metadata standards, e.g. d2m (Dublin Core to MARC) from the Nordic Metadata Project, it will be desirable to allow multiple metadata records to be held for a single article. This implies a separation between the repository and the metadata storage.

Additional “knowledge management” (KM) services may be provided on top of the repository. This may include 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 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. An example of such an environment might be an authoring tool used to create industry digests within the value added services area.

2. SELECTED USE CASES AND BUSINESS OBJECTS

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 and
- support for a refereed journal.

All these publication outlets 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 objects are needed. And if properly developed configuring them to support all of the above is quite feasible. The concepts are shown graphically in Fig. 1.

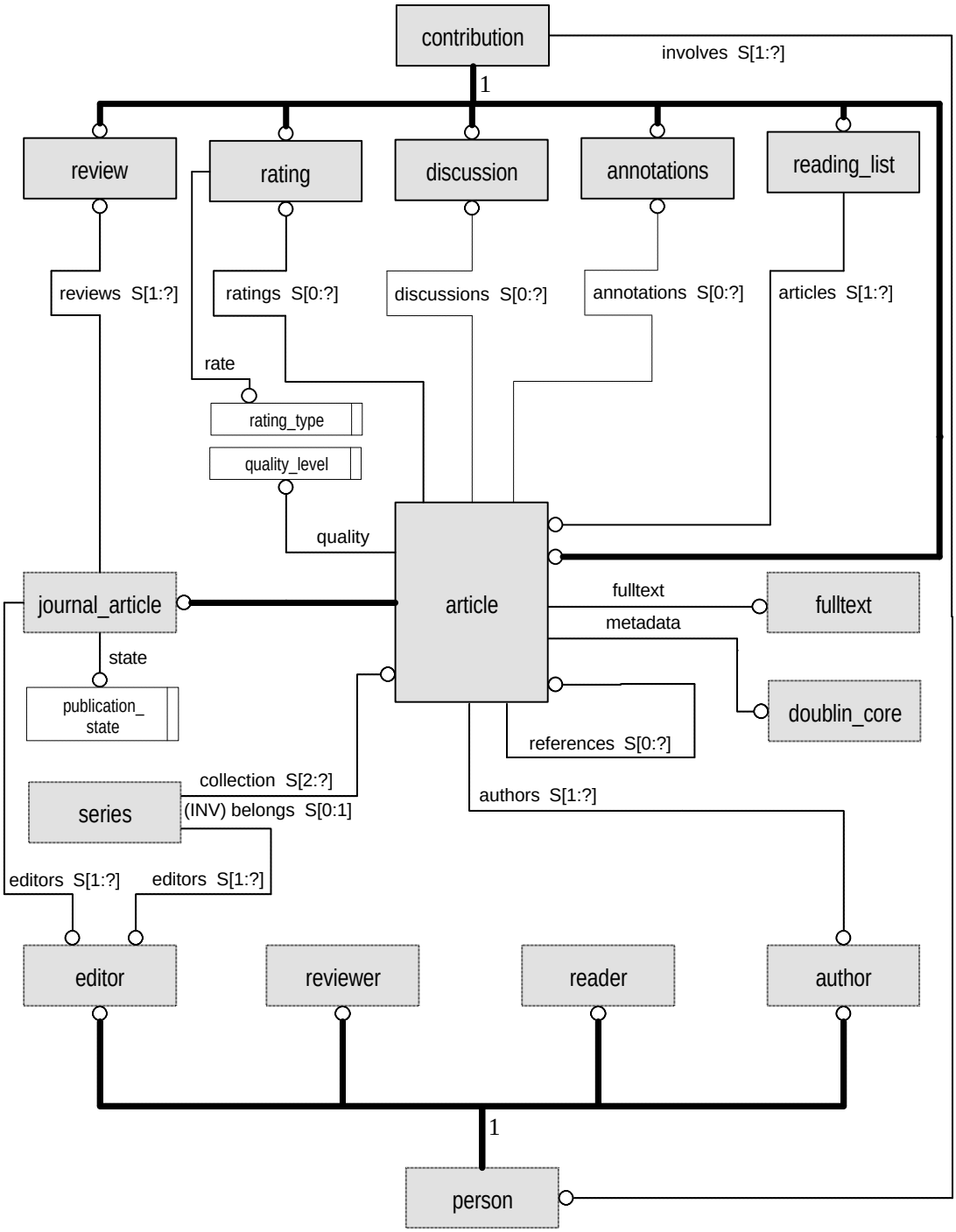

Figure 1: Main concepts to be supported by SciX pilots, drawn in ISO EXPRESS-G language.

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 end user GUI as well as a service-service level interface. Several services are combined into applications, such as an on-line journal. OAI-PMH protocol is used to connect to any compliant metadata services. XML protocols are used for all other interactions.

3. LOGICAL ARCHITECTURE

A modular architecture is proposed for the SciX system, allowing objects 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 Fig. 1 is not implemented in a monolithic relational database application but rather by a numbed of services; the collaboration among them can be established at runtime.

The idea of separating the various services while keeping them interoperable is shown in Figure 2.

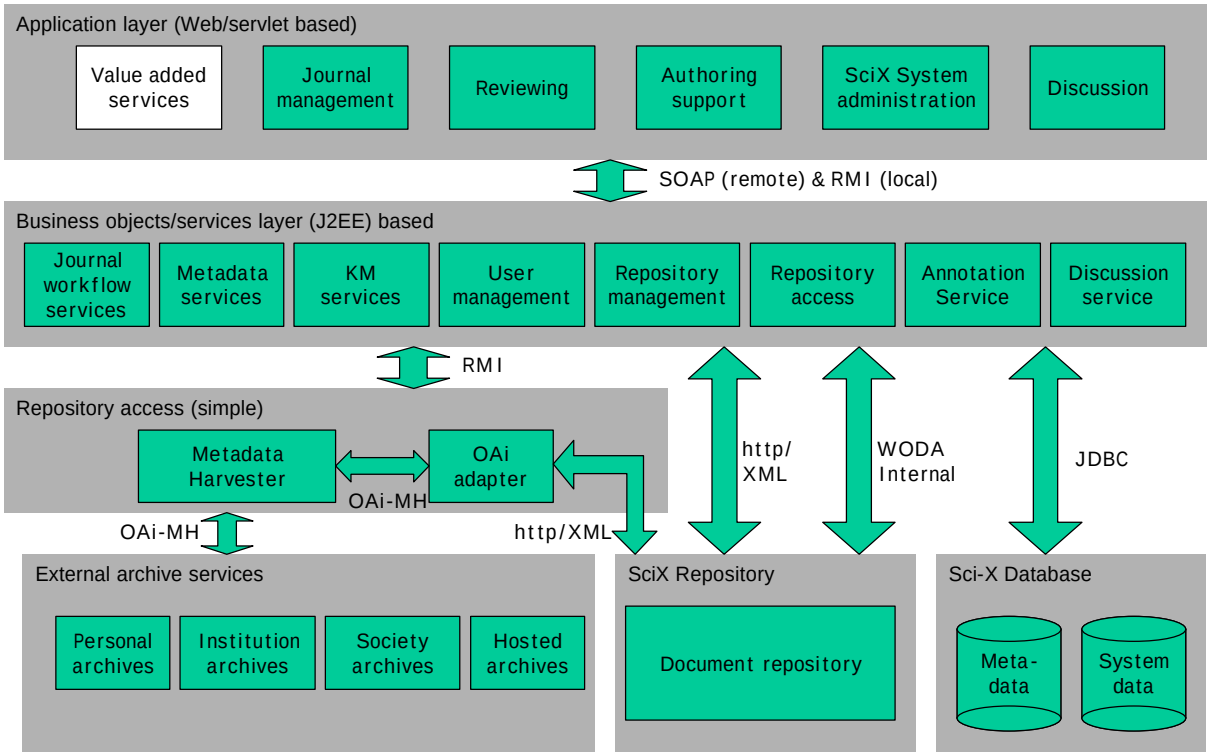

Figure 2: Logical architecture of the main SciX system

The following table relates applications to the services that they are built on to help clarify the relationship between the service and application layers.

Services	Journal Workflow	Metadata	Knowledge Management	User Management	Repository Management	Repository Access	Annotation	Discussion
Applications								
Value Added Services		X	X			X		
Journal Management	X			X				
Reviewing	X			X		X	X	
Authoring Support		X	X		X		X	
System Admin				X	X			
Discussion							X	X

Table 1. Relationship between Applications and Services.

Figure 1 represents a logical architecture that separates storage from services and services from applications. Services are the building blocks that may be used to build applications to suit the requirements of different user communities.

In many cases, elements of the architecture may be combined together on a single server or even in a single software implementation. This will be the case to some extent in the initial implementation of SciX, which will build on the existing infrastructure of the partners, developed through, for example, Woda, Rent-a-db, CUMINCAD, ITcon, and the ECPPM 2002 infrastructure. If the services happen to run on the same server local RMI access may be more efficient to implement than SOAP. Relevant interfaces are to be provided to allow inter-working between alternative implementations at each of the levels.System Architecture

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 Services are treated separately and described in section 5 of this report.)

3.1 Annotation Services

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

3.2 Knowledge Management Services

This will provide enhanced abilities for searching out information and will be based on knowledge management techniques including indexing, clustering and citation analysis.

3.3 Repository Management

This should provide for 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 ...

3.4 Journal Services

Journal services support the workflow in 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 reviewer, 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 ...

3.5 User Management

While anonymous access is important because many people do not like to be bothered with access rights and passwords, any active use of the services requires used identification and other related actions such as: enter & maintain user, list users, manage system access, check user authority, create audit trail ...

3.6 Metadata Services

They provide 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 ...

3.7 Discussion Service

The functions include set up & maintain thread, add entry to thread, remove entry from thread, list threads, search threads, select thread.

4. VALUE-ADDED SERVICES(VAS)

4.1 Logical Architecture

Figure 3 shows an overview of the SciX VAS software architecture, the 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 VAS 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.

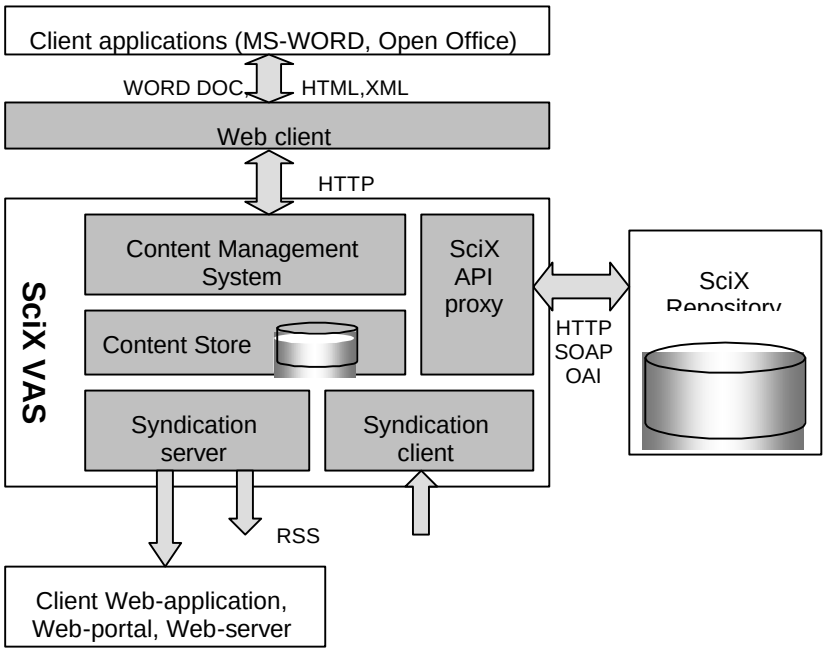

Figure 3 : The SciX VAS software architecture

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

The SciX VAS framework is based on a syndication model. The SciX VAS exists in a peer-to-peer network where the content delivery mechanism allows communication of content between separate SciX VAS and publication services. 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.

A proxy design pattern was chosen to implement the interface with the SciX service both to isolate the SciX VAS from the actual SciX API and as well as to minimize impact from API changes.

4.1.1 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 VAS 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 pluggable 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 Slide store consists of two namespaces, a private shareable namespace where users store their 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.

The architecture is described in fig 4.

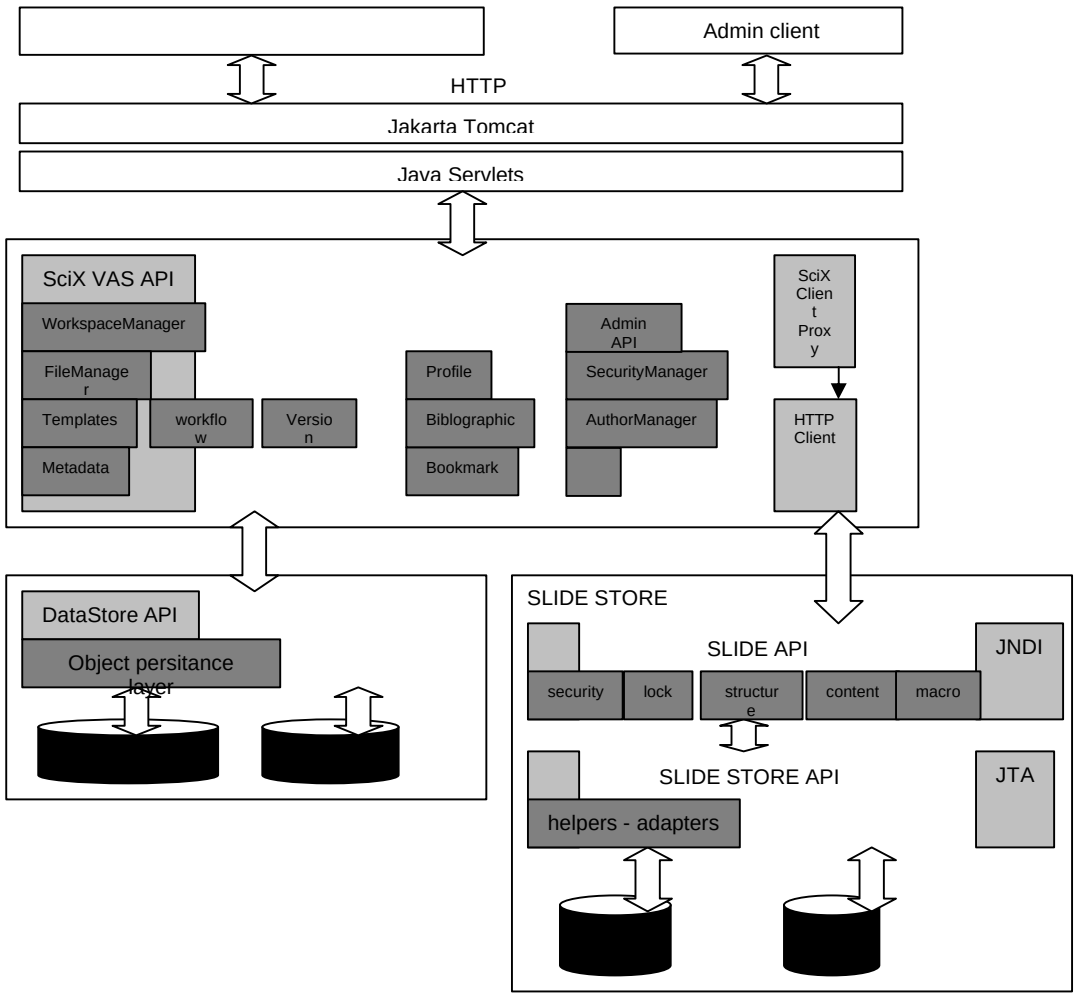

Figure 4: Content Management System Architecture

4.2 System Architecture

- This section lists the separate procedures, which must be carried out by the system in order to achieve the desired result:
- produce and store digests, summaries and reviews
- centrally organize authored content such as digests
- enter & maintain bibliographic list
- enter & maintain citation list
- generate standard citation for inclusion in article
- download SciX data (to local machine)
- enter & maintain search criteria
- create & maintain bookmark
- enter & maintain business partner
- enter & maintain schedule for automatic content delivery
- accept and store content syndicated from SciX vas

- schedule delivery of content to SciX vas or presentation system
- notify author of use of material.

4.3 Syndication System Architecture

The Syndication system architecture provides:

- Automated push capability for syndicating content from server to client
- A server-application and a client-application that can exist side-by-side or as independent standalone applications for matching user needs.
- Server and client share data store (back-end)
- Syndication by subscription and catalogue objects that provide control over syndication of content at specified times to clients
- Authentication and security mechanism in transactions between server and clients
- Use of open protocol standards HTTP, XML-RPC and RSS
- User customisable data access layer for different back-ends
- Easy web interface for administration of the syndication process

Figure 4a. Shows the overall architecture of the Syndication system. It consists of three web applications the Service management application, the Syndication Server application and the Syndication Client application. Servers and Clients operate in a P2P network and can schedule updates of content over the network.

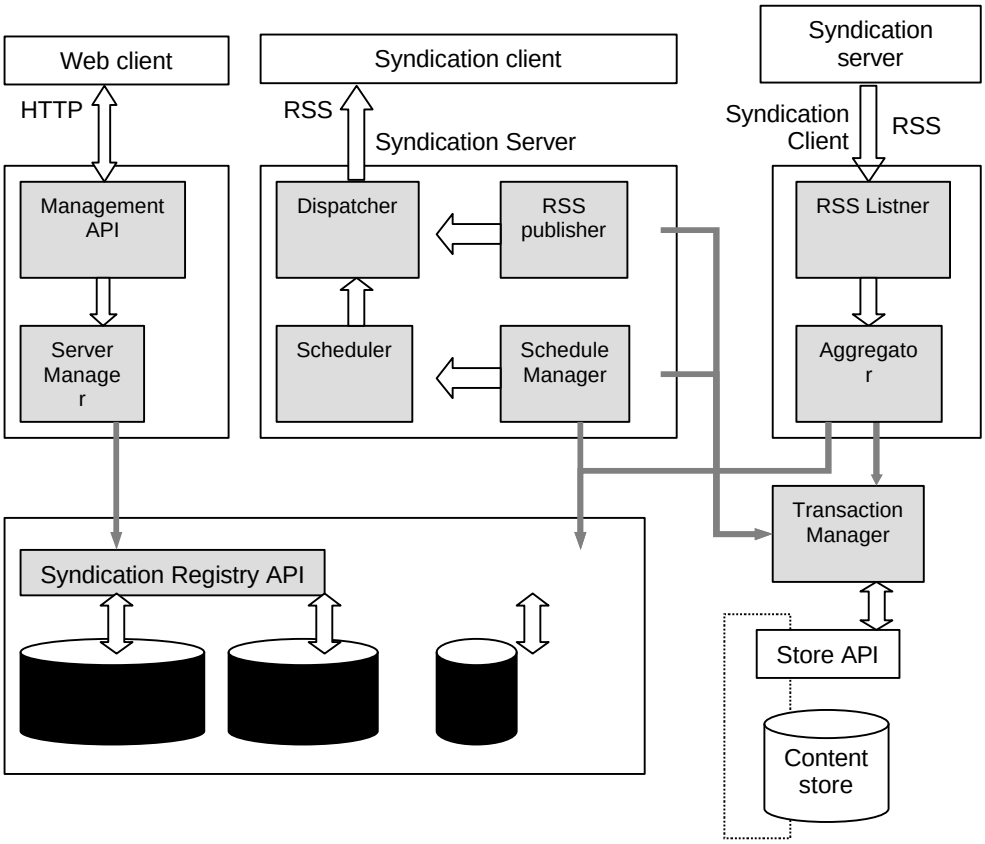

Figure 4a. Syndication System Architecture

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 Scheduler Manager provides the automated mode of the Syndication process. The Schedule Manager runs periodically at defined times and creates a job entry handle in the 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 timestamp matches, it starts a Dispatcher worker thread 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, connects to the syndication client and uploads the RSS channel.

On the Client side the RSS Listener listens for XML-RPC requests from the server. On a request from the server, the client responds by calling an XML-RPC method on the to upload the RSS

channel or optionally cancel the request. The Aggregator can, besides persisting the RSS Channel, retrieve the actual content being advertised by the RSS channel 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.

5. CONCLUSION

The presented architecture is being implemented in two SciX services under development, the http://cumincad.SciX.net/ and the http://itc.SciX.net/

The reader should bear in mind that these web sites are currently work in progress, laying the basis for further development.

6. APPENDIX – ARCHITECTURE OF WODA-BASED SERVICES

Report D8 described in detail some of the technologies and tools that may be used to implement the SciX pilots. The pilots developed in SciX will adopt the OAI-PMH standard and XML related Web standards. In essence, to the SciX pilot as a database application, more precisely a Web Oriented Database application.

6.1 Web-oriented Databases

We define the main features of a Web oriented database as:

- A Web browser is the main interface for creating, using and managing the database. All basic database functions, such as adding, editing, deleting, querying information can be done using a web browser. Database administration and maintenance is also performed through a browser.
- Database engine does independent user authentication or uses server-side authentication. It is important to allow for anonymous users. Even if users are anonymous, they should be able to protect their data from being edited by someone else.
- Export and import of data to and from PC applications such as Excel or Access should be enabled.
- The database should have similar user interface and search syntax as other Web services in order to get quickly accepted by the visitors on the WWW.

Advanced features of a web-oriented database include:

- Support for multimedia information, such as pictures, drawings, printable documents, and other binary data.
- Include software agent technology, which can learn the needs of the user.
- Tie in with other Internet technologies, such as email, chat, and on-line conferencing.
- Enable rapid development of database application, include report, and form generation.
- Support internationalisation, national character sets, collate sequences, and multi-lingual user interface.
- Support SQL and ODBC data storage.

Since 1995 FGGI and TUW have been developing a “web oriented database system” called WODA (http://www/ddatabase.com/). WODA is a CGI application written in the Perl language. Its design goal was to create a smart and simple tool, which would allow very rapid creation of small to medium size database applications that could be used and managed using Web tools. WODA is tightly integrated with Web technology, supports multimedia contents (such as full text articles) file uploads, full-text searches and includes software agent technology.

WODA can talk to the user in English, German, French, Spanish, Russian, and Slovene and may be easily translated into any other language.

6.2 Use of Woda in SciX pilots

The reasons for this decision include:

- good familiarity with the tool;

- no need to rely on 3rd party tools or libraries, both technically or legally,
- several other services using the same infrastructure; this would make the maintenance of the SciX services much less time consuming and would require only minimal additional work;
- last but not least, the tool is very good and is used in hundreds of application world-wide. It is listed as #1 (most relevant) in section Computers > Software > Databases > Web Connectivity of Google and Open Directory Project.

What makes the use of Woda particularly appealing for pilots like SciX is that unlike most other Web database tools, Woda is an application generator. A typical table managed by Woda requires some between 3-10KB of code to define the data structures, related logic and formats and a GUI template (another 5-10KB which is common to several tables required in an application). Based on the definition of the service logic, about 120 different dynamic page types are generated that address all the needs of searching, browsing, adding and managing the data as well as XML connectivity. All end user as well as management activates may be performed through a Web interface.

Woda includes a database driver - current implementation uses raw filesystem for the storage of the data. An SQL driver is under development.

6.3 Interfacing to and from Woda

Several Woda services may be combined to create a complex Web application. They may be combined using:

- calls, if the services are on the same machine
- HTTP links to establish Web-links between different servers
- HTTP calls to move data between services on the internet.

The formats for the data exchange are HTML, XML and CSV.

- HTML is used, for example in a report/subreport situation, where one table needs to make a query into another table for example to display all papers a user has written. The result may come in properly formatted HTML.
- XML is used if raw data is needed,.
- CSV is used for communication with Misrosoft Office tools Excel and Access.

Woda is a self documenting tool and presents its interfaces in both human and machine readable formats (XML).

6.4 Other relevant tools

For advanced management of the repository, text mining etc. freely available libraries from CPAN will be used and integrated into the business logic of the definitions, which are written in Perl as well.

6.5 Anatomy of a service

This section presents how, technically, the pilots will be implemented. It starts with an anatomy of a single service and then elaborates how several services will work together on the Web.

The figure below shows the various pieces of code and how that may work together to implement two services that happen to be hosted on the same server, Cumincad and SciX. All the boxes in one grey box communicate using Perl subroutine calls - each of the white boxes with a title row (works, people) may be regarded as one program running in one process.

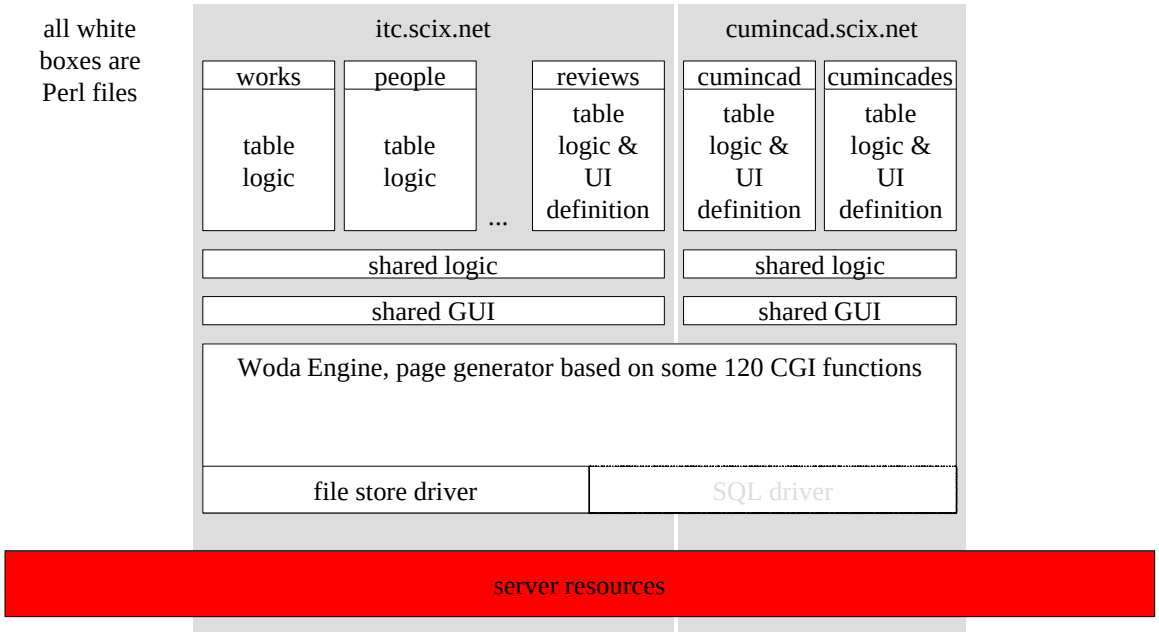

Figure 5: Software architecture as built on top of operating system, Apache and Woda.

Each of the two grey boxes in the previous figure represent two services which are part of a broader picture - a network of services shown in the next figure.