

**Testbed for Interoperability of eBook
Metadata (TIME)
Final report**

Submitted to:	Balviar Notay JOINT INFORMATION SYSTEMS COMMITTEE Hazel Woodward JOINT INFORMATION SYSTEMS COMMITTEE Ebooks Working Group
Version:	Final Report
Date:	24 April 2006

Rightscom Ltd
Lincoln House
75 Westminster Bridge Road
London SE1 7HS
UNITED KINGDOM
Tel: +44 20 7620 4433
www.rightscom.com

1	Introduction.....	1
1.1	Participants.....	1
1.2	Process	2
1.3	Acknowledgements	5
2	Technical Architecture.....	6
2.1	Functional Requirements	6
2.2	Solution outline.....	8
2.3	Software Stack	10
2.4	Custom Software Components	19
3	JISC Technical Architecture References	23
3.1	Software Stack	23
3.2	Standards.....	23
3.3	Unimplemented but with some relevance	24
4	Transformation methodology	25
4.1	Summary of COAX capability.....	25
4.2	Mapping schemas	27
4.3	Generating XSLTs.....	27
4.4	Pre- and post-processing	27
4.5	Transformation scope.....	28
4.6	Transformation limitations	28
5	Technical conclusions.....	29
5.1	Semantic loss: relative strength of metadata schemes.....	29
5.2	Data quality issues: errors.....	29
5.3	Data quality issues: “variant schemes”	30
5.4	Data manipulation: concatenation and string analysis.....	30
5.5	Use of conditional rules	31
5.6	Use of default rules	31
5.7	Limitations of XML/XSLT technology	32
5.8	Alternative approaches	32
5.9	EPICentre observations	33
6	Table of data issues	35
7	Suitability of various metadata schemes for ebooks	40
7.1	Standards for the expression of usage rights in e-books.....	41
8	Recommendations	43
9	Appendix: The Contextual Ontologyx Architecture abstract metadata framework	
	44	

1 Introduction

This is the final report to accompany the TIME testbed of ebook metadata records submitted to JISC in April 2006.

1.1 Participants

The key participants in the project were EPICentre and Rightscom. Book Industry Communications and consultant Helen Henderson also contributed to the project.

1.1.1 Rightscom Ltd

The company provides knowledge of and solutions for the operational, technical, commercial and regulatory processes needed for successful and profitable information commerce. With expertise in rights network infrastructures, rights licensing, business modelling, metadata processes and digital rights management technologies, Rightscom offers a global perspective on information commerce issues.

Rightscom Ltd had two key roles in the project: development of the testbed itself, and overall project management.

1.1.2 EPICentre

EPICentre (The Electronic Publishing Innovation Centre) is a joint research group between the Department of Information Science at Loughborough University and Library and Information Services at Cranfield University. All its members have a research interest in electronic publishing, and expertise in various aspects of this topic. Appropriate members are chosen to take part in different projects according to the expertise required.

In the TIME project, EPICentre members took responsibility for testing the alpha and beta versions of the system, to judge their utility to librarians and to check that conversions between the different systems (DC, ONIX and MARC) had been accurately performed by the system. They were also able to note the quality of initial assignment of metadata by publishers. In addition, one EPICentre member at Cranfield, Paul Needham, had a role in the TIME project as a consultant on matters relating to the metadata harvesting (OAI-PMH) interface of the test bed – iteratively testing and validating outputs and providing feedback to the programming team.

1.1.3 Book Industry Communications

Book Industry Communication is an independent organisation sponsored and part-funded by the Publishers Association, Booksellers Association, the Chartered Institute of Library and Information Professionals and the British Library to promote increased efficiency in the book and serials supply chain – physical and electronic – through the application of standard processes and procedures and e-commerce.

Book Industry Communications facilitated the requirements workshop and provided the analysis of metadata formats included in this report.

1.1.4 Helen Henderson

Helen Henderson is an independent consultant working in the area of Internet publishing and strategic business development. Her clients include publishers, subscription agents, systems houses and information providers.

Her main role in the TIME project was to liaise with the publishers over data provision and to provide background context information to inform the development process.

1.2 Process

1.2.1 Draft requirements

During the desk research, we identified a number of documents and other sources of information on metadata requirements for e-books. None contain complete requirements. A synthesis was developed, focused on needs of libraries.

1.2.2 Validation workshop

The workshop was held on 24 March 2005. It was led by Brian Green of BIC.

Those attending came mainly from a range of university libraries, and for the most part had either systems or bibliographic responsibilities. One delegate was from a public library with experience of lending e-books, and a library systems supplier was also represented, along with a representative of a major e-book aggregator. One publisher attended.

The major focus of the workshop was on standards. The analysis described above was summarised for presentation to the workshop, and the attendees asked both to respond to the issues it raised and to add points of their own.

No radical disagreements or major additional requirements were presented during the workshop. The standards proposed were endorsed at the workshop, and although additional useful information was obtained no major issues that might affect the direction of the project or its objectives were identified.

Some additional points were identified by the libraries, many focused on practical issues of implementing the inclusion of e-books. Key issues identified included:

- Maintaining the integrity of the library catalogue
- Bringing ebook records up to the same standard as other catalogue entries
- How will users get from a catalogue entry to the ebook itself? This is important, as catalogue entry increases usage
- Managing catalogue entries for collections of ebooks (purchased as a collection)
- Converting from UK-MARC to MARC-21
- How should titles acquired as part of consortium membership be handled in the catalogue?
- Ebook inclusion should fit with a strategic view of e-resources
- Cataloguing non-textual material

However, the workshop did identify that there are many different definitions of an e-book and that these should be taken into account: working with only a narrow definition will cause problems later. In particular, it was pointed out that it can be very hard to determine where ebooks end and learning materials begin.

1.2.3 Metadata records

Records were obtained from publishers:

- Oxford University Press
- Taylor and Francis
- Cambridge University Press
- OCLC

A total of 1886 records were received, in DC, MARC and Onix format.

1.2.4 Semantic mapping

“Prototype” records for each record type were initially developed by EPICentre and these were analysed. This gave a target for the semantic mapping the team carried out as part of the transformation engine development.

The different record types were analysed and preliminary decisions about mappings made: as the project developed and comments of the quality of the initial mappings was received from the library and information experts involved in the project, these were refined and the prototype records were set aside. This process continued through each stage of testing and refinement.

1.2.5 Technical approach and architecture

The technical approach to transformation based on that described in the proposal was refined and finalised. A technical architecture was developed and a set of tools chosen.

1.2.6 Rescoping

With JISC’s agreement the feature-set of the system was revised to allow additional resources to be devoted to the core transformation system. The key changes were:

RSS: we found no requirement for this among users, mainly because of unfamiliarity with the concepts and purpose of RSS. We believe that in the future the benefits of RSS will be more widely understood.

Athens or Shibboleth authentication: this did not prove necessary in view of the small and controlled user group for the test phases and the development overhead would have taken valuable time from the core technical work.

SRW-U: we had hoped to provide an SRW-U interface; however, major contributors to the core software used (Fedora) promised SRW-U for some months (this was one of the main reasons for selecting Fedora for the main repository and processing functions) but their software component was not released in time to be used in the project (it has since become available). This has meant we were unable to evaluate the SRW-U software during the lifetime of the project. This was frustrating, but the overall scale of the project made it essential that we should rely on third-party software for some of the functionality, and in this area we have suffered accordingly. In all other respects, we have found the Fedora/MySQL/Kowari/Orbeon web-service based architecture we have built very effective.

OpenURL: although we found no immediate demand for OpenURL, we would have liked to develop an OpenURL interface. However, the additional time spent on the core technical development has meant that we did not have sufficient time to do this, especially in the light of the Fedora team’s commitment to ensuring an “Open URL Access Point” available in the near future (source: <http://www.fedora.info/download/2.1.1/userdocs/server/features/serviceframework.htm>)

Metadata formats: No LOM records were obtained, and so testing of these was not possible, although a mapping of LOM was created to allow for outward transforms. An additional mapping of Dublin Core Qualified was added to the original set of schemas, and towards the end of the project a mapping for the OAI-compliant version of Dublin Core was also produced.

We should emphasise that as far as we are aware, we have not developed the system in a way that will prevent any of these features being incorporated in the future.

1.2.7 Development

The first iteration of the transformation engine was developed and tested with a range of sample records. The ontology to support transformations was enhanced and refined.

Further changes were made following the alpha and beta tests.

1.2.8 Alpha test

An alpha test was carried out.

A set of sample test records was transformed using the system and analysed by EPICentre.

In both the alpha and the beta testing phases, the process adopted by EPICentre staff was to print out the records for a sample of input files in the various formats, the corresponding COAX files (alpha test only), and the corresponding output files in the various formats also. In the beta test, conversions from COAX back into the original input format were also examined. By inspection of the different versions alongside each other the fate of each data element on conversion was ascertained.

In the alpha test phase, a research assistant employed at Loughborough University and directed by Fytton Rowland of EPICentre, Victoria Rae, conducted a thorough test of all conversions between the three real-life metadata systems (DC, ONIX and MARC) and the intermediary format COAX, noting all cases where the system seemed to have wrongly assigned a metadata type in its conversions into and out of COAX. These results were returned to Rightscom and helped their staff to make necessary amendments to the system in the development of the beta test system.

In parallel, Rightscom also carried out detailed element-by-element analyses of a similar number of records.

1.2.9 User interface development

A small number of librarians and other ebook experts were consulted on the requirements for the beta-test functionality and user interface. This was specifically focused on the beta-test, and would not necessarily be applicable to a production service intended to serve the needs of a much wider community.

The majority of the requirements arising from the process were incorporated into the beta-test, although it was not possible to include all.

The user interface developed allows the source and transformed records to be viewed by users and output for incorporation into other systems if desired.

1.2.10 OAI-PMH interface

An OAI-PMH interface has been developed to allow harvesting from the testbed.

1.2.11 Beta test

The beta test was carried out by EPICentre.

A full set of records was transformed using the system and analysed by EPICentre.

In the beta test phase, several EPICentre members (Fytton Rowland and Ann O'Brien at Loughborough and Simon Bevan and Paul Needham at Cranfield) carried out similar tests of the conversions, looking at them from a librarian's or user's point of view rather than a computer systems point of view. Paul Needham also provided 'on-demand' solutions to a number of queries relating to the manipulation and exposure of MARC records in various formats, as well as co-ordinating the testing of TIME outputs at Cranfield. On this occasion the input and output formats were examined while the intermediary COAX format was not. All possible conversions were checked,

including conversions from and back to the same format (via COAX); hence there were nine conversions altogether: DC→DC, DC→MARC, DC→ONIX, MARC→DC, MARC→MARC, MARC→ONIX, ONIX→DC, ONIX→MARC, ONIX→ONIX.

Rightscom staff also carried out a similar range of checks, including the analysis of the intermediary COAX format where errors were found.

1.2.12 Final release

Final changes were made to the transformation system as a result of the beta test. A CD-ROM which would enable anyone to install the JISC-TIME server application and data as a stand-alone application on a PC was distributed to JISC's Programme Manager. Full installation and usage instructions were included.

Two sets of CDs were delivered: one suitable for a PC with a minimum memory of 512MB, the other for a PC with a minimum memory of 1.5GB. These effectively constituted a complete Linux server, preinstalled with all the JISC-TIME application components and databases, running as a single Windows application.

The package consists of two components

- The VMWare Player application: This is a freely downloadable software package that allows users to run virtual machines on their PCs.
- The JISC-TIME server: These are the files used by the player application to run the server.

The virtual server runs completely independently, as a single application – the only software needing to be installed on users' machine's is the VMWare Player application.

1.3 Acknowledgements

We would like to thank the publishers who provided sampler data , and also the JISC Ebooks Working Group for its support and encouragement.

2 Technical Architecture

2.1 Functional Requirements

A primary functional requirement of the Testbed system is to support ebook cataloguing.

Rightscom committed to create “a substantial pilot catalogue of eBook metadata, building and proving a test bed for managing the catalogue so that different users with different requirements can **access the test bed and retrieve metadata in their preferred format** is at the heart of this proposal.”

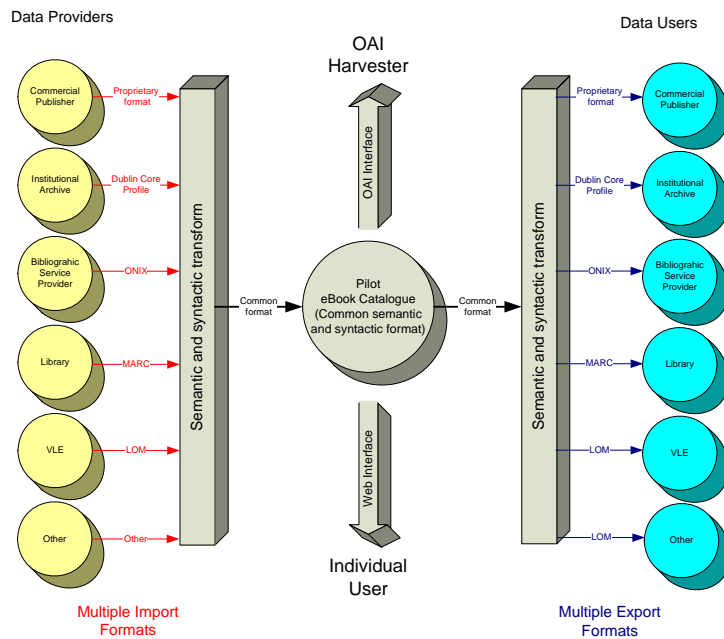


Figure 1 A 'data formats' view of the proposed ebook metadata testbed

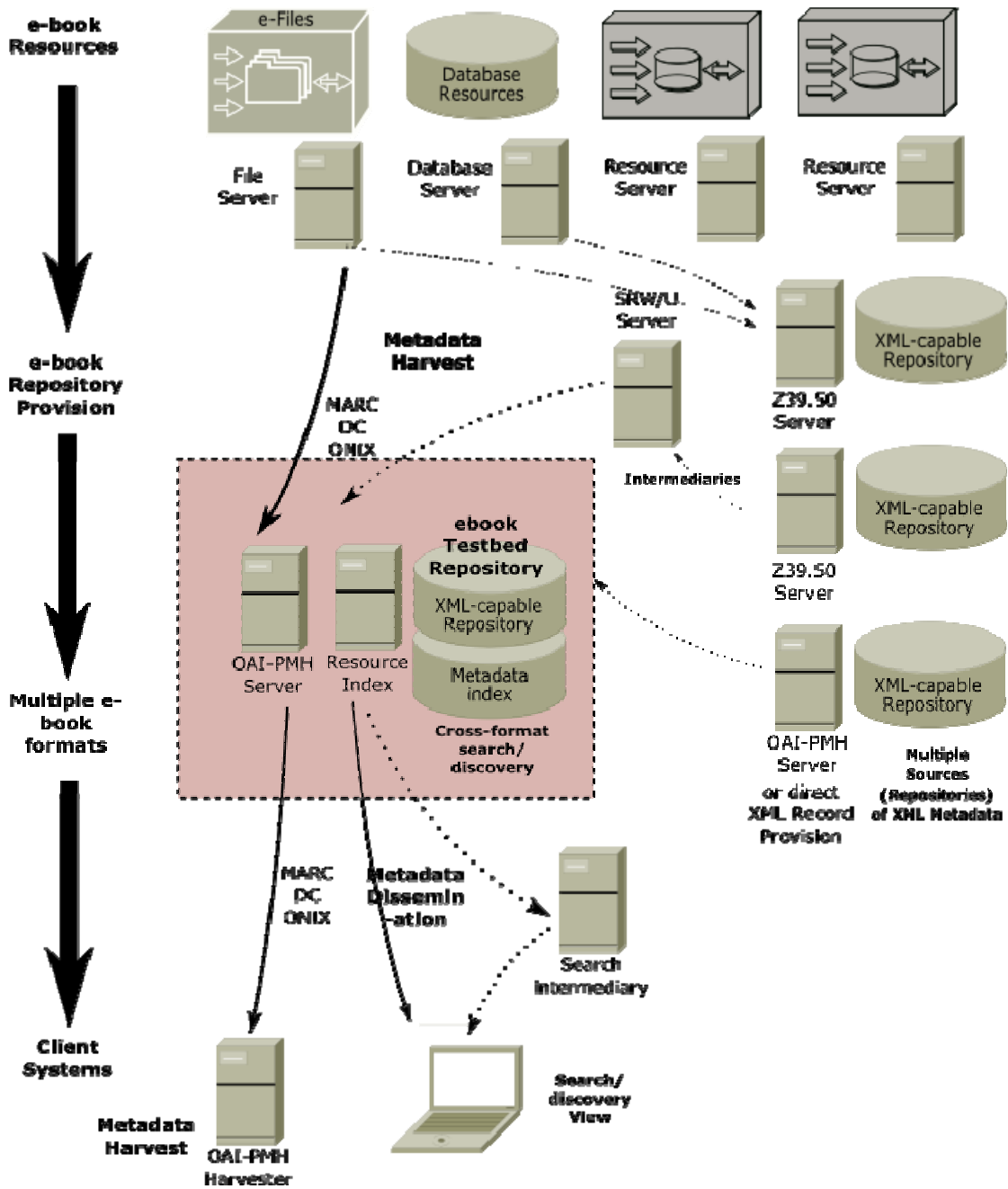


Figure 2 Illustrative scenario of metadata testbed

(NB: Resources on Provider systems and any harvesting they do by the testbed are out of scope)

The formats and protocols for delivery, both inward (ingest) and outward are intended to reflect the JISC IEA: SRW/U or Z39.50 for search/retrieve, and the OAI-PMH for harvest).

This requirement essentially defined the system boundary protocols and formats available for the testbed systems build. In order to build the system as described in the response there was early on a recognition that there would be an impact on the technical architecture and indeed the information architecture to support search, discovery and catalogue management. addition to the definition and mapping of input and output formats in XML for transformation development.

The diagram below shows the resulting system scope.

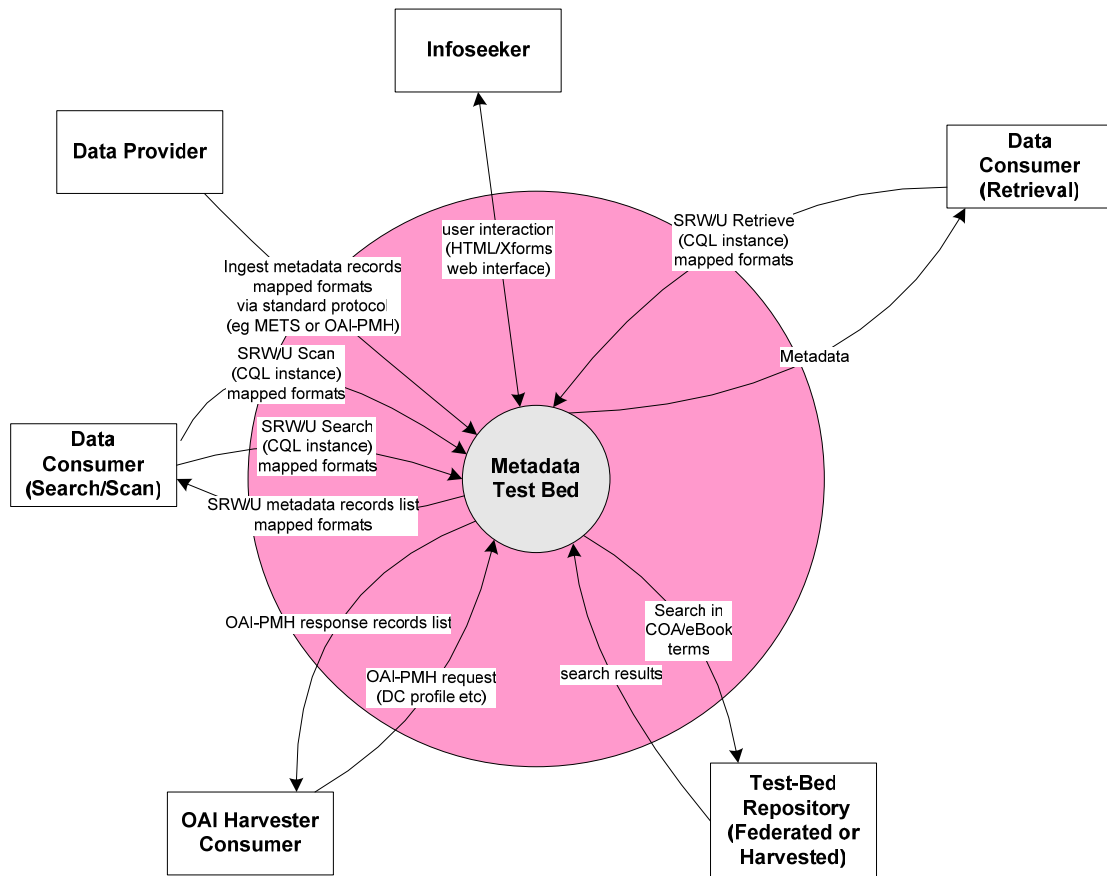


Figure 3 – Context Diagram showing test bed core system scope

2.2 Solution outline

The short time frame and minimal budget available for development against the scope mandated a COTS (component off the shelf) integration approach rather than a self-built, ground-up development.

Desktop research was conducted to find open-source software components available to meet the following system requirements:

Requirement	Solution
Accessing records through several sources in different formats (at least four: (ONIX, MARC XML, LOM and DC).	Ingest into an XML-based repository using a native ingest API of the repository, with the potential ability to harvest from existing repositories using OAI-PMH
Data Storage and Access	Provide a mechanism to index records ingested into a repository in the various different formats, and then to federate queries across them to provide the basis of data access.
User interface to allow data entry and maintenance	The user interface was intended to be built around the ingest mechanism of the chosen repository. The fact that contributor records were sent as single submissions of plain files meant a text-based console solution was chosen to fulfil this requirement.
System (database) interfaces to allow simple search and simple retrieval	For human access, to enable both remote and local maintenance, a web browser forms-based solution was chosen. Care was taken to ensure these forms and their interrelationships were readily configurable external to the implementation code to allow a high degree of customisation as the project proceeded. For machine-based access, an OAI-PMH interface was provided.
To output records in a common format that can be used in library catalogues and virtual learning environments, and the ability to export selected metadata records for delivery to third party users in at least four XML formats (ONIX, MARC XML, LOM and DC).	On request for a record in a certain output format, to transform each ingested record into a common format, into which all input and output formats had been mapped.
The data requirements were to maintain a "critical mass" of data for community testing, allow several contributors for ebook records (at least 2,500 records, with at least 500 each from each of at least four sources).	An underlying storage and retrieval solution scalable to reasonable record volume, together with an API (Application Program Interface) that would shield the developers/integrators of other components from the complexity and detail of the retrieval repository mechanism itself.
Standards support where possible following the JISC TA guidelines; specifically OAI-PMH for metadata harvesting, ZING-based distributed search (SRW/U v1.1 and Z39.50 back-end support), RSS feeds, OpenURL and Authentication/Authorisation via Athens or Shibboleth (although management of this is out of project scope).	A service-based architecture, where functionality is composed of interactions between 'loosely-coupled' services, rather than being realised through a monolithic and difficult to extend software solution.

2.3 Software Stack

2.3.1 Overview

To meet these various requirements, a layered software stack was identified consisting of an integration of distinct 'best of breed' components. It was understood that the components comprising the stack should be open source if possible, and be capable of running on the greatest reach of hardware and operating systems. This stack was then used as the framework for building all specific functionality.

All the components use the common base platform of:

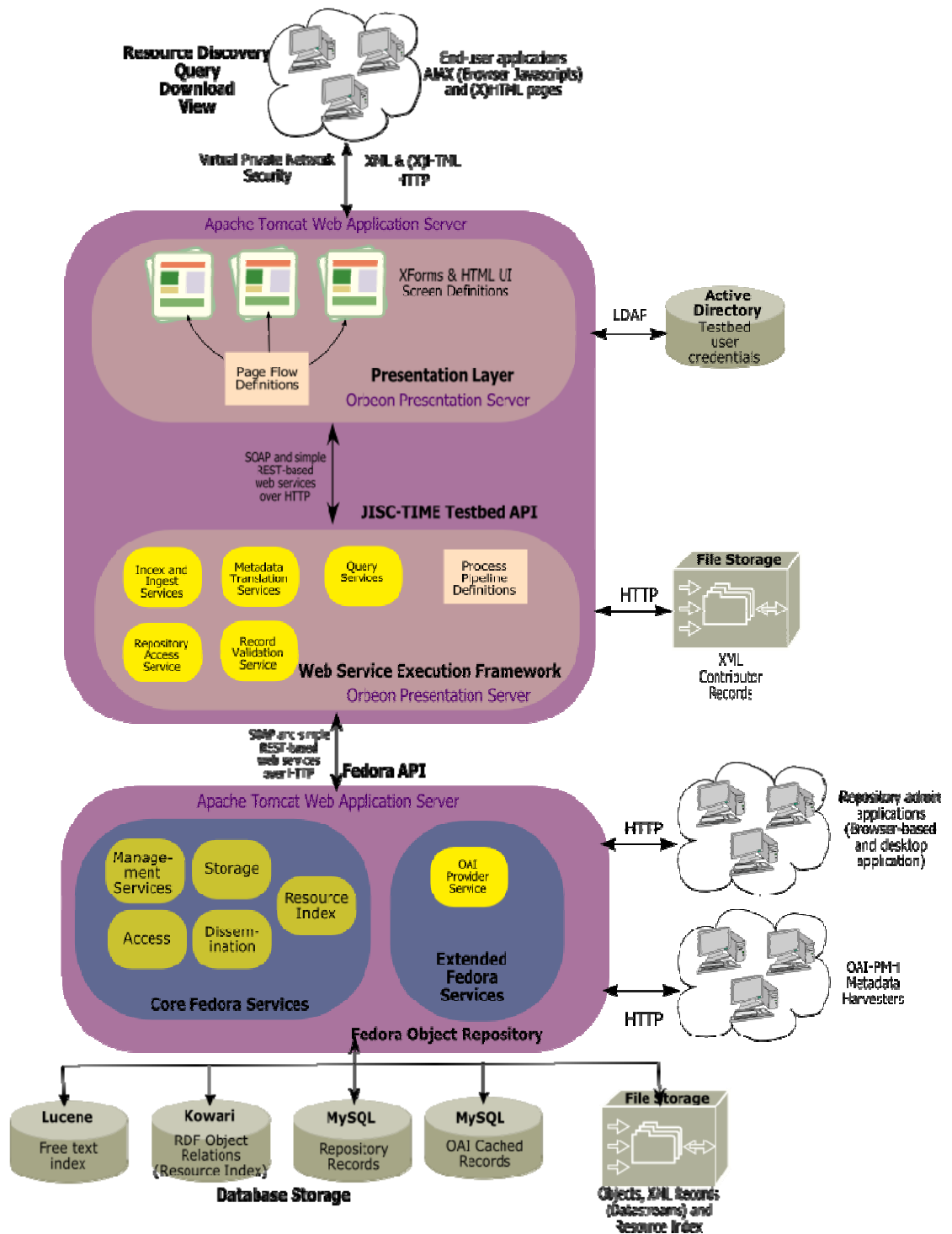
Java runtime (Version 1.4.2_08)

This therefore enabled cross-operating system compatibility, and the system has been tested successfully on the following operating systems:

Microsoft Windows 2003 Server Standard Edition with SP2

Microsoft Windows XP Professional with SP2

Linux (CentOS 4.2 , an enterprise-grade distribution and Red Hat clone)



2.3.2

Figure 4 – Architecture Diagram Showing Component Stack

2.3.3 Web Service Execution Framework

For maximum flexibility of deployment possibilities, scalability and ease of ongoing configuration management, the decision was taken to not only ensure code was executable via cross-platform Java virtual machines, but also to use a web service infrastructure, a ‘service oriented architecture’. This meant that COTS components were integrated using their web service interfaces, and all new functionality was built using a design of discrete granular components that each exposed a web service.

The various series of execution of web service operations, ‘service orchestration’, that defines specific functionality was then defined using the declarative process

definition language pipeline. Evaluation of these pipeline process definitions was then executed by the Orbeon Presentation Server (OPS) run-time engine.

The OPS version was a beta release of v3.0 when the project began, which has several stability and functionality issues; a migration of the Testbed was completed towards the latter stages to use the production release of v3.0.

2.3.3.1 WEB SERVICE IMPLEMENTATION LAYER

Web Service calls (REST or SOAP-based) are handled within the OPS as an underlying XPL pipeline, that takes the service location ('endpoint'), operation name and parameters, authentication information, performs appropriate character encoding and checks. The required service is executed and the results encoded as appropriate and returned to the caller.

2.3.4 Core Repository Services and eBook Resource Object Model

The Repository Services considered as 'Core' are at a basic level the provision of a storage and retrieval mechanism. Using a combination of desktop research and prior experience, the choice of the Fedora was guided by the following advantages over competitive approaches and COTS products:

- All functions of Fedora, both for object behaviours and for the repository as a whole, are exposed as web services, making the repository exceptionally well-suited for use within a (web) service-oriented architecture. Such an architecture allows new functionality to be added with the very minimum of disruption to existing services, many of which are core to the correct functioning of the repository.
- Fedora defines a powerful object model underlying all contents of the repository, irrespective of whether those contents identify physical artefacts or just a conceptual entity. The model allows not only repository objects to be linked to other objects using a controlled vocabulary (using the increasingly widespread W3C Resource Description Framework standard for specifying semantic links), but also allows multiple representations, or 'views', of objects to be defined as Fedora 'Datastreams'. This model was exploited for multiple metadata representations of ebook resources: a single Datastream was defined against ebook for each output metadata format, along with a Datastream for the Contributor format.
- Object identity may be managed using a 'persistent object identifier', or 'PID'. This facility would be useful for future audit and access control requirements, as well as being a reliable, persistent identification scheme for future use with external resolution services, such as OpenURL resolvers.
- Content may be locally-managed or refer to external content. This is pertinent to the Testbed environment, in that the repository aggregates the metadata from multiple sources, but does not itself hold a copy of the ebooks.
- 'Dynamic views' or representations of an object are achieved when a Datastream is populated by some functionality (residing behind a web service endpoint) 'on-demand'. In this way, these Datastreams do not themselves contain data, but act as placeholders or 'interface definitions' applying to an object. This facility was exploited to define a consistent object definition for each ebook resource, yet enable the production of ebook metadata in the target format to be invoked only when requested, thus lowering the overall server load, a factor especially important for large-scale deployments.
- Library and institutional repository-oriented standards support; on inception of the Testbed project, Fedora community development was working on a beta release of the version 2.1. This was expected to deliver the following standards

support: a OAI-PMH metadata provider service, a SRW/U interface support service, and an OpenURL resolution service. Unfortunately the SRW/U and OpenURL services were not ready for the Testbed release, resulting in their omission from the final Testbed release. However, they are now available should they be required, and the OAI-Provider service was included in the Testbed release.

- Fedora comes bundled with desktop and browser-based management tools, allowing end-user access and manipulation of the ebook objects, metadata, formats and transformation functionality. This functionality is subject to configurable fine-grained access control mechanisms (through association with XACML policies).

2.3.5 Application and Service Execution Environment

Both the Orbeon Presentation Server and Fedora execute within a Java 2 Enterprise Edition (J2EE) -capable application server environment, providing resilient, secure and horizontally-scalable capabilities to the Testbed.

Fedora has been tested on the readily-available open source Apache Tomcat v4.1 application server. We understand that Fedora may not function correctly on different application servers.

Orbeon claim to have tested OPS within all the major J2EE application server vendors. The OPS Service Implementation Layer and the OPS Presentation Layers are in fact separate architectural entities, although they utilise the same underlying OPS engine, deployed into the same instance of Tomcat, version 5.0). (However, this Tomcat instance is separate from the instance running Fedora 4.1).

The application server configuration, together with the database instances, may be run on a single machine, given a certain minimum specification, with acceptable performance.

An alternative minimum server configuration was set up for the beta test release as follows:

Server 1: MySQL Dedicated Database Server

Server 2: Fedora, Kowari, Tomcat 4.1

Server 2: Orbeon Presentation Server, Tomcat 5.0

Server 3: Active Directory and VPN-based Security

Host OS (all machines): Windows Server 2003 Standard Edition

However, for reliability and security considerations as well as performance with larger data volumes and a greater number of users, consideration should be given as to alternative deployments, such as moving the OPS/Tomcat 5.0 to a separate server or utilising clustering and load-balancing the servers.

2.3.6 Persistent Storage

2.3.6.1 RELATIONAL DATABASE: MYSQL 4.1

Certain core Fedora services and the OAI Provider are backed by a relational database (RDBMS). For simplicity and good performance, the MySQL was used. However, any JDBC-compliant database including more scalable, enterprise-level ones such as Oracle, may be used instead.

2.3.6.2 METADATA DATABASE: TRIPPI CONNECTORS AND KOWARI

Fedora requires storage for its 'Resource Index', the network of object relations ('triples') underpinning the Fedora object model.

There are several physical storage implementation options available. Fedora is shipped with a ‘driver’ component, ‘Trippi’, which has ‘connectors’ available for three leading contenders: Kowari (the default and preferred option), Sesame, and Jena. For the Testbed implementation, the team has remained with the Kowari-backed option.

Kowari is a massively scalable, transaction-safe database purpose-built for the storage, indexing and retrieval of data in triples. It is shipped with java libraries for compatibility with W3C RDF (and OWL ontology) standards, meaning it is a good choice for use as a database for managing metadata.

Kowari has a SQL-like language for querying the database, called ‘iTQL’, a language gradually converging towards the W3C query language for RDF databases SPARQL. iTQL queries and the Resource Index are extensively used within the Testbed application.

2.3.6.3 FILE SYSTEM STORAGE

The primary persistent store in Fedora is the file system. The databases – relational and RDF triples – are both “operational” in the sense that they provide indexed storage services for query and data retrieval, but the ‘primary’ file system storage allows each of these operational stores to be recreated on corruption or migration of the system.

This arrangement allows for resilience, robustness and portability of the Testbed metadata and identifiers.

2.3.7 Testbed Digital Object Model

The flexibility of the Fedora repository architecture is realised through the interactions of behaviours, implemented in code and connected through bindings to web services, with an underlying object model.

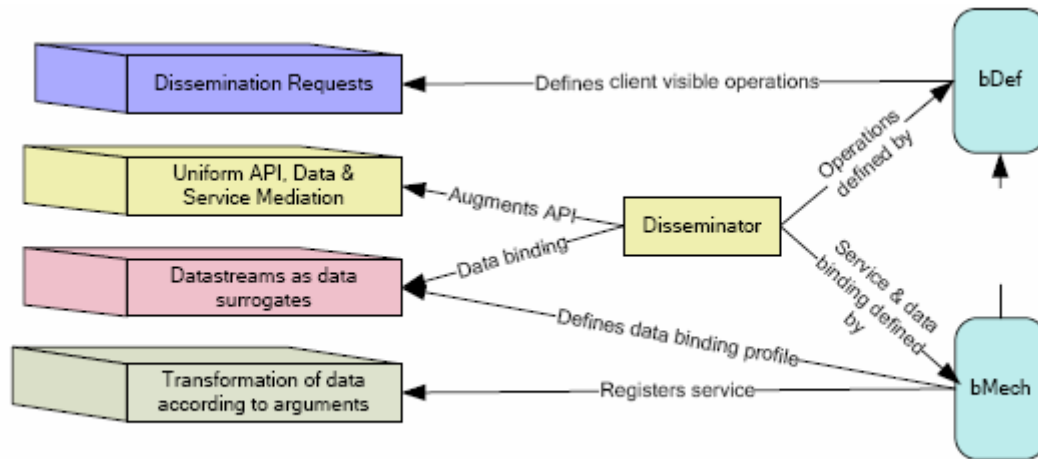


Fig 5 – The Fedora ‘Disseminator’ model binds objects to software behaviours

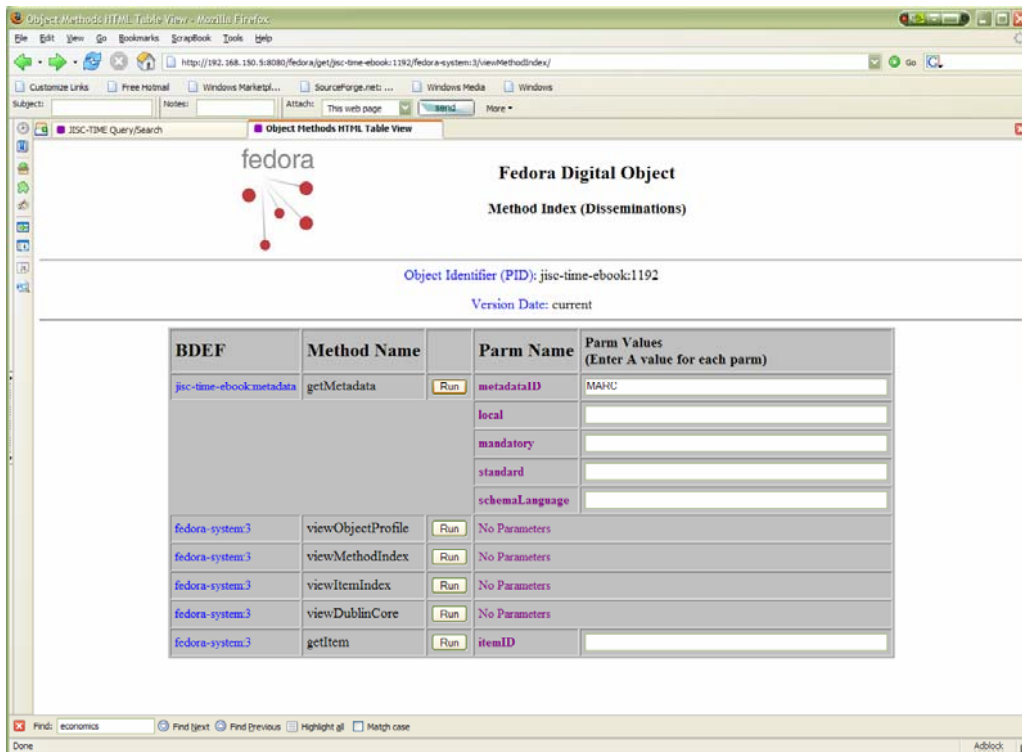


Fig 6 – The Fedora web administration interface showing method bindings for an example contributor e-book record. Administrators can run transforms directly from this interface as illustrated by the MARC example.

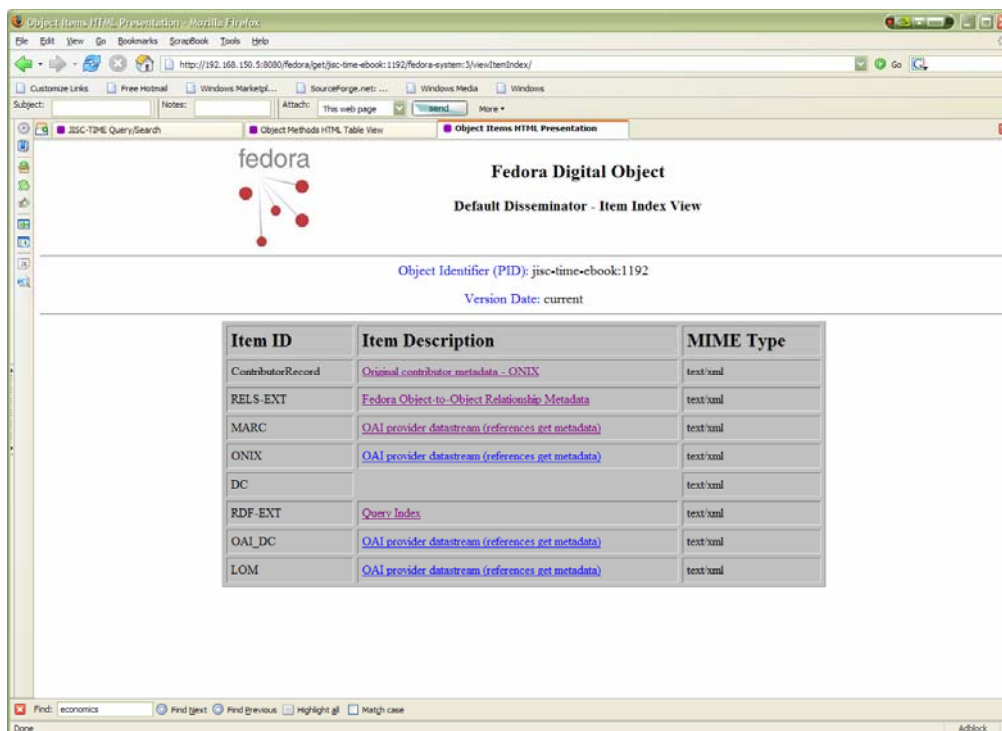


Fig 7 – The Fedora web administration interface showing the datastreams implemented for each example contributor e-book record. Administrators can view the record data in each format directly from this interface; the disseminator relation automatically invokes the correct transforms.

This object model expresses the relations between digital objects ‘registered’ with (ingested into) Fedora, using a controlled vocabulary.

These relations are binary, consisting of <subject predicate object> ‘triples’, where in Fedora both the subject and object are digital objects. Adherence to the W3C RDF standard affords interoperability with other efforts emerging from semantic web standards adoption, such as ontology reasoning and tool support.

In the Testbed, this controlled vocabulary expressed as triples acts as the ‘ontology’, or ‘triples schema’ for the ebook metadata repository.

The Testbed ontology is relatively trivial, and consists of two Datastreams per ebook digital object: the ‘RELS-EXT’ Datastream provided for every Fedora object, and an additional ‘RDF-EXT’ datastream.

The property vocabulary in the RDF-EXT ontology coincides exactly with the Dublin Core element set, in RDF, as shown in the example in Fig 7. This was an intentional choice: users engaged in search would be familiar with the meaning of Dublin Core. However, it does illustrate the important point that the query-search page (the first page presented to the user in the wizard-based application), may relatively easily be replaced with a particular subset of the scheme-neutral COA terms, where the subset would be chosen for resource discovery means.

Note: the Testbed ontology is separate concern from the COA JISC-TIME ontology which was ultimately used to generate the XSLT metadata transforms. This ontology is not represented within the Testbed object model itself.

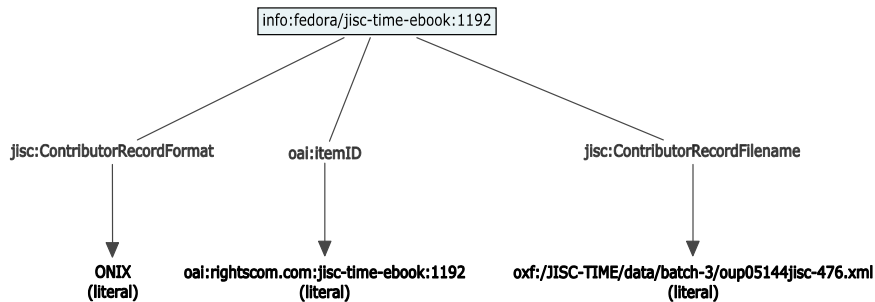


Figure 8 - RELS-EXT Datastream for sample ebook

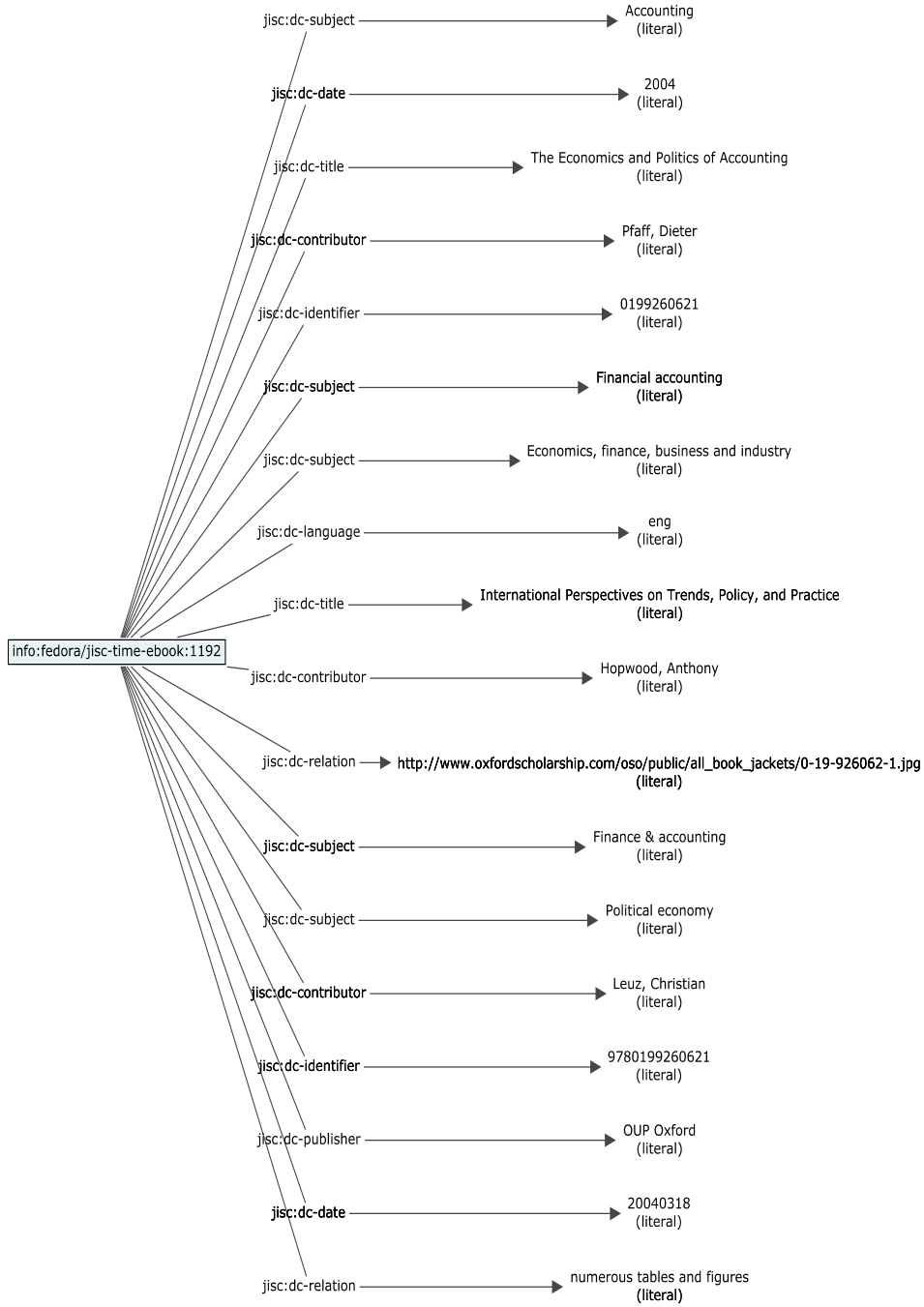


Figure 9 - RDF-EXT Datastream for sample ebook for resource discovery

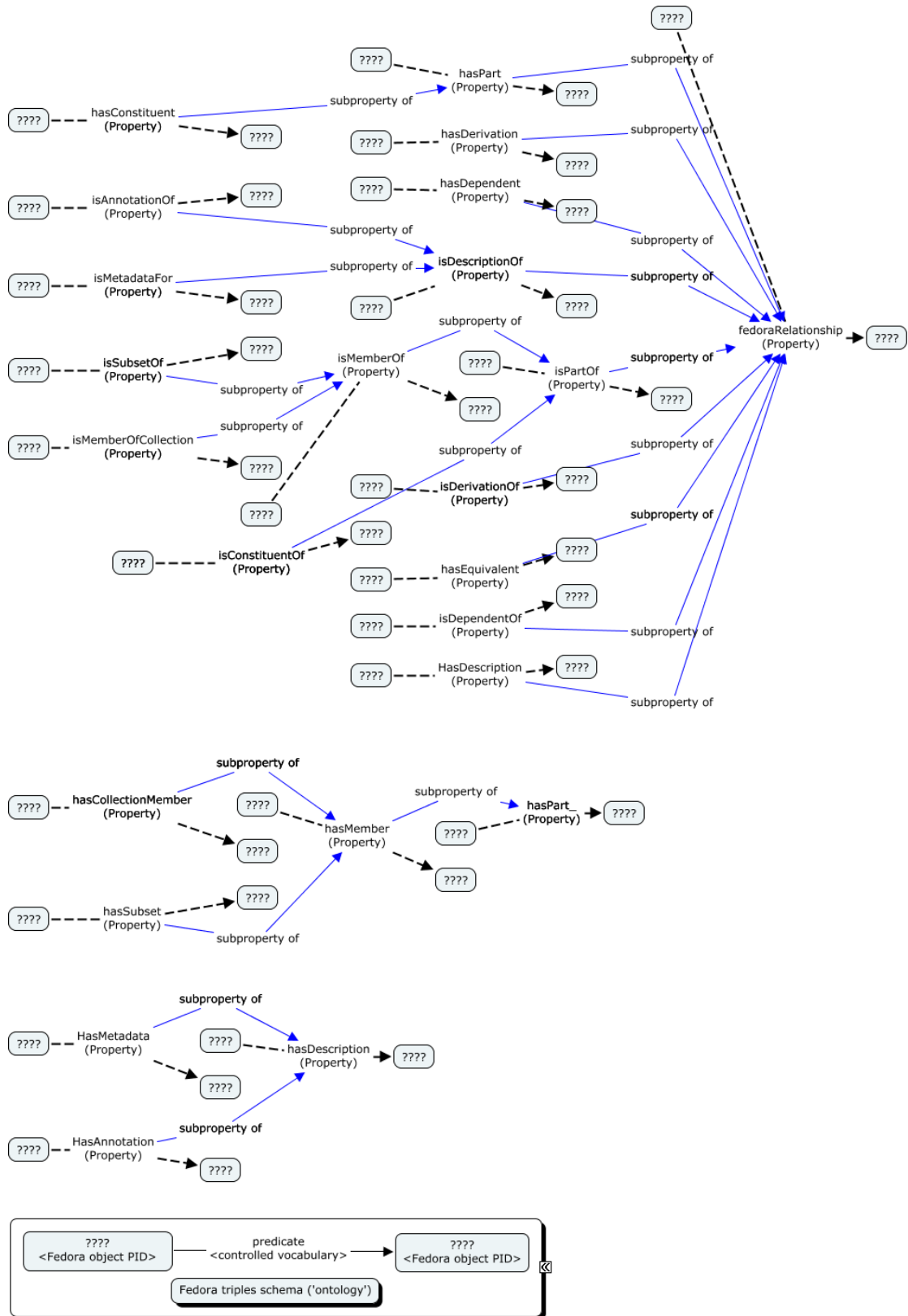


Figure 10 - Testbed ontology for RELS-EXT object-object property relationships

2.4 Custom Software Components

This section describes the software components which were required to deliver the Testbed system.

2.4.1 Presentation Layer (JISC-TIME Web Application)

The Orbeon Presentation Server (OPS) provides a toolkit for building the guided workflow and user-friendly browser-based interface to support all the user functions: record query and search, record viewing, and download.

The user interface was built as a 'wizard'. Each step of the wizard generates entries in a single XML document, thus maintaining state across all the pages viewed by a user. For instance, after querying for certain records, the user is displayed a table of results but may return to the same query to 'refine' these results. Similarly, a record may be selected for the metadata viewing page from the results page, but subsequently return to the results page without requiring a re-query.

The XML instance document is populated by the user in conjunction with other (web service) components built using OPS's XML Pipelining technology. These help configure the user display with various query parameters, and execute system functionality as the user progresses through the wizard pages.

A major advantage of using OPS to drive workflow is the high level of configurability and standards compliance: page-linking is defined in an XML file (Pageflow.xml), and the web pages themselves are defined using XHTML and W3C's emerging standard XForms. The XForms design decision allows the user application to be portable to different user client environments, whether 'thick-client' (outside a web browser), or even on a mobile device.

Furthermore, the XForms implementation of OPS uses a technique referred to as 'AJAX' (Asynchronous Javascript and XML), which means that Javascript is loaded into the web browser, and all communication to the server is via XML (XForms) in the background. This approach results in code which is both clear and portable, with a clear separation of concerns between client and server components of the user application. It also delivers a relatively responsive user experience when compared to other data-intensive web-based retrieval systems.

2.4.2 Metadata translation

2.4.2.1 FUNCTIONAL OVERVIEW

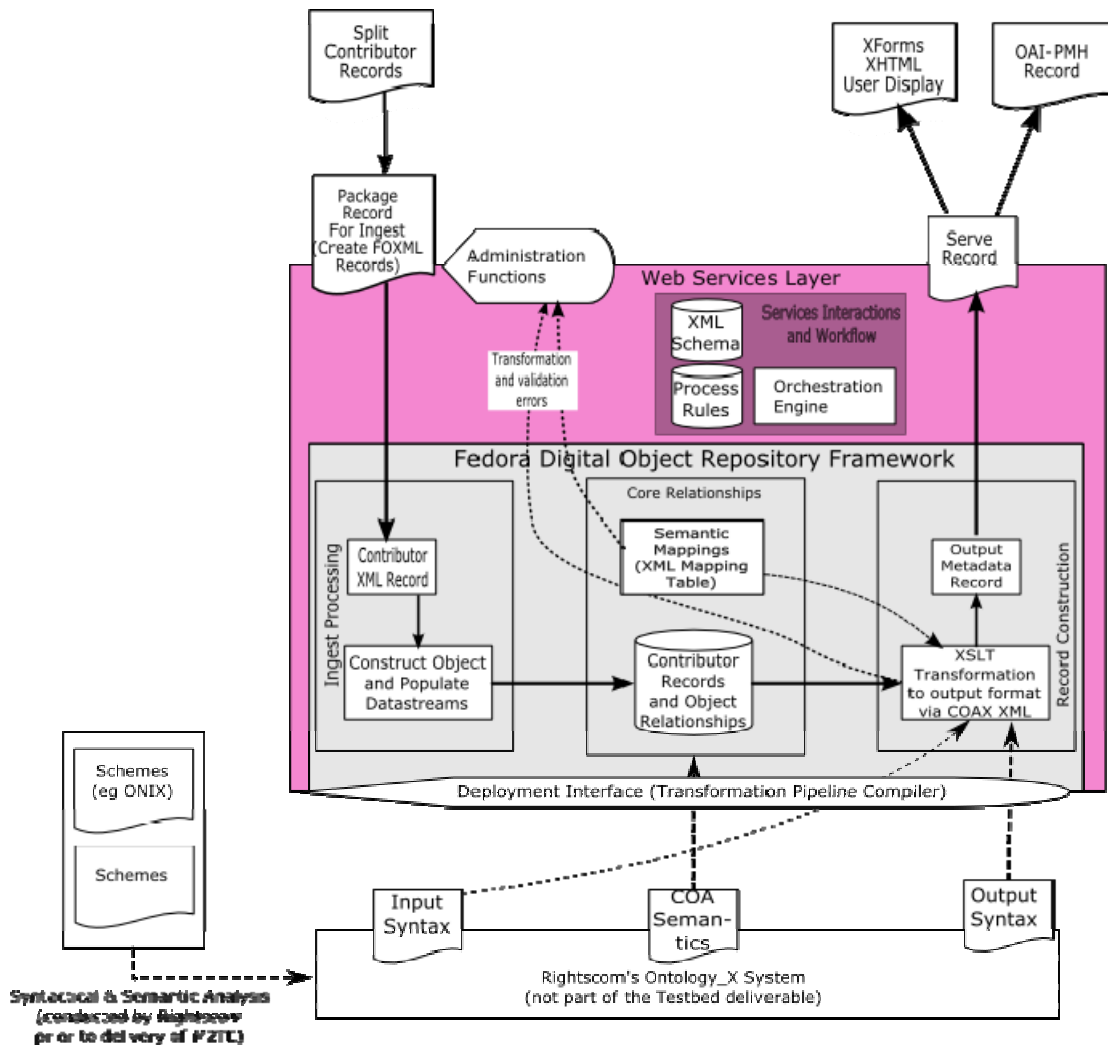


Figure 11 – Functional Overview of Metadata Transformation

2.4.2.2 THE COAX CENTRAL FORMAT

The metadata translation method was designed around a central format and its mappings to the different input/output formats. The central format is an implementation of the COA model using XML markup, so the underlying data model is the XML Infoset. The schema for the data model is called COAX (COA in XML). XML and XML processing languages were chosen for efficiency purposes: to allow a separate parallel development stream to be undertaken for the metadata translation service, completely independently of the remainder of the testbed system, whilst at the same time guaranteeing that the finished XSL translation components could be readily integrated into the heart of the Testbed.

To extract data from a COAX instance document, the XPath language was used to express access conditions over parts of the document, embedded within 'patterns' expressed as XSL stylesheets. These patterns are auto-generated by a component which parses Excel spreadsheets which served as the user input mechanism for defining the scheme mappings, and the XSL patterns serve to constrain further the various XPath constraints.

2.4.2.3 METADATA TRANSLATION SERVICE

A REST-style web service was developed which takes as input an identifier uniquely allocated to an instance of an e-book, an identifier for the target metadata format, and a URI representing the location of an object containing the contributor metadata to translate. The values of these inputs are generated using the wizard and attendant services in the Presentation Layer.

The service uses these values to generate a query to the underlying repository's RDF-based Resource Index to determine the contributor metadata format.

For both 'inbound' translations (translations from contributor to the COAX format) and 'outbound' translations (translations from COAX to target format), the service then looks up the specific sequence of XSL transformations generated by the scheme mapping component. These translations are then arranged into a stack of XSL XPL pipeline components, which are then called recursively using the OPS XPL language and executed in its runtime engine. The results are returned from the service to the calling code.

2.4.3 Repository Query Service

The Query service is a low-level service which serves to manage certain details of the Resource Index RDF database from the caller, such as repository query service location, web service character encoding and certain parameters such as the number of records to return. It accepts a query using the Resource Index engine's query language, iTQL, executes the query and returns the results to the caller in a defined XML format.

2.4.4 Ingest and Index Service

This service receives a request to ingest files located at a URL, using the 'file:' protocol.

It examines each file in turn to determine the contributor format, and uses the Repository Query service to determine whether the file has been loaded already, ie whether a 'create new record' or 'update existing record' is required.

For a 'create', it builds a new XML metadata submission wrapper using the repository's native FOXML vocabulary to prepare the record for ingest. The ingest is then performed by calling another utility service built using OPS, which creates a Fedora digital object for the e-book and submits the FOXML-wrapped object via the repository index. A new ebook record is created, with a unique identifier URI (Fedora natively uses the info: scheme), along with the Fedora Datastreams representing the different metadata views, and object behaviour definitions and concomitant executable web service bindings.

For an 'update', it uses a different utility service to locate the ebook within the repository, and update only the contributor metadata in the relevant Datastream for the ebook object.

In each case, the Ingest and Index service will add the following relationships to the Resource Index: an identifier for the OAI Provider service, the filename of the contributor record, and the contributor record format identifier. The Resource Index is also updated with the underlying relationships required for the correct functioning of the Repository.

The service then performs an additional indexing routine to support user queries via the web application, which allows the user to locate records based on free text field entries against Dublin Core terms. The Metadata Translation Service is invoked with the target metadata format defined to be Dublin Core. This produces a Dublin Core record in RDF for the ebook, which is added as its own new Datastream for the object

identifying the ebook, and the RDF within the Datastream added to the Resource Index to make the ebook record available for resource discovery.

To use the functionality of the Ingest and Index service, the physical files submitted by contributors were first separated out into a single file per ebook record. A utility XML Pipeline was created to automate this process, which were invoked using batch scripts created for Windows, or bash shell scripts for Linux, depending on the host operating system.

2.4.5 Record Validation Service

The Validation service was routinely run during development and testing to validate XML representations of ebook records against the schema for each. The service accepts a list of ebook object identifiers in an XML document, and will then run an appropriate schema check for each. The service is not accessible to users.

2.4.6 Repository Access Service

Receives a document defining the Repository's web service call required, along with the parameters and/or data required for that method, and routes the call to the generic web service implementation layer executed within the OPS engine. This service abstracts certain Repository API-specific details from calling components, such as whether the underlying web services interface call is based on REST or SOAP, and encoding specifics.

3 JISC Technical Architecture References

3.1 Software Stack

Windows Server 2003: <http://www.microsoft.com/windowsserver2003/default.msp>

CentOS Linux: <http://www.centos.org/>

Java (1.4.2_08): <http://java.sun.com/j2se/1.4.2/docs/index.html>

Fedora (2.1) The Fedora™ Project: An Open-Source Digital Repository Management System:

<http://www.fedora.info/>

Embedded Kowari (1.0.5): Tucana Technologies, Kowari metastore,

<http://www.kowari.org/>

Tucana Technologies, iTQL Commands, <http://kowari.org/271.htm>

Clark, K.G., SPARQL Protocol for RDF: <http://monkeyfist.com/kendall/sparqlprotocol/>

MySQL (4.1): <http://www.mysql.com/>

Orbeon Presentation Server: <http://www.orbeon.com/software>

<http://www.objectweb.org>

<http://forge.objectweb.org/projects/ops>

Trippi: <http://cvs.sourceforge.net/viewcvs.py/trippi/trippi/>

OAI-PMH Provider Proai: <http://cvs.sourceforge.net/viewcvs.py/proai/>

AJAX, Asynchronous JavaScript And XML:

<http://www.adaptivepath.com/publications/essays/archives/000385.php>

3.2 Standards

JISC Information Environment Architecture:

<http://www.ukoln.ac.uk/distributed-systems/jisc-ie/arch/>

<http://www.ukoln.ac.uk/distributed-systems/jisc-ie/arch/glossary/>

<http://www.jisc.ac.uk/ie/>

The JISC Information Environment and Web services:

<http://www.ariadne.ac.uk/issue31/information-environments/>

OAI-PMH <http://www.openarchives.org/>

Lagoze, C., Van de Sompel, H., Nelson, M. and Warner, S., The Open Archives

Initiative Protocol for Metadata Harvesting - Version 2.0,

http://www.openarchives.org/OAI_protocol/openarchivesprotocol.html

http://www.openarchives.org/OAI/2.0/oai_dc.xsd

Extensible Markup Language (XML), www.w3.org/XML/

XML Information Set (InfoSet), www.w3.org/TR/xml-infoset/

XML Path Language (XPath) 2.0, www.w3.org/TR/xpath20/

XSL Transformations (XSLT), www.w3.org/TR/xslt

XPL <http://www.w3.org/Submission/xpl/>

XFORMS <http://www.w3.org/Markup/Forms/>

HTTP1.1

<http://www.w3.org/Protocols/rfc2616/rfc2616.html>

(X)HTML <http://www.w3.org/Markup/>

"info" URI Scheme,

http://infouri.info/registry/OAIHandler?verb=ListRecords&metadataPrefix=oai_dc

URI Addressing: <http://www.w3.org/Addressing/>

SRW/U

<http://lcweb.loc.gov/z3950/agency/zing/srw/>

SOAP: <http://www.w3.org/TR/soap12-part1/>

RDF: Manola, F. and Miller, E. RDF Primer, W3C, 2003.

Beckett, D. and McBride, B., RDF/XML Syntax Specification (Revised),

<http://www.w3.org/TR/rdf-syntax-grammar/>

RDF(S): Brickley, D. and Guha, R.V. RDF Vocabulary Description Language 1.0: RDF

Schema. McBride, B. ed., W3C, 2004.

eXtensible Access Control Markup Language (XACML),

http://www.oasisopen.org/committees/tc_home.php?wg_abbrev=xacml

Web Services Description Language (WSDL) 1.1, <http://www.w3.org/TR/wsd/>

Dublin Core: <http://dublincore.org/documents/dces/>

3.3 Unimplemented but with some relevance

Metadata Encoding and Transmission Standard (METS),

<http://www.loc.gov/standards/mets/>

National Information Standards Organization (U.S.), The OpenURL Framework for Context-Sensitive Services, http://www.niso.org/committees/committee_ax.html

Functional Requirements for Bibliographic Records, International Federation of Library Associations and Institutions, 1998.

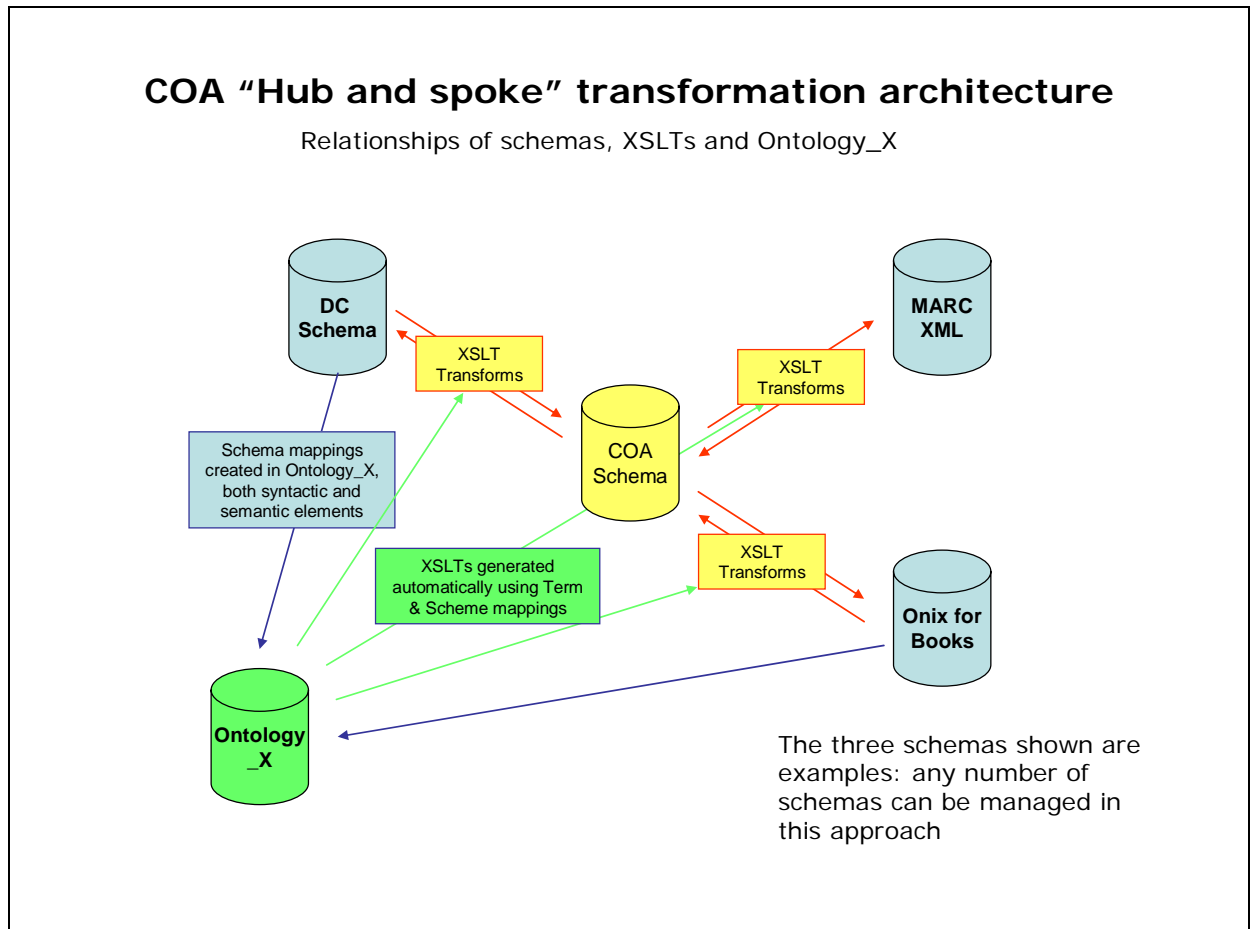
Athens access management system: <http://www.athens.ac.uk/>

Z39.50: <http://lcweb.loc.gov/z3950/agency/>

[Oracle Corporation](http://www.oracle.com/): www.oracle.com/

[OWL Web Ontology Language Overview](http://www.w3.org/TR/owl-features): www.w3.org/TR/owl-features

4 Transformation methodology



The TIME semantic transformation architecture is a “hub-and–spoke” approach based on the mapping of every required schema into a richer central schema, in this case the COA (Contextual Ontologyx Architecture) Xml Schema or “COAX”.

4.1 Summary of COAX capability

Such an approach will only work effectively if the central schema is rich enough to preserve all the semantics of all “spoke” schemas. The COA schema, developed from the <indexc> metadata framework, is a designed specifically for the purpose of interoperability between other schemas. The underlying COA model is based on five basic Entity types:

Resource, Agent, Time, Place, Context

one linking element:

Relator

and nine Attribute types (which are subtypes of the Resource Entity):

(“free text” values)

Descriptor, Name, Identifier, Annotation

(“controlled” values)

Category, Class, Flag, Quality, Quantity

The Entities each relate to one another using types of Relator, and any Entity may have any type of Attribute (Attributes may also have any type of Attribute). The “specialization” of types and Relators is managed in *Ontology_X*, the ontology built by Rightscom on the COA model.

The COA model is “triple” based, representing metadata as sets of Domain-Relator-Range statements such as “A HasAuthor B” or “A IsA EBook”.

The COAX XML schema is built on this model, which allows for any semantic relationship at any level of granularity to be represented. The result is much more verbose than a typical metadata record, but as the purpose is for storing data for interoperability, and not human readability, this is not a problem.

The example below illustrates how information about an author may be represented in three different schemas:

Dublin Core

creator=Kriegel, Mark

MARC

tag=100

subfield a=Kriegel, Mark

subfield e=author

ONIX for Books

Contributor

ContributorRole=B01

PersonNameInverted=Kriegel, Mark

NamesBeforeKey=Mark

KeyName=Kriegel

In **COAX**, the various Entities are assigned arbitrary identities (shown here with letters A and B in this example), and then linked or given attributes in a nested structure (the example shows a simplified form of the coax xml nesting):

COAX

A Class=Resource

 Class=EntityType

A HasPrincipalAuthor B

B Class=Party

 Class=EntityType

B Name=“Kriegel, Mark”,

 Class=NameInverted

 Class=NameType

 NamePart=Kriegel

 Class=KeyName

 Class=NamePartType

 Identifier=1

```
Class=SequenceNumber
NamePart=Mark
Class=NamesBeforeKeyName
Class=NamePartType
Identifier=2
Class=SequenceNumber
```

The more verbose COAX structure contains all that is necessary for mapping and out of the other schemes, provided it is combined with `Ontology_X`. The `Ontology` is essential for the translation of specific terms. For example, as a result of mappings it contains the information that:

- ONIX code “B01” represents “author” and is equivalent to `coa:Author`
- MARC “author” is equivalent to `coa:Author`
- Dublin Core “creator” is equivalent to `coa:PrincipalContributor`

The COAX Relator `coa:HasPrincipalAuthor` shows that B’s role is that of `coa:PrincipalAuthor`, which ontologically is a subclass of both `coa:Author` and `coa:PrincipalContributor`. A COA transformation can therefore preserve the appropriate semantics in each scheme.

4.2 Mapping schemas

For each schema a once-off specific “COA mapping” is made, using `Ontology_X`. This mapping is made in “triples”, and it represents both the syntax and semantics of the schema. For example, it not only contains the syntactic information that element X is called “Author” and has a Datatype of “String” and Cardinality of “1-n”, but it contains the semantic information that “X IsNameOf Y” and that “Y IsAuthorOf Z”. It is this latter dimension which is unusual and distinguishes the COAX approach from more simple syntactic mappings which do not make the semantics explicit.

4.3 Generating XSLTs

From the schema mapping an XSLT is automatically generated, using a Rule generator developed by Rightscom, which then enables the transformation of documents conforming with the schema in and out of COAX. This automated code generation is a significant benefit in time-saving and accuracy. Any change to a schema can be reflected just in making a change to its COA mapping, and then the XSLT is automatically re-generated to support the updated transformation.

4.4 Pre- and post-processing

The Service Oriented Architecture “XML Pipeline” approach which was adopted lends itself to a “staged” approach to transformation where it is useful, with successive transforms being used to achieve the end result. We used this in particular in the management of MARC data. The source MARC schema (MARC-XML) is very basic, so before applying the COAX mapping, we used an additional XSLT to produce a more detailed XML schema for MARC, which enabled us to break out the different semantic elements of MARC control fields into separate XML tags so that the semantics of each could then be mapped separately to COAX. Therefore each transform from MARC to another schema goes through three stages: MARC XML to MARC expanded, MARC expanded to COAX, and COAX to the required schema.

Adding, changing or removing such steps from the pipeline is a simple task, and provides more flexibility.

4.5 Transformation scope

We received about 1886 ebook metadata records from four different publisher or intermediary sources, in ONIX, MARC21 and Dublin Core formats. We were unable to source any LOM input metadata.

We created mappings for MARC21, ONIX, Dublin Core (in three variants: simple, qualified and OAI-compliant) and for LOM.

We only mapped those fields/tags in MARC, ONIX and LOM which contained data in the records supplied, or for which data was available in source records from other formats. There were a considerable number of

Transformations were simply from one document to another, via COAX. No capability for merging or comparing the contents of data was in the scope of the project. Cleaning of input data was also out of scope.

4.6 Transformation limitations

Transformation can only preserve semantics: it cannot add semantic content which is not there to begin with. Where the semantics of two schemas are incompatible with, or orthogonal to, one another, no transformation methodology can bridge the gap, and meaning will be “lost” in transformation because it has nowhere to go.

5 Technical conclusions

The transformations for all formats were successful and achieved reasonably good results: normally very good for ONIX and MARC and less so for Dublin Core and LOM. Transformations happen in an acceptable time, in real time.

There are a number of outstanding detailed technical issues (shown in the Appendix) which we were unable to resolve within the time and resource available to the testbed project. All of the cases of data wrongly “lost” or misplaced in transforms which were identified by testers or by our own team were either resolved or are covered in this list.

A few of these issues (A2, A3, A7, A11, A12, C1, C2) are resolvable just by some further work on the COA Rules engine (though see comments below on the alternative RDF-based approach). However, the majority of issues arise for other reasons, which may be identified under these headings:

- Semantic loss: relative strengths of metadata schemes
- Data quality issues: errors
- Data quality: “variant schemes”
- Data manipulation: concatenation and string analysis
- Use of conditional rules
- Use of default rules

5.1 Semantic loss: relative strength of metadata schemes

Transformations both in and out of Dublin Core were generally poor, because of its relative semantic poverty and ambiguity.

As a source schema, unqualified or lightly-qualified DublinCore has huge limitations. For example, dc:date may be the date of creation, date of publication (and if so, where?) or of anything else. Unless a default assumption is made (see below), such data cannot be transformed and is “lost”. dc:identifier often does not provide the IdentifierType, which renders it meaningless. Text in dc:coverage text may mean more or less anything. The absence of controlled values in basic DC means that code lists such as those supported by Onix and MARC cannot be mapped into. As a human readable record, Dublin Core has its uses, but as a basis for automated transformation it is effectively a non-starter.

As an output schema, DC does much better, and good DC records can be produced from ONIX or MARC input. It would benefit considerably though from data concatenation (See below).

Both ONIX and MARC are good as source schemas for descriptive eBook metadata. They each have some inherent limitations, but most of these can be overcome with the measures described below.

5.2 Data quality issues: errors

Input data inevitably contains errors, both random and systematic. The most frequent systematic error is the misinterpretation of some fields and their use for data that actually belongs elsewhere. This happens mainly in loosely defined schemas such as DC and MARC, but we have also found misplaced data in ONIX files. For example, one set of ONIX input data included the email address of the sender in the

FromPerson element, although there is a specific FromEmail element available for this.

Another type of systematic error is the use of a wrong ONIX code. For example, one set of input data, which had been derived from print book data, contained ONIX format codes showing each eBook incorrectly to be either a Hardback or a Paperback book.

Random errors were discovered by chance when evaluation the quality of the transformations by analysing a few test files. For example, in some data sets, the affiliated role of an author (eg "Professor Of Physics") was sometimes included with his affiliated institute ("University Of Somewhere") and sometimes included in a separate field.

For random data errors there is nothing that can be done apart from manual correction when an error is detected. For systematic or habitual errors, there is considerable scope for management (see next two points).

Generally the quality of data supplied was good, also in terms of the amount of data contained in each record and in terms of the homogeneity of metadata from record to record, but from such a small participating sample this says nothing about the general standard.

5.3 Data quality issues: "variant schemes"

An issue which emerged in some sample data, and which we have found is commonplace from other projects, is that of variations in the way in which a particular schema is implemented. For example, as quoted above, one publisher used an ONIX format code to indicate the code of the source printed book, instead of the eBook. The same was done with ISBNs. Another supplier had a set of controlled values which they used for a particular MARC tag which were not standard, but were internally consistent. In another case (issue B2) MARC tag 043 (Geographic Area Code) was apparently used in a particular and consistent way by the supplier, but as MARC itself is non-specific, nothing could be done with the data in the general scheme.

All of these cases amount to there being "variants" of existing schemes (it has been remarked that as there are over 50 UK publishers now providing data in ONIX, there are over 50 variant ONIX schemas). MARC users also have their own variant practise, especially in the use of "internal" 900 tags. However, for the COAX one-to-many approach this is not in principle a problem. Variant mappings (which typically only require a minor adaptation to the basic schema mapping, from which a new "variant" XSLT can then be generated) can be made for different sources where consistent behaviour is identified, whereas to maintain such variations in conventional pairwise mapping is clearly problematic in terms of volume. This suggests that supporting variations may be a major benefit of the one-to-many approach.

5.4 Data manipulation: concatenation and string analysis

In the testbed we did not do anything to combine or analyse the contents of individual fields (except for MARC control fields as described in Pre- and post-processing above). However there is considerable scope for applying techniques of concatenation (combining two or more fields) or analysis (analysing components of string fields).

For example, MARC and ONIX handle measurements differently. ONIX (like COA) separates out the quantity, the unit of measure and the type, so (for example) "Height

25 cms” goes in three fields in ONIX (and COA) but only one (or sometimes two) in MARC and DC.

Clearly, in one direction this is fine: accurate outputs for MARC and DC can be generated by producing a concatenated field from ONIX data within COAX. From the other end it is less straightforward, but where consistent practice is employed in MARC, it is possible to analyse “Height 25 cms” into controlled value COAX fields. Comparable approaches can be taken for manipulating elements such as date/time variations, personal names and certain identifiers, with the likelihood of a high level of accuracy in results. A number of other examples are given in section B in the Issues List.

An allowed value can always be presented as a text string (so, for example, reporting ONIX or MARC controlled values as text strings in DC is simple). Text strings cannot so easily be mapped to controlled value sets, but where a margin of error is acceptable there are options to do this against `Ontology_X`, with or without support of thesauri.

5.5 Use of conditional rules

There are occasions where application of conditional rules are needed to enable a transformation. For example, to know whether an ONIX contributor is a Person or an Organization, it is necessary to look first at the `PersonalName` and `OrganizationName` fields to see which one is populated. Issue B13 (which results in the duplication of date elements in DC outputs) as an example of where more conditional logic is required.

The application of conditional rules, where unrelated elements in a message need to be queried, is an area where the conventional “tree walking” approach taken in XML can impose limitations.

5.6 Use of default rules

Default rules have two main functions. The first is to meet mandatory schema requirements where insufficient real data is available (if MARC requires a single “main” author, and ONIX does not differentiate between several, then some default convention is needed (such as the assumption that the first reported is the “main” one). This default assumption does not result in changes to the COAX data: it is only applied on output (so, for example, DC which does not differentiate would never know that the default assumption had been applied).

The second function of default rules is to enable “best guess” transformation where there are no definitive semantics. This occurs for a variety of reasons, most typically where sets of values in one schema are orthogonal to another. For example, DC differentiates creator and contributor according to the *importance* of the role, where ONIX differentiates by the *type* of the creative role (for example, `Author` and `EditedBy`), so there is a semantic mismatch in mapping from ONIX to DC (though not vice versa). A default rule here may be based on the role itself, or on the combination of (say) role and resource type (eg an editor may be a secondary role if the resource is a novel, but a principal role if it is a compilation or a newspaper).

In some cases there is no obvious default, or several options. In the latter case, there may be a balance of probability (eg “date” is probably data of publication, but might be date of authorship or date of original publication), and that probability may even be expressed numerically (80%/10%/10%).

The Issues list contains three cases (B3, B5, B6) where default assumptions have not been made in the absence of sufficient advice.

For a production version of a TIME service there is a strong argument for establishing an (optional) human interface for the management of default

assumptions, or the making of ad hoc choices in specific cases. In principle, it is bad practice for defaults to be applied “blindly”: at the very least the user should be aware of the assumptions that have been applied. Each user may have different preferences, in accordance with their own practices, and so the provision of “configurable” defaults and choices would support transparency and respects genuine variation in practice.

5.7 Limitations of XML/XSLT technology

XSLT is, in principle, sufficiently expressive to support COA transformation. However, as remarked above the conventional XML “tree walking” does not always make conditional rule writing as straightforward as it might be.

The XSLT/XML schema approach also does not offer a way forward to deal with two clear possible requirements of a TIME production service: (a) combining data from multiple messages and (b) enabling data integration or cleanup across multiple messages.

In the course of the TIME project, other open-source tool options have become available, making use of an RDF (Resource Description Framework) implementation of the COA model to create an RDF “triple-store” instead of a COAX XML document at the heart of the transformation process. The same COA mappings and rules apply, but the triple-based approach appears to offer a better development environment than XSLT for applying conditional rules, and would also support (a) and (b) above. Rightscom is currently testing this approach, which makes use of the same repository and pipeline architecture.

5.8 Alternative approaches

There are three possible approaches to message transformation:

- “pairwise”, ad hoc one-to-one mapping
- “hub-and –spoke” mapping through COA
- “hub-and –spoke” mapping through another central model/schema

There is no doubt that simple pair-wise mapping, say, from ONIX to MARC, would be a cheaper and equally effective solution. Some of the issues which arise in developing the COA rules engine can be ignored, and ad hoc solutions for specific elements could be implemented. Person or persons sufficiently knowledgeable in both schemas could reconcile the semantics.

However, a point of critical mass is reached with a certain number of schemas, and a certain number of variations, where the pair-wise mappings begin to become increasingly untenable because of (a) their quantity and (b) the lack of sufficient expertise to understand the detailed semantics and practices of both schemas at the same time. It is not clear where that critical mass is reached, but we believe it is already well passed with the scope of the TIME testbed (four formats and two variations), which require 6 mappings in COAX and 15 for pairwise mapping.

The main new thing we have learned in the course of this project is that it is essential to be able to manage *variations* of schemas, with flexibility for local rules and practices and not imposed “central” solutions. The COA approach works well for that.

5.8.1 Other metadata types and schemes

The schemas are fairly homogenous in purpose (Resource Description) if widely diverse in structure and scope. ONIX, MARC and DC represent three extremely different approaches to structure: DC flat and simplistic; ONIX a typical

contemporary XML-based, drop-down-list-driven nested message; and MARC a pre-computing cataloguing tagged format relying heavily on text presentation rules yet more heavily “marked up” with tags than XML. There has been no particular difficulty in mapping all three approaches to COA: that is to say, COA itself has not itself placed any significant barriers in the way of interoperability between them: there are limitations, but they are inherent in the schemas themselves. We believe this confirms (as have our mappings of schemas outside the TIME project) that COA is suitable for mapping metadata from any schema structure.

Within reason, we believe any type of metadata content that is of interest to the JISC community can be accommodated in this methodology. COA is being used as the underlying structure for the development of ONIX for Licensing Terms, and is expected to be used for an ONIX for Rights message, which suggests that an extension of the scope of a TIME service to rights-based metadata would not be a problem. The LOM schema mapping shows that, in principle, specialized pedagogic data can be supported (although real sample data is needed to demonstrate this), and there is no reason to suggest that technical or preservation metadata could not be handled equally well.

Because COA is not eBook or text-specific, Time can broaden its scope, say, to other electronic content types (COA already underpins the new music industry MI3P message standards for digital content)

5.9 EPICentre observations

5.9.1 Transformations

It was noted that, after conversion, most data elements appeared with codes that were reasonable, but were not necessarily the same codes for the same data element for all conversions – for example, conversions into DC placed authors in ‘contributor’ not ‘creator’, even in single-author works, and publishers often appeared as ‘contributor’ as well. Again, in conversions into DC, various types of information appeared as ‘relation’ as a catch-all category, rather than in a more specific and more appropriate category. ONIX → DC conversions put URLs into ‘relation’ while MARC → DC conversion placed URLs into the more appropriate ‘identifier’ category.

As noted elsewhere, ONIX metadata, being designed for publishers’ use, includes some items not easily transformed into the other systems – for example, authors’ biographical/employment information is lost on conversion from ONIX into either DC or MARC (“James Buggins is Professor of Widgetology at Mudville University and was formerly C.E.O. of Acme Widgets Inc.”). In a transformation into DC it would need to be concatenated with other information as DC does not provide any specific fields for this metadata. This is perhaps not too serious, but does represent loss of data. More seriously, in some conversions out of ONIX (where the conversion involves target schemes that do not provide a separate field for it), the city of publication is lost; this is unfortunate, as it is an element of the standard bibliographic citation of a book.

MARC being, on the other hand, a librarian’s system, it has different features that are not found in the others. In MARC → DC conversions, many of these descriptive features are placed in ‘coverage’, which is not really accurate but at least means that the data are there. In MARC → ONIX conversions, however, some MARC information in 5xx MARC codes (‘Notes’) is lost in conversion and, seriously, this includes the URLs.

Conversions from other systems into MARC, and even MARC → MARC conversions via COAX, sometimes place data into a different code from that expected. For example, URLs should be in the 856 field but are often placed in the 024 field upon

conversion. Some conversions into MARC also lost the date of publication, a serious flaw. Some items of data are present more than once in different fields in the MARC input record, and the conversion handles this duplication in unpredictable ways. In particular, authors variously appear in the converted output in the 100, 700, 710 and 720 fields. The 520 field is used for all 'notes' in conversions into MARC; this field should be for summary notes, with 500 for general notes. For the lay user, some of the niceties of MARC may be unimportant so long as the data are there somewhere, but for the skilled information professional searcher incorrect assignment to MARC fields may be a problem.

The most serious flaw in the conversions was found in the DC ONIX conversions, where URL, language and date were all lost, as were 'description' and 'type' fields. Publication date is a key piece of metadata, and the URL is essential for electronic publications to be found. The problem of the missing metadata arises because DC is the simplest of the formats. The meaning of the DC fields is only coarsely defined and the mapping to precisely defined fields in the target scheme requires a refinement of the meaning of the original metadata that can often only be achieved by inspection of the metadata itself. A typical example is the 'date' field which can hold any date, not necessarily the publication date, and here even the inspection of the date does not indicate its precise type which is required for mapping to a schema which distinguishes between different date types.

5.9.2 Publishers' original data

The checking of the conversions also revealed errors in the input data supplied by publishers in some cases. As only a small number of documents were examined, it is not possible to say how widespread the problem is, but as in all computer systems the 'garbage in, garbage out' rule applies, and flawed input data leads to flawed output data. One MARC-input document, for example, had allocated the first author's surname to both authors in the 700 field, but had given the second author their correct surname in the 245 field. This made it possible to identify where the conversions had used code 700 input and where they had used code 245 input – quite useful – and unfortunately in most cases the erroneous data had ended up in the principal location and the correct data in a secondary location.

In two cases, OUP had given two dates of publication to a document, one being the correct one and the other being 1600 (which, it is surmised, was the date of foundation of the Clarendon Press). The conversions had unfortunately set 1600 as the date of publication. In another case, a MARC input document, the input date was given as 200?, and the system converted it to 2000; perhaps the system had not allowed for non-numeric characters in dates.

The lesson here is that book publishers need to take great care to ensure that their metadata is correct, whatever metadata system they have chosen to use, since in future algorithmic conversion between metadata systems are likely to become more common. Journal publishers already face this issue owing to the existence of systems like CrossRef.

6 Table of data issues

This list identifies issues that might be found by users of the testbed.

Known issues				
Schema	ID	Known issues		Issue type
MARC output	A1	Contributors (Marc tags 1xx/7xx)	At present, contributors from ONIX, DC or other schemas are shown in Marc tags 100 and 720. Organizations are mapped to tag 100, as it is often not possible to tell the type of the contributor. The main contributors of tag 100 are also shown in tag 720.	Processing issue
	A2	Publisher (Name) (Marc tag 260b)	The Imprint Name from ONIX is currently not output.	Processing bug / Status unknown
ONIX output	A3	RelationCode	The ONIX output is missing a mandatory element RelationCode in the RelatedProduct element.	Validation issue
	A4	Missing default values	ONIX code lists don't allow for default values for some fields, which is fair for cataloguing items directly, but means that an inappropriate code has to be chosen for mandatory fields (e.g. for ProductIDType).	Schema issue
	A5	SubjectSchemeIdentifier	SubjectSchemeIdentifiers in the ONIX output are set to their default value even if information is available (e.g. when mapping from MARC tags 050 and 082).	Semantic issue
	A6	PersonName	When mapping from DC, ONIX PersonName is also given for Organizations as DC doesn't allow to distinguish between Persons and Organizations.	Semantic issue
DC output	A7	Audience Code	The AudienceType is currently not mapped out to dc:type.	Processing bug / Status unknown
	A8	Publisher	The Imprint is currently not mapped out to dc:publisher.	Semantic issue / Status unknown
	A9	Creator/Contributor	Data from MARC tag 100 is currently mapped to Contributor rather than to Creator.	Semantic issue
	A10	Subject/Format	Data from MARC tag 655 Index Term (Genre/Form) is currently mapped to Subject rather	Semantic issue

LOM output	A11	Namespace	than to Format. All LOM element names should be in the namespace specified in the schema.	Validation issue
Any	A12	Invalid OAI-PMH output	Validation errors with metadata formats means that valid OAI-PMH output cannot be produced	Validation issue

		Future requirements	These have not been resolved in testbed because either (a) they are out of scope or (b) they require more development time than is available under the current project.	
Schema	ID	Semantic issues	Recommended future enhancements	Issue type
MARC input	B1	Nonspecific Relationship (Marc tag 787)	When an Identifier is shown in subfield "c" its type is not indicated (only its value is).	Concatenation required (beyond scope)
	B2	Geographic Area Code (Marc tag 043)	Marc tag 043 is non-specific as to its function and so cannot be mapped to COA with any certainty.	Supplier specific rules (beyond scope)
MARC output	B3	Nonspecific Relationship (Marc tag 787)	At present, Annotations which describe parts of a Resource (for example, "Numerous figures and tables") are put into tag 787.	Clarification needed
MARC input & output	B4	Extents (Marc tag 300)	The 300 subfield does not necessarily indicate the type of extent which is described (eg "22 cm") so Marc data cannot unambiguously be mapped to COA. Data from other schemas such as ONIX also needs to be shown in Marc as a concatenated field [eg 22 cm (Width)].	Concatenation required (beyond scope)
ONIX output	B5	Electronic location and access	Elements from MARC tag 856 (Electronic Location and Access) have no clear counterparts in ONIX for Books.	Clarification needed
	B6	Subject	Library of Congress Call Number (MARC tag 050) has no clear counterpart in ONIX for Books.	Clarification needed

DC output	B7	Semantic limitations	Elements from ONIX and MARC are output appropriately to Dublin Core, but many DC fields suffer from inadequate definition if data is stored simply. Concatenation can be done in many cases if it is felt to be beneficial and not merely confusing (see specific issues below).	Semantic inadequacy
	B8	Identifier	dc:identifier would benefit from the concatenation of IdentifierType (eg "ISBN: 0201615924")	Concatenation required (beyond scope)
	B9	Relation	dc:relation would benefit from the concatenation of relationship type (eg "Hardback: ISBN: 0201615924 ")	Concatenation required (beyond scope)
	B10	Creator, Contributor	dc:creator and dc:contributor would benefit from the concatenation of contributor type where one exists (eg "Author: Smith, John"). Also affiliation details could be added when mapping from ONIX.	Concatenation required (beyond scope)
	B11	Title	dc>Title would benefit from the concatenation of a TitleType where one exists (eg "Subtitle: Advanced Use Case Modelling")	Concatenation required (beyond scope)
	B12	Publisher	This field currently only holds the publisher name when mapping from MARC or ONIX, not the place of publication. It seems that the latter is attached as a qualification (something like "Springer, New York and Heidelberg").	Concatenation required (beyond scope)
	B13	DC extended	How to suppress dc:date when a Date already goes into dcterms:available?	Rule-based processing required (beyond scope)

Schema	Processing issues	Recommended future enhancements	Issue type
MARC output	C1 Varying Form Of Title (Marc tag 246)	Currently three tags are merged into one. Since they have different ind1, there is a conflict and the first indicator of tag 246 is not currently mapped.	Processing issue
ONIX input	C2 Subtitle	At present, a Subtitle in ONIX is shown only as a separate Title and not as a Subtitle of the Title.	Processing issue

Any	C3	PlaceName as Subject	We have to review how we map Places that are Subjects (or Subjects that are Places). The current syntax works fine for identity mappings, but it's not possible to map this to dc:coverage (or dc:subject).	COAX enhancement required (no impact currently)
	C4	Attributes of Links	onix:NumberOfPages and onix:IllustrationsNote are currently shown as attributes of a Resource, rather than as attributes of a link between Resources.	COAX enhancement required (no impact currently)
	C5	Displaced attribute naming	The specialized naming we introduced for displaced attributes is fine for the present, but it does have the unhelpful side effect that it means the name is specialized both in the Displaced Attribute and in the fully contextualized one, so the latter is an over-specialization.	COAX enhancement required (no impact currently)
	C6	Transforming classes into Annotations	Produce an Annotation from a Name of a Class (i.e. a text-equivalent of a controlled field).	Processing issue
	C7	Probabilities in case of semantic ambiguity	Add probabilities for Types, Classes and Categories (and Qualities/Quantities) where there is no 1-to-1 map. At present the few examples we have (eg Place) operate on 100% default.	Processing issue

Schema	Choreography issues	Issues that will require solutions when the platform is put into production	Issue type
--------	---------------------	---	------------

Any	D1	Message Audit Data	Message Audit Data is not input or output in the testbed.	Depends on choreography
-----	----	--------------------	---	-------------------------

Schema	Publisher/format specific issues	Issues that will require solutions when the platform is put into production	Issue type
--------	----------------------------------	---	------------

MARC input	E1	MARC punctuation	The NetLibrary records include MARC punctuation at the end of fields (eg Dalai Lama, 216 pp;), which is not required in other formats.	Supplier/format specific data cleansing
------------	----	------------------	--	---

(beyond scope)

	E2	Missing data	Some data records are missing important metadata such as publication dates.	Supplier specific data cleansing (beyond scope)
ONIX input	E3	Incorrect ONIX Data	OUP input data incorrectly sometimes puts ProfessionalPosition data in Affiliate field. OUP codes the ProductForm as "Paperback" (presumably the format it is based on), although it is an eBook. OUP input records contain many <YearFirstPublished>1600</YearFirstPublished> entries.	Supplier specific data cleansing (beyond scope)

Schema		Systems issues	Known issues with system functionality	Issue type
Any	F1	Web application	Slightly odd query results colour highlighting	Functionality
	F2	Transformation framework	Redundant XML namespaces are sometimes inserted in XML record output; this does not affect a record's validity or subsequent usage	Processing bug

LOM transformation issues

No LOM input data has been provided.

7 Suitability of various metadata schemes for ebooks

Of the four metadata schemes under discussion, ONIX is widely used by publishers, provides the richness of information required for discovery, acquisition and retrieval but is weak, compared with LOM on specifically educational metadata designed to be used for management purposes. ONIX is highly structured with many code lists, such as the UK book trade standard BIC categories, basic audience level codes and a comprehensive codelist of e-book formats. ONIX records can also include descriptions, reviews, tables of contents and price and availability information

Dublin Core provides a limited set of 15 data elements, many of which map to a group of structured fields or subfields rather than to a single element in richer schemes such as MARC or ONIX. Although qualifiers such as sub-elements may be used in Dublin Core, the lack of a standard scheme would create problems of interoperability if they were used. Despite Dublin Core metadata could certainly provide a level of assistance in discovery but the lack of information about source of supply, price etc is a serious limitation for acquisition purposes.

LOM was designed by IEEE to describe learning resources at a highly granular level. It is very little used by publishers but is regarded by the learning community as an extremely valuable tool in the selection and management of resources. It's main benefit over the other metadata schemes here mentioned is its ability to assign learning-related

metadata elements such as *Typical AgeRange, Difficulty, Typical Learning Time, Interactivity Level etc.*

MARC is almost universally used by librarians in one national "flavour" or another. Designed to automate the creation and communication of library catalogue information, it is hardly ever used by publishers. It provides a rich record although, once again, availability and source of supply are not included. The great benefit of MARC, especially for cataloguing purposes, is that there are well defined and well observed rules (AACR2) concerning the representation of data within a record.

In summary, the only schemes likely to be implemented by e-book publishers at the moment are Dublin Core and ONIX. Dublin Core provides simplicity, but at the cost of losing the richness of information that ONIX delivers. Both formats will facilitate discovery but ONIX will better allow a user to make a decision about the suitability of the resource and will also facilitate acquisitions since it can include detailed table of contents information, reviews, sample text etc., as well as price and availability.

However, to adequately fulfill the purpose of discovery, acquisition and retrieval, a prime requirement of e-book metadata is reasonably comprehensive coverage of the considerable mass of e-books available. Without this, any comparison of the suitability of different metadata schemes is somewhat academic. Furthermore, although an option to restrict a search to digital resources may be useful, in many cases users will want to search on both physical and electronic books. We therefore recommend that the publishers of the existing books-in-print databases be approached and encouraged to carry e-book product and availability information including those titles only available from intermediaries.

Theoretically, it may be possible to search across publishers' databases and harvest metadata using Z39.50 or, more appropriately, OAI-PMH that could deal with ONIX data since it is expressed in XML. It is, however, unlikely that sufficient numbers of publishers would have the technical competence or commercial incentive to expose their metadata in a sufficiently standard format to make this feasible unless this were

a condition of purchase by JISC. Even so, it would be preferable to have an easily accessible database, preferably integrated with a database of physical books.

7.1 Standards for the expression of usage rights in e-books

In many cases, e-books are currently hosted and made available through intermediaries that apply and technically enforce their own conditions of use, depending upon their particular business model. In this closed situation, standards for the expression of usage rights have not been an issue. However, libraries have not been universally content with this approach and, as more and more publishers develop and market their own repositories of e-books, models may change to resemble the more open approach taken in the case of e-journals, where conditions of use are not technically enforced but rely on libraries to oversee compliance with licences and ensure that access is only granted to authorized users.

Even where intermediaries continue to apply technical protection, the increase and diversity in digital resources will result in greater variation in usage terms and libraries will need to be able to store these terms in their electronic resource management systems form and communicate them to users.

A report by Intrallect for JISC included the following requirements gathered from libraries:

- Rights should be expressed in machine readable form
- Whenever a resource is described its rights should also be described
- Users should be able to see the rights information associated with a resource

As libraries have attempted to incorporate digital resources into their collections, services and operations, they have found that existing integrated library systems lack the required functionality to support the management of these resources.

In 2002, the Digital Library Federation (DLF), a grouping of the major US academic research libraries, set up their Electronic Resource Management Initiative (ERMI) to aid the rapid development of library systems by providing a series of papers to help both to define requirements and to propose data standards for the management of electronic resources.

ERMI's deliverables included a study of the functionality required from electronic resource management systems, a data element dictionary that included licensing terms, and an investigation of the suitability of existing rights expression languages for the communication of licensing terms such as the Open Digital Rights Language (ODRL) and XrML, the basis of the MPEG21 Rights Expression Language (ISO//IEC 21000-5).

The ERMI report noted that both these initiatives were intended not only to enable rights holders to express what a user may do with a particular resource but also to drive technical means of enforcing those terms and conditions. This reflects their origins in the music and video industries whose prime concern was to prevent illegal copying and distribution of their intellectual property. Libraries and academic publishers prefer to rely on ERMI led them to the view that these RELs were unsuitable for the purpose of expressing publisher library licensing terms. Instead, they proposed development of a purpose-built XML format for expressing rights and drafted a model "ERMI native format".

Neither ODRL nor the MPEG21 REL can adequately deal with exceptions. Another major issue is that they are designed to have a one-to-one relationship to a resource whereas, typically, a publisher/library or aggregator/library licence will cover a whole manifest of resources.

EDItEUR, the international e-commerce standards body for the book and serials sectors, had already identified the requirement to express licensing terms in their ONIX for Books and ONIX for Serials standards and commissioned a review of the DLF ERMI work to assess its suitability. The review found that ERMI was a valuable starting point for the development of a standard for expressing licensing terms but that it required considerable further development in order for the proposals to be implementable in a standard way by integrated library systems.. To meet requirements of extensibility and interoperability, the report found that licensing terms required organising into an ontological structure and proposed “ONIX for Licensing Terms” a simple events-based rights model based on ONIX and the work of the <indecs> project.

Following a proof of concept project jointly funded by JISC and the Publishers Licensing Society (PLS), an initial specification was developed for ONIX for Licensing Terms. This is currently being further developed in two JISC-funded projects under the PALS 2 Metadata and Interoperability call. A joint working party of EDItEUR, DLF, NISO and PLS has now been set up to monitor and provide international input to the further development of ONIX for Licensing Terms.

We believe that ONIX for Licensing Terms will be a valuable format for expressing the licensing terms attached to library use of e-books and enabling them to be incorporated into library electronic resource management systems.

8 Recommendations

- A properly-controlled evaluation of the current testbed be carried out, with standardised test activities and close analysis of results. The testbed has been used in libraries related to EPICentre and its overall value could be further evaluated through use in a number of libraries with differing systems environments. This would act as a major input to a requirements gathering process for a production service.
- A full set of detailed user requirements should be developed for a robust service that could be deployed in working libraries. The requirements process for TIME focused on standards and other issues relevant to the conceptual development of the testbed and was not intended to cover the full range of operating requirements that would be found in a working environment.
- The feature-set could be extended to cover ingest of records by individual libraries. At present, the testbed does not support local ingest: this might be desirable for a working service.
- Synergies with Copac should be investigated to determine if there is an opportunity for developing a union catalogue for e-books that would serve the dual purpose of identifying availability of ebooks across a range of libraries, and providing a collection of good-quality records for libraries acquiring ebooks to use in their own catalogues.
- The issue of supporting the later enhancement of source metadata by third parties (to create a “metadata pipeline”) should be investigated. Specifically, publisher ONIX data may be enhanced by third parties to meet the requirements of MARC or LOM users (or other, future, specialized formats). Onix for Licensing Terms should be considered if rights expression is required. The SOA-based architecture of the transformation platform lends itself to this approach.
- As definitions of an ebook can cover many different types of digital content, any future system should consider expansion to any specialised digital content. This will help facilitate the use of electronic content metadata for the widest range of resources.
- The current testbed should be further developed and re-engineered as necessary for intensive use as a production service. It has been developed using a set of technologies that were appropriate to testing the fundamental concepts, but are not intended to provide a robust delivery environment. The team has now developed systems using other tools that have recently become available and these would offer several advantages for a production version.
- The ebook metadata issue should be scaled and scoped. At present, it is not clear how many ebooks are published, how many publishers there are, what metadata is produced and what the quality of that metadata is.
- Based on our observation of publishers' practice in preparing metadata for ebooks, emphasis should be put onto encouraging publishers to improve the quality of their metadata provision by adopting and better using the ONIX format.
- Publishers of the existing books-in-print databases such as Nielsen Bookdata and Books in Print be approached and encouraged to carry comprehensive e-book product and availability information. This should include titles only available through collection aggregated by intermediaries and well as those available from directly from publishers.

9 Appendix: The Contextual Ontologyx Architecture abstract metadata framework

The COA MetaModel

The abstract metadata framework of the Contextual Ontologyx Architecture

Document History	
Title	The COA MetaModel
Principal Creator	Godfrey Rust
Contributors	Chris Barlas, Stephen Bayliss, Mark Bide, Martin Dow, Paul Hatcher, Steffen Lindek, Daniel Mahler, Niels Rump
Publisher	Rightscom Ltd
Copyright owner	© 2006 Rightscom Ltd
Version	0.58
Distributed to	Rightscom, NDA Clients
Editorial Status	Draft
Commercial Status	CommercialInConfidence
Date of Publication	2006-04-18
Changes in this version	<p>Add section 1.4 on COAX schema.</p> <p>Add further examples of PrimaryEntities</p> <p>Amend para on ContextualRoles (remove Relator and Verb as this is confusing)</p> <p>Section 3: Add copy to Relators section</p> <p>Section 7: Remove Class/Quality/Characteristic from Attributes: covered adequately at this level by Category. Add Datatype column to table.</p> <p>Add example (section 8)</p>
Revisions required	<p>URLs for references in 1.1 required.</p> <p>Compelte example (section 8)</p>

1 Introduction

The COA (**Contextual Ontology Architecture**) is designed to support integrated and extensible systems development and interoperability between systems. The **COA MetaModel** is a generic ontology-based metadata framework comprised of a set of defined types of Entity and Attribute, and the Relators which link them within a “contextual” model structure.

1.1 Background

The COA MetaModel is a proprietary data model which has its origins in the development of the **<indecs>** metadata framework (1998-2000). The subsequent development of COA has been influenced and advanced by a number of other metadata and identifier schemes and projects: the bibliographic initiative **FRBR**, **ABC-Harmony**, the **Digital Object Identifier**, **ONIX**, **DDEX (formerly MI3P)** and the **MPEG-21 Rights Data Dictionary**, the last three of which are supported by ontologies based on the COA. The COA MetaModel has a good deal in common structurally with the CIDOC Content Reference Model (**CRM**), although the development of the two has been entirely independent. The COA MetaModel is the upper ontology of Rightscom’s ontology **Ontology_X**.

1.2 Extensions to the MetaModel

The COA MetaModel has a number of Extensions which provide further specialized capability. These are detailed in separate documents.

1.3 Relationship to the COA Conceptual Class Model (CCM)

The COA CCM¹ is a UML conceptual class model which is the basis of an Object-Oriented (OO) implementation of a data management system based on the COA MetaModel. The Top Level of the CCM is supported by the COA MetaModel. For each COA MetaModel Extension there is a corresponding Extension to the CCM.

1.4 Relationship to the COA XML Message Schema (COAX)

The COAX schema is an XML Message schema based on the COA MetaModel. It provides an XML representation of the COA MetaModel structure to which any other schema can be mapped. Its function is to support interoperability through transformation to and from other schemas, or for use as a means of preparing data to be ingested into a COA-based system.

1.5 Conventions of this document

Terms in the COA MetaModel are identified with English language Headwords. Headwords are normally human-readable Identifiers (that is, they are unique within the COA). By convention, Headwords are single “camel-case” text strings with an initial capital letter. Illustrative examples are shown in blue font.

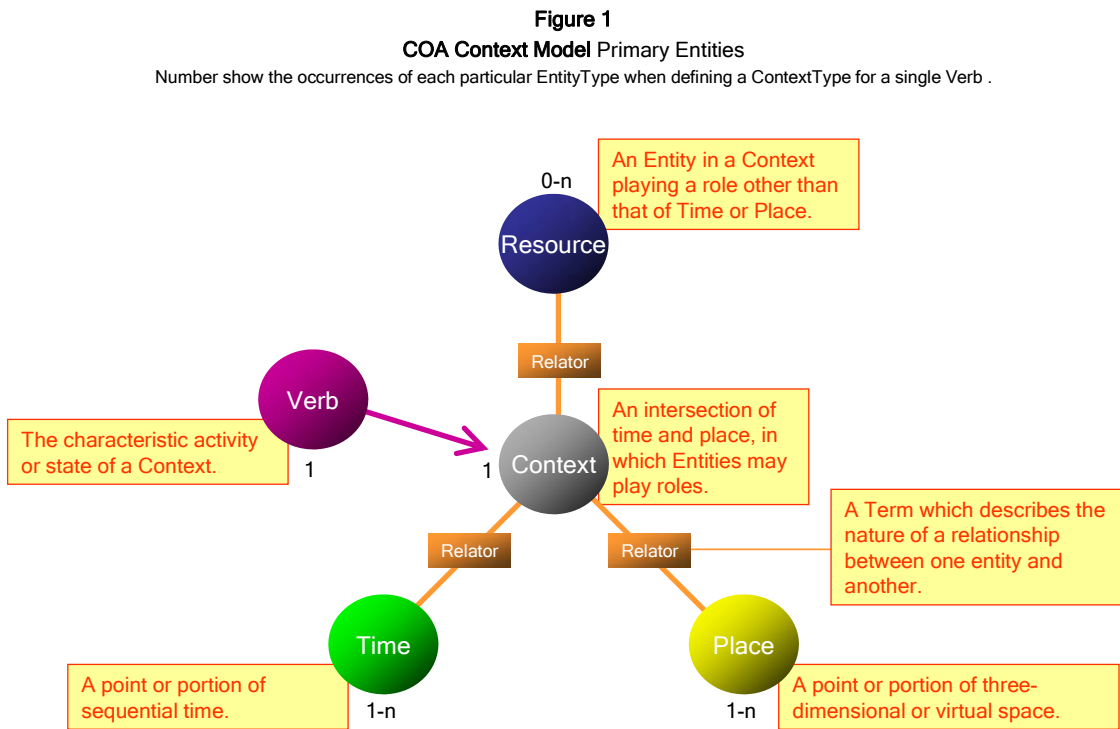
¹ See separate document *COA Conceptual Class Model (CCM)*

2 COA Context Model

The COA has a relatively simple primitive structure called the **COA Context Model**.

2.1 Primary EntityTypes

Each entity belongs to at least one of the six primary classes of Entities (**EntityTypes**) which form the basic components of the Context Model. These are illustrated and defined in Figure 1 and Table 1.



Context therefore has a specific meaning in COA. There are two primary types of Contexts: **Events** (in which something is recognized as changing) and **States** (in which nothing is recognized as changing).

COA semantics are based on the principle that *meaning is derived from the particular functions which Entities fulfil in Contexts*. An Entity retains its distinct *identity* across any number of Contexts, but its attributes and roles (and therefore its classifications) change according to the Contexts in which it occurs.

Verbs are therefore the most influential terms in an ontology, and nouns, adjectives and linking terms such as relators all derive their meanings, ultimately, from contexts and the verbs that characterize them. The contextual approach is used in COA as the basis for process, system and message design, and for the definition of terms required in each of those disciplines.

Table 1 COA Primary EntityTypes Definitions and example			
COA EntityType	Definition	Card ² .	Examples
Time	A point or portion of time.	1-n	October 4th 2003 at 3.35pm 1997 The Nineteenth Century 1939-1945 1997-03-14:23-14-11 Yesterday Always Circa 1554 Before 2005-01-01 During March 2001
Place	A point or portion of three-dimensional or virtual space.	1-n	Belgium San Diego, CA 15 High Street, Woking, Surrey, UK Everywhere johnsmith999@hotmail.com Tel:020-8567-1047 Outside London
Context	An intersection of Time and Place in which Entities may play roles.	1	Canada in 1950 Next Wednesday in John's house The Battle of Waterloo Having breakfast at Tiffany's Somewhere, Sometime Here, Now Always, Everywhere
Resource	An Entity in a Context playing a role other than that of Time or Place.	0-n	A maple tree Ludwig van Beethoven The novel "Moby Dick" The Holy Grail My computer Page 322, column 2 of book, ISBN:019861182X A fingernail
Relator	An Entity that describes the nature of a relationship between one Entity and another.	4-n	HasAuthor IsDateOfPublicationOf HasFormalTitle IsPlaceOfMovingFrom IsA
Verb	The characteristic activity or state of a Context.	1	Contextualize (the Verb characterizing a Context) Make Have Create Publish

² Cardinality shows the number of occurrences of each particular EntityType which must occur when defining a ContextType for a single Verb .

2.2 SubTypes of EntityTypes

EntityTypes may be specialized to any level of granularity. For example:

- [Creator](#), [JpegFile](#), and [Automobile](#) are examples of ResourceRoles (subclasses of Resource);
- [Territory](#), [WebLocation](#) and [CityOfPublication](#) are examples of PlaceRoles (subclasses of Place);
- [CopyrightYear](#), [CalendarMonth](#) and [TimeOfCreation](#) are examples of TimeRoles (subclasses of Time).
- [Performance](#), [Ownership](#) and [TranslatingEvent](#) are examples of ContextTypes (subclasses of Context);
- [HasAuthor](#), [IsA](#) and [GreaterThanOrEqualTo](#) are examples of SubRelators of Relator; and
- [Create](#), [Photocopy](#), [TranslateIntoFrench](#) and [KeepInFridge](#) are examples of Verbs .

The development of a COA ontology (see 6.1) is based on the development of a specialized hierarchy of EntityTypes.

2.3 ContextualRoles

Three of EntityTypes - Resource, Time, and Place - are known as the **ContextualRoles**, as they represent roles which may be played by independent Entities in Contexts. One Entity may play different roles the same or different Contexts. For example,

- the same individual may play the Resource Role of [Creator](#) in one Context, of [User](#) in another and of [Owner](#) in another, or may play all three roles in the same Context.example,;
- the year [1963](#) may play the TimeRole of [YearOfDeath](#) for [Aldous Huxley](#) and [YearOfFirstPublication](#) for the Beatles' single *She Loves You*,
- the territory [France](#) may be the [PlaceOfResidence](#) of [Samuel Beckett](#) for a particular period of time, and the [TerritoryOfUse](#) for a television programme under the terms of a license agreement.

2.4 Disjunction of EntityTypes

Any Entity may play ResourceRoles. However, TimeRoles, PlaceRoles, ContextTypes, Relators and Verbs are mutually and universally *disjoint*. that is, the same Entity cannot ever play more than one of these roles, even in different Contexts.

3 Relators

3.1 Primary COA Relators

Each Entity playing a ContextualRole has a direct relationship with the Context, and also a relationship with *each of the other Entities in the Context*. In the COA model, it is through Contexts that all relationships are established. For example, when a group of people attend a meeting, they establish particular relationships with one another, the various subjects, papers, presentations, coffee mugs and biscuits which were deployed at the meeting, and the time and place at which it occurred.

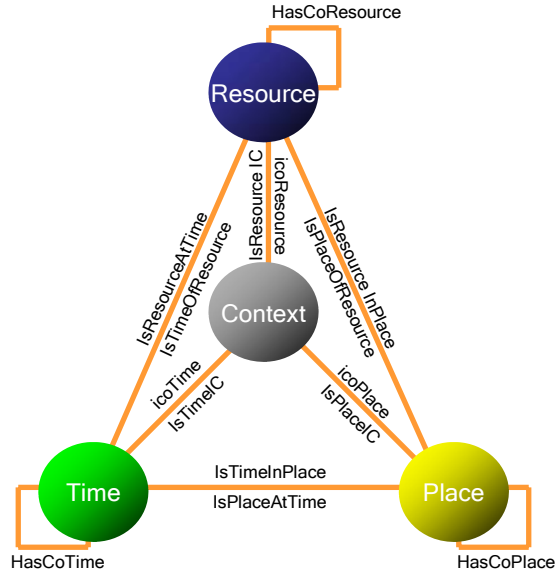
Each of these types of relationship has a different character according to the ContextualRole(s) played by the Entities. Each type of relationship is therefore named with a specific Relator. Each COA Relator has a reciprocal (for example, [icoResource](#) and [IsResource IC](#)) so that each relationship may be stated from the point of view of either Entity.

The Primary COA Relators are shown in the matrix in Table 2. Note that [ico](#) is an abbreviation for [IsContextOf](#) and [IC](#) is an abbreviation for [InContext](#).

Table 2 Primary COA Relators				
	Context	Resource	Time	Place
Context	(none)	icoResource	icoTime	icoPlace
Resource	IsResourceIC	HasCoResource	IsResourceAtTime	IsResourceInPlace
Time	IsTimeIC	IsTimeOfResource	HasCoTime	IsTimeInPlace
Place	IsPlaceIC	IsPlaceOfResource	IsPlaceAtTime	HasCoPlace

Figure 2 shows these Relationships figuratively.

Figure 2
COA Context Model Relationships
 Each line indicates a named two-way relationship



The Relators between the role-playing Entities and the Context are called **ContextualRelators**. These are shown in blue font in Table 3. The Relators between the role-playing Entities are called **NonContextualRelators**. These are shown in violet font in Table 3.

This set of Relators supports the expression of every possible relationship within a particular type of Context. For example, in a Context **C** which contains Resource **R**, Time **T** and Place **P**, the following statements are true:

- C icoResource R
- C icoTime T
- C icoPlace P
- R IsResourceIC C
- R IsResourceAtTime T
- R IsResourceInPlace P
- T IsTimeC C
- T IsTimeOfResource R
- T IsTimeInPlace P
- P IsPlaceC C
- P IsPlaceOfResource R
- P IsPlaceAtTime T

If Context **C** contains multiple occurrences of each Role (for example **R1**, **R2**, **T1**, **T2**, **P1**, **P2**), then the above statements apply to each, and in addition the following statements are true:

- R1 HasCoResource R2
- R2 HasCoResource R1
- T1 HasCoTime T2
- T2 HasCoTime T1

P1 HasCoPlace P2
P2 HasCoPlace P1

3.2 Interdependency of ContextualRelators and Non ContextualRelators.

One practical, ontological consequence of the Relator matrix is that Contextual relationships can be derived from NonContextual ones, and vice versa.

For example, if

C icoResource R
C icoTime T

then

R IsResourceAtTime T.

It is also true in reverse, so that if

R IsResourceAtTime T.

then there exists a Context C such that

C icoResource R
C icoTime T

4 Qualities of ResourceRoles

4.1 Adjectives and Properties

ResourceRoles may also be described in terms of qualities of the Entity itself in the form of corresponding **Adjectives** or **Properties**.

An Adjective is defined as “an adjectival Characteristic of a Resource”³. As an alternative to defining a ResourceRole as a Class, its characteristics may be described using an Adjective. For example, a **Creator** may be described by the adjective **Creating**, and a **Creation** by the adjective **Created**.

A Property⁴ is defined as “a nounal Characteristic of a Resource”. As an alternative to defining a ResourceRole as a Class, its characteristics may be described using a Property. For example, a **Creator** may be described as possessing the Property **Creativity** and a **Creation** as possessing the Property **Createdness**.

³ When defining a term in COA the normal lexicographic rule against defining a term with reference to its own name does not apply, because COA Headwords have no semantic value. So, for example, the term Adjective may be defined as “an adjectival Characteristic of a Resource”, as the semantic content lies only in the use of the word “adjectival” in the definition. This Term is called *Adjective* for convenience of reference: it might just as well be called *Foo* or *Term12345*.

⁴ Note that the term *Property* in COA does not have the same meaning as it commonly has in some other domains, such as the Resource Description Framework (RDF), where the name is used to describe what COA calls a Relator. All terms in the COA MetaModel, or any ontology built on it, are defined in terms of the COA Primary Entities themselves. The co-occurrence or opposition of meaning with terms of the same name in another scheme is irrelevant to their meanings in either scheme.

The distinction between Adjectives and Properties is in their linguistic form rather than in underlying semantics. The following groups of statements are therefore equivalent:

- X IsA Creator
- X Is Creative
- X Has Creativity

- X IsA Creation
- X Is Created
- X Has Createdness

4.2 Temporal modes of Qualities

Each ResourceRole may have corresponding Adjectives and Properties describing its past (**Historic**), present (**Current**) or possible future (**Potential**) Qualities. For example, the ResourceRoles **Creator** and **Creation** have the attributed Qualities shown in Table 3.

Table 3 COA Qualities Temporal modes and examples		
QualityType	Creator	Creation
HistoricAdjective	FormerlyCreative	Created
CurrentAdjective	Creative	BeingCreated
PotentialAdjective	PotentiallyCreative	Creatable
HistoricProperty	HistoricCreativity	HistoricCreatedness
CurrentProperty	Creativity	Createdness
PotentialProperty	PotentialCreativity	Creatability

5 Context Families

The ContextModel can be applied to any Verb concept to generate a set of ContextualRoles and Qualities to make up a **ContextFamily** embodying the meaning of the Verb in each of these roles. So, for example, the Verb **Create** has a ContextFamily containing the Terms shown in Table 4:

Table 4 ContextFamily Example					
Entities	ContextType	ResourceRole	ResourceRole	TimeRole	PlaceRole
	CreatingEvent	Creator	Creation	TimeOfCreating	PlaceOfCreating
Relators					
CreatingEvent	(n/a)	icoCreator	icoCreation	icoTimeOfCreating	icoPlaceOfCreating
Creator	IsCreatorIC	HasCoCreator	IsCreatorOf	IsCreatorAtTime	IsCreatorInPlace
Creation	IsCreationIC	IsCreatedBy	HasCoCreation	IsCreationAtTime	IsCreationInPlace
TimeOfCreating	IsTimeOfCreatingIC	IsTimeOfCreatingBy	IsTimeOfCreatingOf	HasCoTimeOfCreating	IsTimeOfCreatingInPlace
PlaceOfCreating	IsPlaceOfCreatingIC	IsPlaceOfCreatingBy	IsPlaceOfCreatingOf	IsPlaceOfCreatingAtTime	HasCoPlaceOfCreating
Qualities					
HistoricAdjective	(n/a)	FormerlyCreative	Created	(n/a)	(n/a)

CurrentAdjective	(n/a)	Creative	BeingCreated	(n/a)	(n/a)
PotentialAdjective	(n/a)	PotentiallyCreative	Creatable	(n/a)	(n/a)
HistoricProperty	(n/a)	HistoricCreativity	HistoricCreatedness	(n/a)	(n/a)
CurrentProperty	(n/a)	Creativity	Createdness	(n/a)	(n/a)
PotentialProperty	(n/a)	PotentialCreativity	Creatability	(n/a)	(n/a)

5.1 The Context Model as a ContextFamily

The elements of the ContextModel comprise the ContextFamily for the Verb [Contextualize](#).

6 Specialization of EntityTypes

6.1 COA Ontology

One of the functions of the COA is to supporting the definition of terms through specialization from the ContextModel. The elements of a data model or system built on the COA MetaModel comprise a **COA Ontology**. A COA Ontology includes all of the elements of the COA MetaModel, supplemented by any number of specializations. Rightscom's COA Ontology is called **Ontology_X**⁵.

6.2 Inheritance hierarchies

A COA ontology is built by creating hierarchies of specialized classes of Entity in the form of SubClasses or SubRelators to the Primary EntityTypes. For example:

Entity HasSubClass Resource
 Resource HasSubClass Output
 Output HasSubClass Creation

5 Further information on Ontology_X can be found in the [forthcoming] documents *A Guide to Ontology_X* and *Implementing COA with Ontology_X*.

establishes this hierarchy:



in which each subclass inherits the constraints of its parent. Entities are specialized in a COA ontology by adding further constraints (for example, an **Output** is a **Resource** which did not exist at the beginning of a Context, but exists at the end of it). Those attributes may be expressed in formal relationships or otherwise introduced as primitive concepts in the textual definition of a term⁶.

6.3 Principal Specialized Roles

In implementations of COA, a number of specializations (**Principal Specialized Roles**) are commonly implemented at the primary level. The five most common are shown in Table 5.

Table 5 Principal COA Specialized Roles Definitions and examples			
Specialized Role	Parent Role	Definition	Examples
Agent	Resource	A Resource that is responsible for the activity in an Event.	Ludwig van Beethoven The Rolling Stones Warner Music The University of Koblenz Mickey Mouse
Event	Context	A Context in which there is some acknowledged change in the attributes of any Entity.	The Battle of Waterloo A performance of "The Marriage of Figaro" A wedding A making of an agreement The sending of an email
State	Context	A Context in which there is no acknowledged change in the attributes of any Entity.	The life of Winston S Churchill The availability status of a product A marriage A conflict of truth value between two statements
PointInTime	Time	A Time represented as a single point in time (as opposed to a Period with a StartTime and EndTime). While theoretically of no duration, all declared PointInTimes actually have a Duration (for example, of a Second or a Year) within which the actual point occurs.	October 4th 2003 at 3.35pm 1997 The Nineteenth Century Last Wednesday
Period	Time	A Time represented as a range between two other Times (with measurable Duration, as opposed to a PointInTime).	1939-1945 12.00 to 12.45

In particular, **Agent** is commonly shown as a Primary COA Entity, and so the most common representation of the COA Context Model is the specialized form shown in Figure 2. Although in this form Agent is shown alongside Resource, it remains ontologically a subclass of Resource, and any Entity which is an Agent may fulfill other Resource roles.

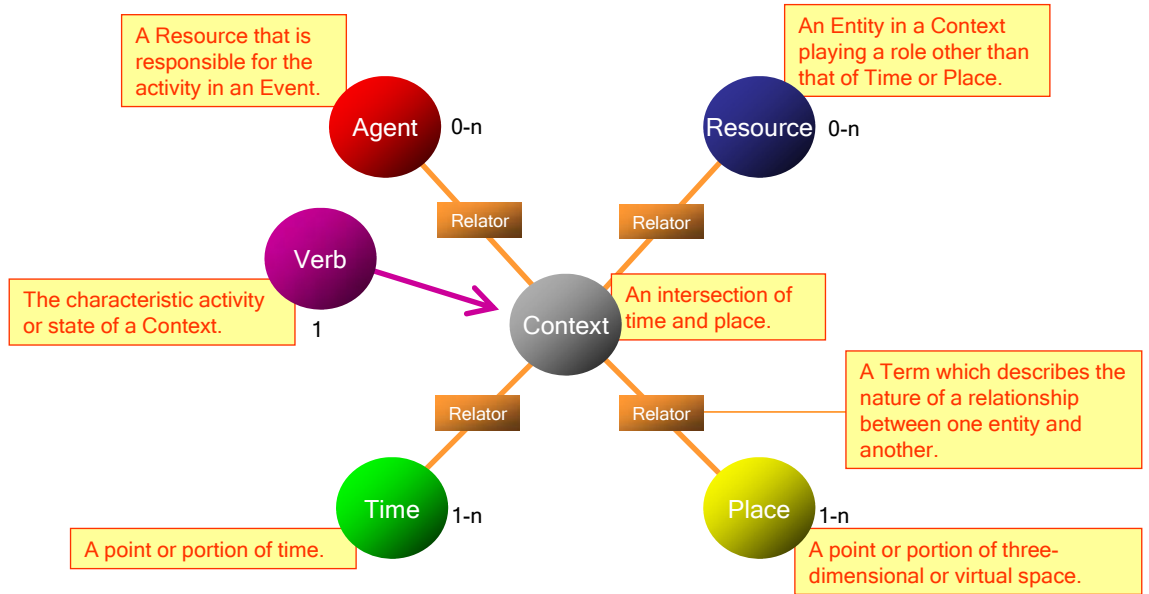
⁶ Further details and examples of the methodology of specialization of a COA Ontology will be found in *A Guide to Ontology_X*.

The Common Specialized Model of the Context Model, showing Agent as if it were a Primary Entity Type but not including the other PrincipalSpecialized Roles is shown in Figure 3. This is the version on which the COA CCM and COAX schema are based.

Figure 3

COA Context Model Common Specialized Model

Number show the occurrences of each particular Entity Type when defining a Context Type for a single Verb



6.4 Formal specializations

The COA can incorporate Relators and other constructs which express axioms of formal logic or mathematics. The COA MetaModel relies on a number of these, shown in Table 6.

Table 6 COA Metamodel Formal Axioms Definitions and equivalents		
Class	COA Definition and	Equivalent terms in RDF, OWL, RDFS
Class	An abstraction representing a group of entities with one or more common attributes.	<i>Owl:Class</i>
Relator	An abstraction expressing a relationship between two entities (the Domain and Range of the Relator).	<i>Rdfs:Property</i>
Relator <i>Reciprocal</i>	COA Definition and <i>equivalent terms</i> in other formalisms	
HasDomain <i>IsDomainOf</i>	The Relator that joins a Relator to the Class of its Domain. For example IsNameOf HasDomain Name	<i>Rdfs:Domain</i>
HasRange <i>IsRangeOf</i>	The Relator that joins a Relator to the Class of its Range. For example IsSubClassOf HasRange Class	<i>Rdfs:Range</i>
IsSubClassOf <i>HasSubClass</i>	A Relator between two Classes in which the class extension of the Range is a subset of class extension of the Domain. For example: Performance IsSubClassOf Creation	<i>rdfs:subClassOf</i>
IsSubRelatorOf <i>HasSubRelator</i>	A Relator between two Relators (R1 and R2) in which all instances of the Domain and Range pair of R1 must also be a valid Domain and Range pair or R2, but not vice versa. For example: IsTitleOf IsSubRelatorOf IsNameOf	<i>rdfs:subPropertyOf</i>
IsA <i>HasInstance</i>	A Relator between an entity and a class of which it is a member. For example: France IsA Country	
IsDisjointWith <i>IsDisjointWith</i>	A Relator between two Classes whose extensions have no individuals in common. For example: Time IsDisjointWith Place	<i>owl:disjointWith</i>
IsUnionOf	A Relator between a Class and an enumerated set, where the Domain is a class for which the class extension contains those individuals that occur in at least one of the class extensions of the class descriptions in the set comprising the Range. For example: Descriptor IsUnionOf {Name, Annotation}	<i>owl:unionOf</i>
IsReciprocalOf <i>IsReciprocalOf</i>	A Relator between a Relator and the Relator which expresses its reciprocal or inverse. For example: IsSubClassOf IsReciprocalOf HasSubClass	<i>owl:inverseOf</i>

7 Attributes

The COA MetaModel is extended to include seven types of Attribute which may belong to any Entity. These are shown in Table 7.

A COA AttributeType is a specialized ResourceRole. The set of COA MetaModel AttributeTypes has been developed to represent a comprehensive set with which all textual and numeric attributes and categorizations of Entities in a schema can be modelled at a convenient level.

Table 7 COA Metamodel AttributeTypes Definitions and examples			
COA Attribute	Definition	Examples of SubClasses	Datatype
Name	A referential designation of an Entity which is capable of being Written.	Title, PartyName, PlaceName, EventName	Text string
Identifier	A Name which is unique within its namespace.	ISBN, ISRC, ProprietaryId, CopyrightDate ⁷	Text string
Annotation	A descriptor which provides descriptive or other information or opinion about an Entity.	Description, Comment, Note, Review	Text string
Descriptor	A Name or an Annotation of an Entity. Descriptor is a "union" of Name and Annotation for those lexical attributes which are both referential and descriptive.	<i>(A parent attribute type only used when a schema fails to determine whether a string value functions as a Name or an Annotation).</i>	Text string
Category⁸	A Class to which the Entity belongs or a Quality which it possesses.	Format, Genre, NameType, Color, Status, Continuity, Validity	Ontology Headword
Flag	A Category whose Instances are limited to the Boolean values (True/False).	IsAvailable, IsBlackAndWhite, IsAuthorized	Boolean
Quantity	A characteristic of an Entity expressed as a numerical measure.	Speed, Duration, Percentage, NoOfUnits	Number or formula

7.1 Attribute constraints

An Entity may, in principle, have any number of attributes of any AttributeTypes: the limitations are imposed by the identification of allowed values which apply to any particular EntityType in a COA Ontology. For example, in a given implementation a Place might be given any number of Names, but only from the classes which are members of the allowed value set of [PlaceNameTypes](#) according to the particular ontology being used.

7.2 Attributes of Attributes

Because they are Resources, Attributes may have Attributes of their own to any level of granularity. For example, a Category may have a Name, and that Name may have an Annotation (such as a comment on the Name), and that Annotation may have a Quantity (for example, the number of words it contains), and so on.

⁷ Note that a Date is one form of Identifier of a Time.

⁸ A more detailed analysis of Category into AttributeTypes of *Class, Flag, Characteristic, Quality* and *Quantity* is incorporated into the formal ontological expression of the COA MetaModel. This models the phenomenon that the same characteristic (eg the height of a person) may be expressed as a Class (TallPerson), a Flag (IsTall), a Quality (Tall) or a Quantity (> 1.92m)

8 MetaModel example

Table 8 shows a simple example of how the COA MetaModel may be applied to conventional descriptive metadata (in this case for a hardback book), using a combination of Entity Relationships and Attributes. The COAX schema and CCM model provide more detailed structures for the representation of this data, respectively in XML Message and Object Model form. In all cases, the specific allowed values (shown here in blue font) would be taken from a COA-based ontology. The example is shown in a “nested” hierarchy not unlike a nested XML tree.

When an Entity is referenced in an XML Message or a database it would have its own unique internal identifier with which Relationships to other Entities would be established. These are not shown here in the interests of simplicity and readability. Where an Entity-to-Entity Relationship is shown, only a summary name is given in the Value column. Each Entity will, of course, have its own further set of Attributes and perhaps Entity Relationships. Examples of five of these (with cells highlighted in pale blue) are given in further tables below. Other examples of linked Entities highlighted in light yellow do not have such expanded examples (again, for simplicity). It should be seen that using the MetaModel any kind of Attribute or Entity Relationship can be modelled, provided there are appropriate supporting ontology values.

Entity 1 (a Resource)

COA EntityType or AttributeType	Relator (for Entities)	Value
Category		Resource
Category		COAPrimaryType
Name		Words And Rules
Category		PrimaryTitle
Category		TitleType
Identifier		1
Category		SequenceNumber
Name		The Ingredients Of Language
Category		Subtitle
Category		TitleType
Identifier		2
Category		SequenceNumber
Identifier		0297816470
Category		ISBN
Category		ResourceIDType
Annotation		Dazzling...words can hardly do justice to the superlative range and liveliness of Pinker’s investigations.
Category		ReviewQuotation
Category		AnnotationType
Agent	HasAuthor	Agent [Name=Robert Winder, Independent]
Category		HardbackBook
Category		ResourceType
Quantity		14.99
Category		PublishedRetailPrice

Category		QuantityType
Category		Pounds
Category		Currency
Agent	HasAuthor	Agent [Name=Stephen Pinker]
Resource	IsPartOf	Resource [Series, Name=Science Masters]
Context	IsPublicationIC	Context [Name=UK Publishing Event for Words and Rules]
Time	HasDateOfPublication	Time [Date=1999]
Place	HasPlaceOfPublication	Place [Territory, Name=UK]

Entity 2 (an Agent)

Category		Agent
Category		COAPrimaryType
Name		Stephen Pinker
Category		FullName
Category		PersonalNameType
Name		Stephen
Category		NameBeforeKeyName
Category		PersonalNamePart
Name		Pinker
Category		KeyName
Category		PersonalNamePart
Name		Pinker, Stephen
Category		PersonalNameInverted
Category		PersonalNameRepresentationType
Category		Author
Category		CharacteristicRole
Agent	IsProfessorOfPsychologyOf	[Name=Center for Cognitive Neuroscience, MIT]
Resource	IsAuthorOf	[Name=The Language Instinct]
Resource	IsAuthorOf	[Name=How The Mind Works]