002068

ALVIS
Superpeer semantic Search Engine

STREP / IST

Deliverable D3.1 Month 24 December 2005 Metadata frameworks for peer capability descriptions and semantically enriched documents

Title of contract	ALVIS - Superpeer Semantic Search Engine
Acronym	ALVIS
Contract number	IST-1-002068-STP
Start date of the project	1.1.2004
Duration	36 months, until 31.12.2006
Document name	Deliverable D3.1 Month 24 December 2005– Metadata frameworks for peer capability descriptions and semantically enriched documents
Date of preparation	January 23, 2006
Author(s)	Mike Taylor
Coordinator of the deliverable	Index Data Phone : +45 3341 0100 Fax : +45 3341 0101 Email: mike@indexdata.dk
Document location	http://project.alvis.info/copies/d3-1.pdf

Project funded by the European Community under the "Information Society Technology" Programme (2002-2006)

Contents

Abstract

The Alvis project requires machine-readable representations of two crucial kinds of information: the capabilities of individual peers in an Alvis network, and the semantically annotated documents that are indexed, searched for and retrieved. Metadata formats representing both peer capabilities and enriched documents are here described. They use XML as their concrete syntax, and are described by W3C XML Schemas. Example documents of both kinds are provided.

Peer-capability records typically include the peer's human-readable name and unique identifier, a set of addresses it can be contacted on, an indication of what subject areas its data falls into, specifications of its support for various parts of Alvis functionality and statistics about its performance.

As documents are passed through the Alvis pipeline, being progressively enriched prior to indexing, they are successively represented by each of a family of closely related formats, known collectively as the Enriched Document formats: First, Acquisition Format, which is what the various Document Source modules produce; then Linguistic Format, which is a superset including information produced by the Linguistic Analyser; then Relevance Format, a further superset including information produced by the Document Probability package. It is this last and most complex form of the document that is eventually fed to the indexer to facilitate subsequent semantically rich queries.

Chapter 1

Introduction

The EU-funded Alvis[1] project runs for three years from 2004-2006, and is tasked with building a semantic peer-to-peer search engine. Alvis's unique P2P architecture involves two fundamentally different kinds of peer: "superpeers" (also known as "fat peers") which contain the subject-specific knowledge that users want to find, and which pass queries among themselves; and "infrastructure peers", which provide a route into the superpeer network by using lookups in a distributed hash table[2] to identify superpeers likely to provide a good starting point for a given query's journey.

WP4 (Distributed Search) is responsible for the infrastructure peers and the communication between them; a separate report[3] describes the interfaces whereby superpeers communicate with the network of infrastructure peers. WP3 is not directly concerned with the work at the infrastructure peer level. Since the metadata formats described by this document pertain only to superpeers, we use the simpler term "peer" throughout.

1.1 Peer Capabilities

In order for any peer-to-peer system to operate effectively, it is necessary for each peer to understand the capabilities of nearby peers, so that appropriate operations can be requested. In very simple peer-to-peer systems such as classic Gnutella, this is achieved trivially, because all peers have the same capabilities; or at least they vary only in very parameterisable ways such as their network bandwidth. In Alvis, however, different peers may vary substantially in their strengths and weaknesses; so a mechanism is needed by which any given peer can express those capabilities in a machine-readable form suitable for other peers to understand.

Task T3.1 (Network Node Metadata Framework) in Workpackage WP3 (Data Model and Standards) is to provide such a format. Since the Alvis proposal document was written, the consortium partners have adopted more consistent terminology, so that what the WP3 part of the proposal referred to as "nodes"

[1]**http://alvis.info/**

[2]Aberer et al. 2006, "Building a peer-to-peer full-text Web search engine with highly discriminative keys". http://infoscience.epfl.ch/getfile.py?recid=63674;mode=best

[3]Podnar 2005. "Implementation of the communication protocol between a superpeer and IR peer using Web Services".

are now called “peers”. Accordingly, Task 3.1 is now more properly defined as the provision of a peer-description metadata format.

This format is described in Chapter 3.

1.2 Enriched Documents

Within each individual peer that makes up an Alvis network, a great deal of semantic work is done. Documents may be added to a peer’s local database from a variety of sources, to become available for subsequent semantic searches.

As documents are added to an Alvis peer, they pass between a series of components that perform various analyses and transformations, each enriching the document in different ways. Since the components have been created by different partners, using different programming languages running on different platforms, it is necessary to define a rigorous shared format in which the documents, together with their associated enrichments, can be expressed for interchange between components.

Task T3.2 (Semantic Document Metadata Framework) in Workpackage WP3 is to provide such this format; or, rather, family of formats. Since the Alvis proposal document was written, the consortium partners have adopted more explicitly descriptive terminology, so that what the WP3 part of the proposal referred to as “semantic documents” are now called “enriched documents”. Accordingly, Task 3.2 is now more properly defined as the provision of a metadata format for enriched documents.

This format is described in Chapter 4.

It is important to understand that the communication described here is all within the context of single Alvis peer. Communication between peers expresses an entirely different set of concepts, and consequently uses an entirely different format, to be described elsewhere as part of Workpackage 4.

Chapter 2

Design Issues

Creation of formats such as those described herein must be based on a coherent set of principles. With those principles in mind, format design consists of two essentially independent decisions – one semantic and one syntactic. The former involves deciding what information needs to be in the records; the latter with how that information is encoded.

The real work is done in the semantic area: deciding which aspects of peers' functionality and of document analysis should be described, what elements should represent that information, how the elements should be related to each other, what prescribed vocabularies they can draw their values from, etc. In contrast, the syntactic decisions are relatively straightforward, so we deal with these in this chapter.

2.1 Principles for Designing the Formats

In designing the Alvis metadata format for enriched document, we have been guided by four principles:

Clarity We make all element names explicit, even at the expense of making them longer, so that the documents are so far as possible self-describing. We can revisit the names of the elements and attributes towards the end of the project, making them shorter if necessary to improve efficiency; but until and unless profiling shows document size to limit performance, clarity is the overriding concern.

Structure Peer-capability records and enriched documents should consist of clearly delineated sub-records, each generated by a specific piece of software, and each further subdivided where appropriate.

Generality We prefer to name elements and attributes according to what they are for rather than according to some implementation-specific detail such as the name of the particular program that generates them.

Simplicity We introduce no more complexity into the record structure than is necessary to fulful the first three principles.

2.2 Choice of Meta-Format

In choosing a syntax with which to represent peer-description records and enriched documents, the desiderata are as follows, in roughly descending order of preference:

1. The syntax should be easy for computers to understand.
2. It should be easy for humans to read.
3. It should be easy for humans to write, as peer-description records in particular may often be hand-crafted rather than automatically generated.
4. It should be compact, so as to minimise network traffic.

Of these criteria, the last is regarded as a luxury for peer-description records, but rather more important for enriched documents, as the former will be of near-negligible size compared with the latter, and will also be transferred much less often.

Of the remaining three criteria, 2 and 3 are correlated, and 1 is not incompatible with either. These criteria might easily enough be satisfied by a custom-designed format, but practical concerns dictate that it is preferable to use an "off-the-shelf" metasyntax, so that peers implemented in various languages can take advantage of existing parsers rather than having to use specially developed code for parsing.

Three strong candidate metasyntaxes are in widespread use today. Two of these, SGML[1] and XML[2], are very similar to each other in most respects. SGML is the older and more powerful of the two; XML is a deliberately cut-down descendant of SGML because its developers felt that simplicity and uniformity were more important that expressiveness. The third contender is YAML[3], a more elegant format that places more emphasis on human-readability and is also rather less verbose.

We regretfully decided to pass up the most technically appealing contender, YAML, on the pragmatic grounds that it is much less widely implemented than either of the others, and also because it does not yet have a constraint language.

In recent years, the pace of development of XML utilities has hugely outstripped that of its rivals, including SGML, and the addition of extrinsic functionality such as XML Namespaces[4] and XPath[5], means that the expressive power of the entire XML toolset now comfortably exceeds that of SGML. Accordingly, we selected XML as the metalanguage for Alvis peer-description records and enriched documents.

2.3 Choice of Constraint Language

The structure of XML records can be constrained using a number of different constraint languages, including the older DTDs (document type definitions) inherited from SGML, the newer and more complex XML Schema[6], and its

[1]**http://www.w3.org/MarkUp/SGML/**
[2]**http://www.w3.org/XML/**
[3]**http://www.yaml.org/**
[4]**http://www.w3.org/TR/REC-xml-names/**
[5]**http://www.w3.org/TR/xpath**
[6]**http://www.w3.org/XML/Schema**

more intuitive but less widely deployed rival Relax NG[7].

These are all broadly equivalent in their ability to express the acceptable structure of XML documents, though with some important differences. In contrast to the older DTD technology, XML Schema and Relax NG both include support for XML namespaces and provide more control over the values that can be used for particular XML elements and attributes.

In the early versions of these specifications described in the milestone documents M3.1 (June 2004), M3.2 (December 2004) and M3.5 (June 2005), we preferred the use of DTDs, finding that their limitations were more than counterbalanced by their ubiquitous support in XML toolkits such as the widely-used libxml2[8] which, perhaps surprisingly does not fully support XML Schema at the time of writing. However, it has subsequently become apparent that support for XML namespaces is necessary for several possible uses for enriched records: for example, they must be namespaced in order to be returned from OAI repositories using OAI-PMH – a procedure beyond the scope of this report and not strictly necessary in a running Alvis network, but which we nevertheless do not wish to inhibit.

Accordingly, we provide in this document XML Schemas prescribing the format of peer-description records and enriched documents, and accompany this with prose specifying the controlled vocabularies of attributes that can take only a small number of of pre-defined values.

[7]**http://www.relaxng.org/**
[8]**http://www.xmlsoft.org/**

Chapter 3

Peer-Description Records

3.1 Sample Peer-Description Record

The following sample peer-description record serves as a motivating example for the discussion to follow.

```
<?xml version="1.0" ?>
<!-- $Id: peer-description.xml,v 1.3 2006/01/23 09:08:45 mike Exp $ -->
<peer xmlns="http://alvis.info/peer/"
      version="1.1"
      name="Mike Taylor’s Example Peer"
      id="anyOpaqueAndUniqueIdentifierAYFGDYHFGAS">
  <addresses>
    <address type="tcp" bandwidth="128">192.168.0.106:12368</address>
    <address type="mail">alvis@miketaylor.org.uk</address>
  </addresses>

  <subjectAreas>
    <subject scheme="ddc22">
      <code>560</code>
      <caption>Paleontology Paleozoology</caption>
    </subject>
    <subject scheme="ddc22">
      <code>570</code>
      <caption>Life sciences</caption>
    </subject>
    <subject scheme="lcsh">
      <code>QE731</code>
      <caption>Paleontology -- Mesozoic</caption>
    </subject>
  </subjectAreas>

  <support>
    <!-- query-type names are taken from a well-known enumerated set -->
    <query type="cql"><!-- See http://zing.z3950.org/cql/ -->
      <!-- List of document formats that can be searched with this language -->
      <searchFormat type="text/xml"/>
      <searchFormat type="text/plain"/>
      <searchFormat type="audio/mpeg"/>
      <index>creator</index>
      <index>title</index>
      <index set="http://z3950.org/dc/1.0/">date</index>
    </query>

    <query type="bag"/><!-- bag of words -->

    <query type="xpath"><!-- subset of XQuery: xpath=term -->
      <searchFormat type="text/xml"/>
    </query>

    <rank type="field">creator</rank>
    <rank type="field">title</rank>
    <rank type="algorithm">tfidf</rank>

    <!-- metadata subset of object that can be returned -->
    <subset type="id"/>
    <subset type="dc"/>
    <subset type="xpath"/>
    <subset type="fulltext"/>

    <recordFormat type="text/xml">
      <schema name="Dublin Core" tag="dc">
        <spec type="xmlschema">http://foo.com/dc.xsd</spec>
        <spec type="dtd">http://bar.com/quux/dc.dtd</spec>
        <spec type="relaxng">http://baz.com/dublinCore.rng</spec>
      </schema>
      <schema name="Text Encoding Initiative" tag="tei"><!-- ... --></schema>
      <schema name="Some Random Schema" tag="x-abc"><!-- ... --></schema>
    </recordFormat>

    <!-- MIME-types of the kinds of object that can be returned -->
    <recordFormat type="text/plain"><!-- ... --></recordFormat>
    <recordFormat type="application/x-pdf"/>
    <recordFormat type="audio/mp3"/>
  </support>

  <statistics>
    <statistic type="meanSearchTime">1.56</statistic>
    <statistic type="medianSearchTime">0.67</statistic>
    <statistic type="numberOfRecords">18398</statistic>
  </statistics>
</peer>
```

3.2 Discussion of the Peer-Description Record

Recall that the peers described by these records are the "superpeers" discussed in the architectural overview above, and not the infrastructure peers built in WP4. All of the communication described below is between superpeers, as queries make their way through the network in search of peers that can satisfy them. The format does not describe communication among infrastructure peers, nor between superpeers and infrastructure peers.

The peer-description record is discussed in detail here in order to elucidate the meanings of the elements and attributes that comprise it. In particular, where certain attributes may take only a small number of well-known values, these are enumerated and discussed.

The record has a top-level element `<peer>`. This, like all the other elements and all attributes, is in the namespace `http://alvis.info/peer/`.

We now discuss the attributes of the top-level `<peer>` element, then the four sections within the record: `<addresses>`, `<subjectAreas>`, `<support>` and `<statistics>`.

3.2.1 Namespace, Version, Name and Identifier

The top-level `<peer>` element of the peer-description record carries four attributes, all of them mandatory:

namespace The constant namespace URI `http://alvis.info/peer/`. This namespace qualifies all the elements and attributes of the record, indicating that they are the particular elements and attributes described in this document. It is an error to omit this attribute from a peer-description record, or to specify a different namespace URI.

version A two-faceted number of the form `major.minor` indicating the version of the peer-description record specification that a record adheres to. At the time of writing, the version number is 1.1 – in other words, this document describes version 1.1 of the peer-description format.

This attribute is specified so that Alvis peers that understand a recent version of the peer-description format may recognise and adapt to peer-description records conforming to earlier versions of the format. Changes to the format that merely add new information will be indicated by incrementing the minor facet of version number (e.g. from `1.1` to `1.2`): such changes are easily handled by well-behaved readers, since all version `x.y1` peer-description documents are also version `x.y2` documents for all values of `y2` greater than `y1`. If it is ever necessary to make incompatible changes, such as removing or renaming elements or changing the record structure, this will be indicated by incrementing the major facet of the version number (e.g. from `1.4` to `2.0`).

An alternative approach, sometimes used in XML-based data-representation formats, would have been to embed a version-number in the namespace, like so: `http://alvis.info/peer/1.1/`. This approach suffers from the problem that the same-named elements in documents using different namespaces – even if those namespaces differ only in the minor facet of the version numbers – are actually different XML elements. This seemingly pedantic point has significant practical ramifications when such records have to be handled by inflexible software libraries such as SOAP toolkits. Implementation experience clearly indicates that the approach we have taken here, separating namespace and version, is more easily implementable and leads to better forward- and backward-compatibility.

name This attribute holds a human-readable string to be displayed as the name of peer (in applications that show individual peers to users at all). Examples might include "Library of Congress Catalogue", "University of Portsmouth Research Papers", "Pipedreaming Music Soundtrack Demos" and "Mike Taylor's Totally Objective Music Reviews".

id This is an opaque cookie that uniquely identitifies the peer. It is not intended for humans to see, but only for computers to use in determining peer identity.

There is no single co-ordinating server on an Alvis network to hand out these identifiers, since such a server would constitute a single point of failure, an unacceptable constraint on a robust distributed system. Accordingly, the responsibility rests with individual peers and their maintainers to choose identifiers that are, in the immortal words of RFC 1341[1] "as unique as possible". One possible mechanism for generating such globally unique IDs is for the peer maintainer to use an Internet domain-name that they own, and append a locally unique token. Another is simply to generate a long string of random bits.

3.2.2 Addresses

In general, a single peer may be accessible on more than one address, and those addresses may be of different types. For example, a peer hosted on a machine with a dynamically allocated IP address might have as its preferred peer address a direct TCP/IP connection to a listening socket, but such a peer address will become stale when its host is allocated a new IP address. It might, therefore, also be addressable by email to an unchanging post-office domain. Connections made by means of the email address will furnish connecting peers with a peer-description record; after reading this, they might elect to switch to the new TCP/IP-connection peer address.

Each of the peer's addresses is indicated by an element of the form

```
<address type="xyz" bandwidth="n">addr</address>
```

in which the interpretation of `addr` is dependent on `type` and the bandwidth is expressed in kilobits per second.

The following address `type`s are recognised:

tcp A direct TCP/IP connection to an open port. The content of the `<addr>` element consists of two components separated by a colon (`:`) – first an Internet host name or IP address, and second a port number. Possible examples include "192.168.0.106:12368", "alvis.info:9999" and "alvis.pipedreaming.org:9876". There is no default port-number, so the second element may not be omitted: "192.168.0.106" would be illegal.

[1]**http://www.ietf.org/rfc/rfc1341.txt**

mail An email address to which suitably encoded Alvis network messages may be sent. For example, "alvis@miketaylor.org.uk", "ajasghd@alvis.info". The details of email transport are out of scope for this specification and will be documented elsewhere.

Additional address-types may be added in the future.

It is legal for a single peer to have addresses of multiple types, and to have multiple addresses of the same type.

3.2.3 Subject Areas

Once an Alvis network is up and running, we expect peers to discover each others' coverage of subject areas by adaptive learning from the results of sending queries and receiving responses. However, a peer may "prime the pumps" by indicating *a priori* that it has good support for queries in particular subject areas. This is done by providing one or more elements of the form

```
<subject scheme="ddc22">
  <code>567</code>
  <caption>Fossil cold-blooded vertebrates; fossil fishes</caption>
</subject>
```

Each such element indicates a subject area drawn from a standard scheme of subject headings, and carries both the subject code in that scheme and a textual caption suitable for showing to users.

The supported schemes are:

- `ddc22` – Dewey Decimal Classification
- `lcsh` – Library of Congress Subject Headings

More schemes may be added in the future as required.

3.2.4 Support for Elements of Alvis Functionality

This is the most important part of the peer-description records, carrying as it does information about the kinds of requests that can be made of the peer:

- The Alvis architecture supports multiple independent query languages. Therefore, each peer must specify which query-types it supports; for some query-types, additional information is required.
- In the same way, different ranking algorithms may be supported, and peers need to advertise which algorithms they support so others can request the use of supported algorithms in their requests. This facilitates the merging of results from disparate peers.
- Peers may return records using different metadata subsets: unique identifier only, Dublin Core summary, full text, etc.
- Finally, different peers are in general able to return records of different kinds – XML, PDF, MP3, etc.; and for some of these record formats, retrieval may be supported in multiple schemas.

We will now consider each of these dimensions separately.

Support for Query Types

Different Alvis peers are in general built on top of different existing databases using a variety of database toolkits that support various types of query. Accordingly, those peers will support different query types, which they need to describe using `<query>` elements within the `<support>` portion of the record:

```
<query type="abc">
  <!-- ... -->
</query>
```

Some types of query require further specification as to what subsets of all possible functionality they support. For example, in structured queries, individual query terms may be submitted to specific indexes such as "creator", "title" and "date". Provision is made in the description record for peers to indicate such details.

The precise set of query types to be supported in Alvis has yet to be finally established, and would in any case perhaps be kept open-ended. For now, we consider three types, to be represented by `<query>` elements with `type` attributes taking the values `cql`, `bag` and `xpath`, as discussed below.

Support for CQL Queries

CQL[2] is an abstract language for expressing structured queries in a rigorous and powerful yet human-readable syntax. It is used by some Z39.50[3] servers and in all SRW/SRU[4] implementations.

Support for CQL queries is indicated by a `<query type="cql">` element, containing zero or more `<searchFormat>` and `<index>` elements further specifying the degree of CQL support.

The `<searchFormat>` element is common to all query-types, and is described below.

Each `<index>` element contains the name of a CQL index-name that is supported, such as `creator`, `title` or `geographicName`. The optional `set` attribute may contain the URI of a particular CQL context set (analogous to an XML namespace) to which the specified index belongs. For example, `http://zing.z3950.org/cql/bath/2.0/` is the URI of the Bath Profile context set, version 2.0.

Support for "Bag-of-Words" Queries

Support for the standard information-retrieval "bag of words" query-type may be indicated by a `<query type="bag">` element. In this model, queries such as `sauropod cartilage stress biomechanics` are treated as short documents, to be associated with relevant target documents by similarity measures.

Zero or more `<searchFormat>` elements may be provided within the `<query type="bag">` element: see below.

[2]**http://zing.z3950.org/cql/**
[3]**http://lcweb.loc.gov/z3950/agency/**
[4]**http://lcweb.loc.gov/srw/**

Support for XPath Queries

XPath[5] is a language for selecting portions of XML records. In Alvis, an XPath query consists of an XPath specification together with a term; the query is satisfied by XML documents in which the portion specified by the XPath contains the indicated term.

Support for XPath queries is indicated by a `<query type="xpath">` element, containing zero or more `<searchFormat>` elements as described below. It is not clear that this type of query can be usefully used against records of formats other than XML.

A future version of the Alvis peer-description format may include means for peers to indicate which parts of the XPath specification they support.

Specifications Shared by Multiple Query Types

`<query>` elements of any `type` may contain zero or more `<searchFormat>` specifications. Each `<searchFormat>` element carries a `type` element, the value of which must be an IANA-registered MIME-type[6]. Examples include `text/plain`, `text/xml`, `audio/mpeg` (for MP3 files, see RFC 3003[7], etc. The element indicates that queries of the appropriate type can find documents of the specified type.

Support for Ranking Schemes

Each ranking scheme supported by a peer can be indicated by a `<rank>` element with a `type` element which may have the value `field` or `algorithm`.

Rankings of type `field` are implented by sorting target documents on the fields whose names are specified in the content of the `<rank>` element. Such field-names are separated by commas, and listed in descending order of precedence; the meaning of field-names is dependent on the record-format. For example, `<rank type="field">creator,date</rank>` indicates the ability to sort of the value of the creator field, with records having the same creator sorted by date.

Rankings of type `algorithm` are taken from a list to be enumerated elsewhere. Additional attributes, or possibly elements, may be added to a future revision of the peer-description record to allow the specification of parameters to algorithms.

Additional ranking types may be supported in subsequent revisions of the peer-description format. Means of combining rankings may be introduced. This area of the specification remains fluid due to the lack of implementation experience so far.

Support for Metadata Subsets

In search responses, target records may be returned from peers in various levels of completeness; that is, representing different subsets of the data and metadata. Search requests may indicate the level of completeness they require, and peer-description records can indicate the supported levels. This is done with

[5]**http://www.w3.org/TR/xpath**
[6]**http://www.iana.org/assignments/media-types/**
[7]**http://www.ietf.org/rfc/rfc3003.txt**

a `<subset type="name"/>` element, in which the `type` may take the following values:

id Only the identifier of the target record is returned. This is a short, opaque string that uniquely identifies the record within the peer providing the response. This identifier may subsequently be fed back to the same peer in a request for more of the record – typically, but not necessarily, the full text.

dc A record containing Dublin Core metadata (creator, title, subject, etc.) for the target record, typically including the record's identifier.

xpath An XML record containing those parts of the full target record satisfying a specified XPath expression. This only makes sense for XML target records.

fulltext The full text of the target record is returned.

Support for Record Formats

A given peer may support records in multiple formats (XML, plain text, email messages, PDF, etc.) Some of this information is communicated in the `<searchFormat>` elements of the query-support part of a peer-description record, but more detail may be exposed in a `<recordFormat>` section, including the schemas supported for each format and the ways in which those schemas can be expressed.

Each supported format is described by a `<recordFormat>` element:

```
<recordFormat type="mime/type">
  <!-- ... -->
</recordFormat>
```

This element has a mandatory `type` attribute, which must be an IANA-registered MIME-type[8].

A `<recordFormat>` may contain zero or more `<schema>` elements, each of which describes an abstract schema (i.e. not necessarily an XML Schema in the W3C sense) to which some target records conform:

```
<schema name="Dublin Core" tag="dc">
  <!-- ... -->
</schema>
```

The `name` attribute contains a human-readable name, and the `tag` attribute contains a short opaque string which other peers can use when requesting records to be returned in this schema.

`<schema>` may contain zero or more `<spec>` elements, each of which describes a concrete schema using some specific constraint syntax or a human-readable description: for example,

```
<spec type="xmlschema">http://foo.com/dc.xsd</spec>
```

The mandatory `<type>` element must take one of the values enumerated in the following list, and the content is a URI indicating the location of a schema specification of the appropriate type.

Possible `<spec>` types include:

[8] **http://www.iana.org/assignments/media-types/**

xmlschema The specification is a W3C Schema[9] formally describing XML documents.

dtd The specification is an XML DTD[10] formally describing XML documents.

relaxng The specification is a Relax NG[11] specification formally describing XML documents.

prose The specification is simply human-readable prose describing the schema (much as this document describes the peer-description schema). Unlike the first three specification types, this is suitable for describing non-XML schemas as well as XML.

3.2.5 Statistics

This area provides a place for peers to communicate various statistics about themselves, such as their mean and median search times, number of records held, degree of connectivity to other peers, etc.

Each such statistic is expressed as a `<statistic>` element with a `type` element that contains the name of the particular statistic whose value is the content of the element.

No specific interpretation is here placed on any particular statistic.

[9] **http://www.w3.org/XML/Schema**
[10] **http://www.w3.org/TR/REC-xml/#sec-prolog-dtd**
[11] **http://www.relaxng.org/**

Chapter 4

Enriched Documents

4.1 The Alvis Pipeline and The Family of Formats

Documents in the Alvis system go through several processes before entering the indexing engine, and this series of stages is known as the “pipeline”.

- First, documents are acquired from somewhere, and translated into a neutral format for the rest of the system to use. The process of document acquisiton may be through crawling the Web, exporting a local database, or any other means; and the type of document read in may be HTML, PDF, MS-Word, etc. – provided only that the output is in the prescribed acquisition format. An Alvis component that acquires documents and feeds them into the pipeline in acquisition format is called a Document Source. The sole Document Source in the prototype Alvis system is the semantically enhanced Web harvester developed in WP7 (Topic Specific Crawl).

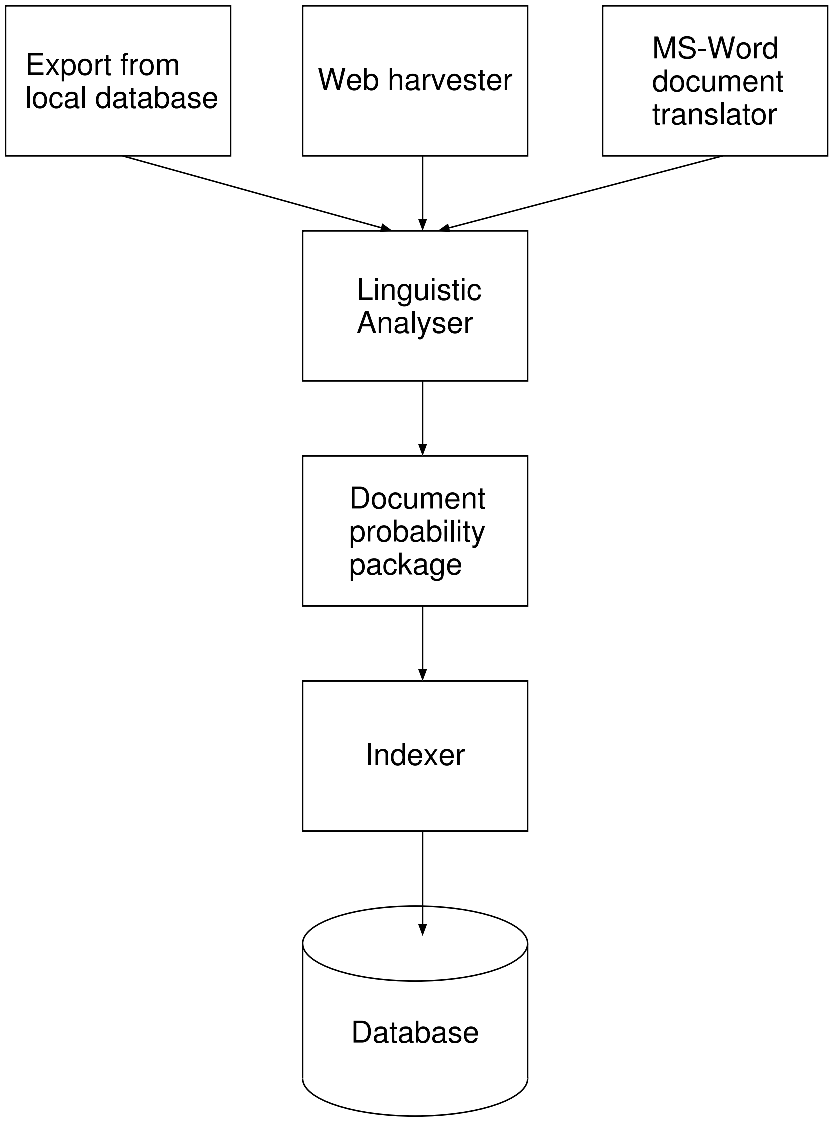
  The key part of acquisition format is the canonical representation of the acquired document. An important design principle for the canonical format is to avoid the ever-present temptation to think purely in terms of HTML. The goal we seek is *not* a semantically rigorous way to represent HTML, but a semantically rigorous canonical format for representing documents which have been converted from any one of a number of formats, including but not limited to HTML, OpenOffice.org and stacks of EBCDIC-encoded 80-column cards.

- Second, documents in the acquisition format are fed into the Lingustic Analyser for tokenisation, lemmatization, recognition as parts of speech and as domain-specific concepts, etc. The result of this process is a version of the document including the acquisition part but also augmented with a great deal of additional information in the form of stand-off annotation. This is called linguistic format.

- Third, documents in linguistic format are fed into the document probability package for relevance analysis. The result of this process is yet another version of the document, further augmented with the relevance information. This is called relevance format.

- Finally, relevance-enriched documents are passed to the indexing engine, which adds them to a database and creates the necessary indexing information to facilitate subsequent searching using semantic query criteria. (A digest of the indexing is also passed down into the network of infrastructure peers, so that they can update the distributed hash table that they maintain.)

The Alvis document-processing pipeline looks like this:


Export from local database
Web harvester
MS-Word document translator
Linguistic Analyser
Document probability package
Indexer
Database

The metadata format for enriched documents described in this document, then, is really a family of three closely related formats, each building on the last:

acquisition format, linguistic format and relevance format. These are each described in more detail in the following sections.

4.1.1 Acquisition Format

The first format – known as "acquisition format" – is a simple representation of a document broken into several semantically distinct parts (title, creator, body of text, etc.), including information about the acquisition process. It does not seek to represent all the details of the original form of the document. (It may, however, carry with it a copy of the original document, as described below.)

The goal of the canonical document carried in acquisition format is to capture all and only the semantically significant content of the original document, and to put it in a single well-defined form that subsequent components of the pipeline can easily handle without needing to know anything about the vagaries of, for example, HTML. The task of comprehending the various input formats, then, is encapsulated neatly by the various Document Sources, and the other Alvis components need know nothing of them.

4.1.2 Linguistic Format

The second format – known as "linguistic format" – is an augmented version of acquisition format. That is, a linguistic document contains all the same XML elements and content as the acquisition document from which it was derived, but has additional elements describing the linguistic information extracted from the canonical document itself. This means that every acquisition document is also a linguistic document, but with zero linguistic information.

Linguistic documents are created by the Linguistic Analyser built in WP5 (Document Analysis and Normalization), and the linguistic document format is designed to capture the kind of information that WP5 generates.

Such information could be added to the document in one of two ways: either by embedding it within the core document, inserting additonal XML tags to elucidate the structure; or by placing the additional information alongside the core document, with pointers indicating what parts of the core document each annotation pertains to. This second approach is known as "stand-off annotation", and this is the approach taken by the linguistic format.

4.1.3 Relevance Format

The third format – "relevance format" – is an augmented version of linguistic format. That is, a relevance document contains all the same XML elements and content as the linguistic document from which it was derived, but has additional elements describing the relevance information extracted from the linguistic document. This means that every relevance document is also a linguistic document,but with zero relevance information (and every acquisition document is a relevance document, but with zero linguistic or relevance information).

Relevance documents are created by the relevance package built in WP2 (Document Probability Model), and the relevance document format is designed to capture the kind of information that WP2 generates.

Document Sources such as the harvester should create enriched documents consisting only of of a acquisition section. The Linguistic Analyser should not alter the acquisition section of documents that pass through it in any way, only adding a linguistic section (or altering an existing linguistic section). The relevance package should not alter the acquisition or linguistic sections of documents that pass through it in any way, only adding a relevance section (or altering an existing relevance section).

Implementation Experience and Status

We have some implementation experience with this family of formats, which indicates that the design is sufficiently expressive to serve the needs of an operational Alvis peer. In an integration workshop in late 2005, an Alvis pipeline was built that used WP7's topic-specific web crawler to acquire HTML documents, translate them into canonical form, and wrap them into records of Aquisition Format. These documents were passed down the pipeline to WP2's document probability engine, which processed the incoming documents into Relevance Format. These augmented documents were then passed to WP3's indexing engine, Zebra, which stored and indexed them. Finally, the documents were searched for, and retrieved, using a simple command-line client.

This exercise demonstrated that the formats suffice to represent all the information generated by at least one Document Source, that the process of transforming a document as it passes through the pipeline is sound, and that the resulting enriched documents can be usefully indexed.

At present, it remains to integrate WP5's linguistic analyser into the pipeline: as a result, the integration workshop's pipeline did not provide the document probability engine with all the information it would normally use to calculate its scores, and the indexed documents lacked linguistic analysis. Until the WP5 software is integrated, the sufficiency of Linguistic Format will remain untested.

4.2 Sample Canonical Document

The following example canonical document serves as a motivating example for the discussion to follow. (Note that this is not an acquisition document, but only one element of one: a complete acquisition document includes not only a canonical document such as this, but also extracted metadata, timestamps and other information described below.)

```
<?xml version="1.0"?>
<!-- $Id: canonical-document.xml,v 1.2 2006/01/12 11:15:34 mike Exp $ -->
<canonicalDocument xmlns="http://alvis.info/enriched/"
                   version="1.1">
  <section title="Why Dinosaurs are Cool">
    Dinosaurs are much cooler than mammals because they are so much
    bigger.  T. rex could eat a tiger, easy.
    <section title="Saurischians">
      Saurischians include the sauropods (the biggest of all
      dinosaurs) and the theropods (the carnivorous dinosaurs).
    </section>
    <section title="Ornithischians">
      Ornithischians include Triceratops, Pachycephalosaurus,
      Stegosaurus, Ankylosaurus and Iguanodon.
    </section>
  </section>
  <section><!-- no title could be found for this section -->
    The coolest
    <ulink url="http://www.dinodata.net/">dinosaurs</ulink>
    of all were the sauropods.  They were way
    huge.  I mean, you may think it’s a long way from the top of a
    giraffe to the bottom, but that’s peanuts to a Brachiosaurus.
    The coolest sauropods of all were:
    <list>
      <item>Bruhathkayosaurus</item>
      <item>Amphicoelias fragillimus</item>
      <item>Brachiosaurus, which has the species:
        <list>
          <item>altithorax</item>
          <item>brancai</item>
          <item>?nougaredi</item>
        </list>
      </item>
      <item>
        <ulink url="http://www.snomnh.ou.edu/pdf/2000/00-27.pdf">
          Sauroposeidon
        </ulink>
      </item>
      <item>Migeod’s mysterious M23 sauropod</item>
    </list>
  </section>
</canonicalDocument>
```

4.3 Discussion of the Canonical Format

In this chapter, we are concerned only with the canonical record itself: that is, the canonical representation into which source documents are transformed. We do not discuss here the remaining elements of the enriched document's `<acquisition/>` section: these are covered below.

4.3.1 Intent of the Canonical Format

The intention of the `<canonicalDocument>` element is that it contains markup transformed from the original document where and only where that markup indicates document structure with some semantic significance that can usefully be indexed and subsequently searched. For example, if a Web harvester is preparing an HTML document that includes the following fragment:

```
Here is what <font face="Ariel">Holtz</font> says:
<blockquote>
  All those great coelurosaur fossils from Liaoning are a
  couple of lucky rolls of the taphonomic crap shoot.
</blockquote>
```

then the `<font>` tag, which serves only a cosmetic purpose in the HTML and conveys no information at all, must be discarded, and may not be represented in the `<canonicalDocument>` but the `<blockquote>` tag is conveying real information, so we define our format such that it doesn't preclude the possibility of preserving such information in the text that gets passed into the semantic workpackages.

(If experiments indicate that it would be useful, a future version of this format may preserve questionable markup such as italics, which can have semantic significance in some fields – e.g. indicating the use of a formal scientific genus or species name in biology).

4.3.2 Elements Included in the Canonical Format

In accordance with the design principles expounded above, we define canonical format to be capable of representing the following structural elements as well as plain text:

Sections with Titles A canonical document consists of a series of `<section>` elements, which may represent any logical division of a source document: chapters of a book, pages of an article, HTML `<div>` elements, etc.

Each `<section>` element may have a `title` attribute. This may be extracted from any suitable part of the source document: the `<h1>`...`<h6>` elements in HTML, lines with a large point-size in MS-Word documents, etc.

Each `<section>` contains a mixture of plain text, `<list>`s and `<ulink>` elements:

```
<section>
This is a simple section with
no contained lists or links.
```

```
</section>
```

Lists with Items A list is represented by a `list` element contains a sequence of `<item>` elements. No distinction is made between bullet lists, numbered lists, etc.

Each `<item>` contains a mixture of plain text, `<ulink>`s and sublists, represented by `<list>` elements.

```
<list>
<item>First entry</item>
<item>Second entry</item>
<item>Third entry</item>
</list>
```

Links with URLs Links to other documents are represented by `<ulink>` elements. (The `u` in the name is to avoid a clash with the unrelated `<link>` element in the `<inlinks>` and `<outlinks>` parts of the `<links>` section. The name is chosen because it's what DocBook XML uses for its analogous element.)

The URL of the linked document is specified by the `<ulink>` element's `url` attribute. The anchor text of the link is the content of the element, and may not contains further sections, lists or links:

```
<ulink url="http://google.com/">
The Google search engine
</ulink>
```

4.4 Sample Enriched Document

The following example enriched document (actually a collection consisting of a single document) serves as a motivating example for the discussion to follow.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: enriched-document.xml,v 1.3 2006/01/12 11:29:30 mike Exp $ -->
<documentCollection xmlns="http://alvis.info/enriched/"
                    version="1.1">
  <documentRecord id="12345678">

    <!-- Information generated during the initial acquision of the
    document, whether by a web crawler, MS-Word converter, etc. -->
    <acquisition>

      <!-- Information, in the current WP7 format, to do with the
      acquisition process: acquisition date, URLs where the document was
      found, expiry date, size, etc. -->
      <acquisitionData>
        <modifiedDate>2001-04-19</modifiedDate>
        <expiryDate>2004-11-06</expiryDate>
        <checkedDate>2004-10-06</checkedDate>
        <httpServer>WebSTAR/4.4(SSL) ID/72915</httpServer>
        <urls>
          <url>http://www.snomnh.ou.edu/pdf/2000/00-27.pdf</url>
        </urls>
      </acquisitionData>

      <!-- Original document represented as cleaned HTML, or text
      extracted from MSWord, PS, PDF, etc.  May be a binary format, with
      an attribute specifying base64 or quoted-printable encoding -->
      <originalDocument mimeType="text/plain" charSet="us-ascii">
        ...
      </originalDocument>

      <!-- Visible text from document, together with what internal
      structure we can express through canonical markup -->
      <canonicalDocument>
        <!-- As in previous example -->
      </canonicalDocument>

      <!-- Information, in the current WP7 format, that is _about_ the
      document rather than part of it: e.g., creator, title, subject, DOI -->
      <metaData>
        <meta name="dc.creator">Wedel, Mathew J.</meta>
        <meta name="dc.date">2000</meta>
        <meta name="dc.title">Sauroposeidon proteles, a new sauropod from
                the Early Cretaceous of Oklahoma</meta>
      </metaData>

      <!-- Link information from WP7 format. All URLs will contain an
      internal ID (not guaranteed to be unique across multiple crawler
      instances) -->
      <links>
        <outlinks> <!-- links to external pages -->
          <link type="a"> <!-- repeatable -->
            <anchorText>Text from this document</anchorText>
            <location documentId="...">URL</location>
          </link>
        </outlinks>
        <inlinks> <!-- links from external pages -->
          <link type="a"> <!-- repeatable -->
            <anchorText>PDF ( 1 MB)</anchorText>
            <location documentId="...">http://www.snomnh.ou.edu/publications/Articles/inde
          </link>
        </inlinks>
        <!-- Number of unique other hosts with links pointing to this page -->
        <inlinkHosts> ... </inlinkHosts>
      </links>

      <!-- Results of analysis done as part of the acquisition process,
      e.g. genre intuited from top-level domain name of the site from
      which a Web document was crawled -->
      <analysis>
        <!-- analysis also containes other analysed properties (mainly from
             the URL) with property name as tag and content as value -->
        <property name="topLevelDomain">edu</property>
        <property name="language">en</property>
        <property name="genre">article</property>
        <ranking scheme="..."> ... </ranking> <!-- repeatable -->
        <topic absoluteScore="150" relativeScore="570">
          <class>ALL</class>
        </topic>
        <topic absoluteScore="100" relativeScore="380">
          <class>CP</class>
          <terms>carnivorous plant[^\s]*, carnivor[^\s]*, </terms>
        </topic>
        <topic absoluteScore="50" relativeScore="190">
          <class>CP.Dionaea</class>
          <terms>flytrap[^\s]*, venus flytrap[^\s]*, </terms>
        </topic>
      </analysis>
    </acquisition>

    <!-- Annotations from WP5 -->
    <linguisticAnalysis>
      <!-- Details omitted: see Deliverable D5.1 -->
    </linguisticAnalysis >

    <!-- Relevance information added from WP2 -->
    <relevance>
      <scoreset type="ranking">
        <score topicId="1">8.36536</score>
        <score topicId="4">4.25395</score>
        <score topicId="19">0.44538</score>
```

```
        <score topicId="36">2.35349</score>
      </scoreset>
      <scoreset type="content">
        <score topicId="1">40.25395</score>
        <score topicId="4">2.947</score>
        <score topicId="17">0.44538</score>
        <score topicId="23">1.4629</score>
        <score topicId="36">2.35349</score>
      </scoreset>
    </relevance>
  </documentRecord>
</documentCollection>
```

4.5 Discussion of the Enriched Document

4.5.1 Representing Multiple Documents in a Single Package

A `<documentCollection>` is merely a wrapper for several `<documentRecord>` objects, and has no information of its own: it's not a collection about some specific subject, or a collection of documents harvested from a particular place, or anything similar: it's just a simple, unstructured aggregate like a TAR archive or or ZIP file.

This top-level element has two mandatory attributes, `xmlns` and `version`, with the same meaning as the corresponding attributes of the `<peer>` element at the top level of peer-description records. For enriched document collections, the namespace is `http://alvis.info/enriched/`.

4.5.2 Document Identifiers and Identity

Each document in a complete Alvis network is identified by an opaque, unique identifier, represented by the `id` attribute on the `<documentRecord>` element. This identifier must remain constant as the record takes on its various forms (acquisition, linguistic, relevance) during its journey through the Alvis pipeline. This identifier may be subsequently used to specify records for deletion or update.

Because Document Sources are the first components to handle the documents that are passed through the Alvis pipeline, it falls to them to allocate the identifiers. However, the identifier identifies the entire enriched document, in each of the forms that it takes, not just the `<acquisition>` section that is a Document Source's main responsibility. Accordingly, the `id` attribute is on the `<documentRecord>` element rather than `<acquisition>`.

Document Sources must choose identifiers to be "as unique as possible", as must peer identifiers. As with peers, globally-unique ID generation mechanisms include using an Internet domain-name together with a locally unique token and generating long strings of random bits. A third candidate approach is to use an MD5 checksum of the document.

Using the MD5 checksum (or any checksum) has the property that two identical documents acquired at different times and by different Sources will have the same identifier, and will thus be, for Alvis purposes, "the same record". Is this property desirable or broken? That is a matter of application-level policy rather than of protocol-level mechanism: the Document Source must allocate identifiers depending on what its notion of a record's identity is.

Some consequences follow if MD5 checksums are used as document identifiers: for identity purposes an document is *a specific sequence of bytes*. This means that, given one document to compare with:

- If a document consisting of the same sequence of bytes is found at another URL (or is acquired by another process that uses the same ID-generation algorithm) then that is deemed to be The Same Document.

- If the document is re-fetched from the same URL, and it includes a counter or a time indication, then it's A Different Document the second time (and each subsequent time).

- If another document is identical except that a typo has been fixed, or a tab character replaced by eight spaces, then it's A Different Document.
- If another document consists of exactly the same characters in the same order, but encoded in a different character set (e.g. UTF-8 vs. ISO-Latin-1) then it's A Different Document. (This is because MD5 considers documents as sequences of bytes rather than of characters.)

In contrast, if the harvesting URL of a document were used as its identifier, then:

- Another document, byte-for-byte equal but from a different location, would be considered A Different Document.
- When the document is modified (e.g. a spelling mistake corrected) and subsequent re-harvested, it is correctly recognised as a new version of The Same Document, superseding the original.
- When the document is completely changed (e.g. the "most recent post" page of a blog replaced by a newer post) and subsequent re-harvested, it is incorrectly considered as a new version of The Same Document.

It can be seen that using MD5 checksums or URLs as identifiers yield different "correct" and "incorrect" cases, judged according to what we intuitively consider to be variants of the same document. Clever document sources may manage to contrive an identifier generation scheme that combines the strong points of these approaches.

4.5.3 The acquisition Section

Acquisition Data

The `<acquisitionData>` subsection describes the process by which the document was acquired, rather than the acquired document itself. It consists of the following elements, in the specified order.

`<modifiedDate>` **(mandatory)** The date and time at which the document was last acquired, the most recent version superseding any previous versions. Like all other datestamps in the enriched record format, it must be provided in accordance with the ISO 8601 specification, which allows (among others) the following formats:

- `1998` – year only.
- `1998-03` – year and month.
- `1998-03-18` – year, month and day.
- `1998-03-18 03` – year, month, day and hours.
- `1998-03-18 03:28` – year, month, day, hours and minutes
- `1998-03-18 03:28:12` – year, month, day, hours, minutes and seconds.

`<expiryDate>` **(optional)** The date at which the record's currency expires, so that the Document Source must revisit the document at its original location to verify that it has not changed.

`<checkedDate>` **(optional)** The date at which the document was last checked at its original location, to determine that it had not changed since being acquired.

`<httpServer>` **(optional)** For Document Sources that work by crawling the Web this provides a way to indicate the server software, e.g. `Apache/2.0.40 (Red Hat Linux)`

`<urls>` **(optional)** For Document Sources that work by crawling the Web, each of the potentially many URLs where the document was found is recorded inside a `<url>` element, of which there may be any number within `<urls>`.

Original Document

The `<originalDocument>` element contains a copy of the original document, perhaps compressed and encoded. This original document is *not* used in subsequent analysis, but only for delivery to users. Accordingly, it may be in any format, including binary formats such as PDF (so long as it is suitably encoded).

In general, this element should contain a byte-for-byte copy of the document as it was originally acquired. However, for some formats, it may be beneficial to use a transformed version of the document: for example, the non-conformant HTML found on many web pages can profitably be cleaned up using a tool such as HTML Tidy[1].

The `<originalDocument>` element has the following attributes:

mimeType (mandatory) The type of the original document, chosen from the controlled list maintained by IANA[2]. Example values include `text/plain`, `text/xml`, `text/html`, `application/pdf` and `application/msword`.

charSet (mandatory) The character set used by the original document, chosen from the controlled list maintained by IANA[3]. Example values include `UTF-8`, `ISO-8859-1`, and `US-ASCII`.

Note well that this attribute indicates the character set used by included (possibly compressed and/or encoded) document – not that of the document that it's included in, which is specified by the XML declaration.

compression (optional) Indicates that the document has been compressed from its original form to yield the sequence of bytes that form the content of the `<originalDocument>` element. This attribute may take the following values:

deflate The document is compressed using the DEFLATE Compressed Data Format Specification version 1.3, as specified in RFC 1951[4] (May 1996).

gzip The document is compressed using the GZIP file format specification version 4.3, as specified in RFC 1952[5] (May 1996).

[1]**http://www.w3.org/People/Raggett/tidy/**
[2]**http://www.iana.org/assignments/media-types/index.html**
[3]**http://www.iana.org/assignments/character-sets**
[4]**ftp://ftp.isi.edu/in-notes/rfc1951.txt**
[5]**ftp://ftp.isi.edu/in-notes/rfc1952.txt**

encoding (optional) Indicates that the (perhaps compressed) document has been encoded to protect its content from being interpreted as XML markup. This is necessary if the content contains either of the characters `<` or `&`, or if it contains characters such as ASCII 007 (BEL) that cannot be represented in XML. This attribute may take the following values:

- **quoted-printable** The document is encoded using the Quoted-Printable transfer endoding, as specified in section 6.7 of RFC 2045[6] (November 1996).

  Note that use of the Quoted-Printable encoding does not in itself guarantee that the encoded document is safe to embed as text in an XML document. The particular Quoted-Printable implementation needs to ensure that, among any other translations that it does, it translates `<` to `=3c` and `&` to `=26`.

- **base64** The document is encoded using the Base64 transfer endoding, as specified in section 6.8 of RFC 2045[7] (November 1996).

- **xml** The document is encoded using the XML escapes `&amp;` for `&`, `&lt;` for `<`, and `&gt;` for `>`.

  When this encoding is used, some provision must also be made for characters such as ESCAPE (ASCII 0x1b) which simply cannot be represented in XML – not even as numeric entities such as `&#x1b;`. See the XML 1.0 specification, section 2.2 (*Character*)[8] for details of which characters are acceptable; do not, however, look there for an explanation of this moronic and arbitratry restriction.

  Since XML does not allow characters such as ESCAPE to be represented, there may be no realistic alternative other than the discard them, which for some formats may be disastrous. For this reason, we recommend that one of the other encodings be used in place of `xml`.

Example:

```
  <originalDocument mimeType="application/msword" charSet="utf-8"
    compression="gzip" encoding="base64">
H4sICDwAXEECA3RleHQuZG9jAO19C5xcRZlvdRKS4ZEQAoSAEdoYYRI6Qx6TyQOuy2QmIQkJGTLh
/ZAzMz2ZJjPdQ3fPhGFZFjG+EBBZBJZFRS4quMCPdV1EdL1cFlG5XNd1uVzkclcX0SuK3BhZVlwg
+6+vvjpVdU6d09OTUMIv3flP+nSfU6fqq6+++l5V5wf/eMhPPv83R/2riLxOFBPFG7v3F5Ot7zLA
...
</originalDocument>
```

Canonical Document

The section contains the canonical document itself, as described above.

Metadata

This section contains information *about* the document, as opposed to the content *of* the document. Document Sources are at liberty to acquire this information in any way they can. Examples include:

[6] **ftp://ftp.isi.edu/in-notes/rfc2045.txt**
[7] **ftp://ftp.isi.edu/in-notes/rfc2045.txt**
[8] **http://www.w3.org/TR/2004/REC-xml-20040204/#charsets**

- If the document is acquired from a database, the a metadata record associated with the document may be harvested from the database along with it.
- If the document is acquired from a simple filesystem, some metadata may be harvested from the attributes of the file, e.g. date of last modification.
- If the document is acquired from a complex filesystem, additional metadata may be available. For example, in Apple filesystems, there is a "resource fork" corresponding to the "data fork" that contains the actual document.
- Documents in some formats may carry their own metadata with them: for example, HTML documents harvested from the Web usually provide a `<title>` element, and additional metadata is often available in `<meta>` tags indictating the creator, keywords, description, etc.

Whatever the source of the metadata, Document Sources should express it in simple context=value pairs using `<meta>` tags within the `<metaData>` section, like this:

```
<meta name="dc.creator">Wedel, Mathew J.</meta>
<meta name="dc.date">2000</meta>
```

The valid values of the `name` attribute are the names of the fifteen Dublin Core Simple elements[9].

Links

The acquisition record may contain a `<links>` section which provides an indication of both the inbound and outbound links for this core document. The former are held within an `<inlinks>` container, the latter in an `<outlinks>` container.

Outbound links are easy to discover by static analysis of the document. By contrast, a comprehensive list of inbound links can by obtained only by analysing an entire corpus; and such a list can only be exhaustive with respect to a specific corpus, since it is always possible that there is another document somewhere in the world that links to it.

Both inbound and outbound links are represented by the same structure: a `<link>` element with a `type` attribute, containing `<anchorText>` and `<location>` subelements. The `type` attribute takes a value indicating the kind of link. The possible values for this attribute are taken from the corresponding HTML tags as follows:

a A conventional hypertext anchor.

img An embedded image that is part of a page.

frame An embedded frame that is part of a frameset.

text A URL found in plain text.

[9] **http://dublincore.org/documents/dces/**

Analysis

The acquisition record may contain an `<analysis>` section which contains the results of simple pre-processing done on the data by the harvester. This information may serve as a guide for the more sophisticated analysers later in the Alvis pipeline.

The analysis section may contain the follow subelements, all of them optional and repeatable:

`<property>` Specifies the value of any named property of the document: for example, if the language of the document is known to be English, this can be specified using `<property name="language">en</property>`

`<ranking>` Specifies the ranking of the document under a specific scheme named by the `scheme` attribute. For example, `<ranking scheme="abc">42</ranking>`

`<topic>` Indicates whether or not the document belongs a specific topic.

The `<class>` subelement is a topic or sub-topic specifier indicating the topic for which the document is here classified. It often comes from a hierarchical classification system such as Engineering Index.

The two attributes, `absoluteScore` and `relativeScore`, indicate the document's score in the specified topic, as assigned by the Document Source, the latter being normalized by the size of the document. The scores are calculated as described in the milestone document MS7.1 and are based on the terms indicated by regular expressions in the `<term>` subelement.

4.5.4 The linguisticAnalysis Section

The `<linguisticAnalysis>` section describes the results of WP5's linguistic analysis, and follows the form described in Deliverable D5.1, *Report on method and language for the production of the augmented document representations.* That document both describes the format of this element and includes a DTD formally specifying it.

4.5.5 The relevance Section

The `<relevance>` section describes the results of WP2's document probability calculations as a sequence of `<scoreset>` subsections, each of which provides scores for a set of topics. Each `<scoreset>` has a `type` attribute specifying its applicability. Types likely to appear in relevance score-sets include:

content A topic score based on the relevance of the document content.

ranking A topic-sensitive authority score that merges the topical relevance of the document content as well as the topical relevance of documents that link to it. "Authority" in this sense implies that other documents on the topic link to it, and thus it is considered important for the topic.

Each `<scoreset>` element contains zero or more `<score>` elements, which in turn carry a `topicId` attribute and contain a a floating-point number measuring the score of the identified topic according to the document probability model. The form of the `topicId` attribute remains controversial: either an opaque numeric or symbolic identifier may be used, or a short phrase identifying the topic in a human-readable way.

In principle, this section of a relevance-format enriched document carries information analogous to that generated at harvesting time and encoded in `<topic>` elements within the `<acquisition>` section's `<analysis>` subsection. The Document Source and Document Probability software components use different approaches to trying to determine the topic of a document. For example, the subject-specific web crawler developed in WP7 is based on an an explicit topic definition (ontology), while the WP2 software's relevance figures are based on statistical modeling of documents in a collection.

Chapter 5

XML Schemas

5.1 Schema for Peer-Description Records

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: peer-description.xsd,v 1.6 2006/01/23 09:09:54 mike Exp $ -->
<!-- This Schema prescribes the format of Alvis peer-description records -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
           targetNamespace="http://alvis.info/peer/"
           xmlns:peer="http://alvis.info/peer/">

  <xs:element name="peer">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" ref="peer:addresses"/>
        <xs:element minOccurs="0" ref="peer:subjectAreas"/>
        <xs:element minOccurs="0" ref="peer:support"/>
        <xs:element minOccurs="0" ref="peer:statistics"/>
      </xs:sequence>
      <xs:attribute name="version" use="required"/>
      <xs:attribute name="name" use="required"/> <!-- human-readable -->
      <xs:attribute name="id" use="required"/> <!-- "as unique as possible" -->
    </xs:complexType>
  </xs:element>


  <!-- A peer in general has multiple addresses -->
  <xs:element name="addresses">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:address"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="address">
    <xs:complexType mixed="true">
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NCName">
            <xs:enumeration value="tcp"/>
            <xs:enumeration value="mail"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="bandwidth"/> <!-- in kbps -->
    </xs:complexType>
  </xs:element>


  <xs:element name="subjectAreas">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:subject"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="subject">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="peer:code"/>
        <xs:element ref="peer:caption"/>
      </xs:sequence>
      <xs:attribute name="scheme" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NCName">
            <xs:enumeration value="ddc22"/>
            <xs:enumeration value="lcsh"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="code" type="xs:string"/>
  <xs:element name="caption" type="xs:string"/>


  <xs:element name="support">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:query"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:rank"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:subset"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:recordFormat"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>


  <xs:element name="query">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:searchFormat"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:index"/>
      </xs:sequence>
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NCName">
            <xs:enumeration value="cql"/>
            <xs:enumeration value="bag"/>
            <xs:enumeration value="xpath"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="searchFormat">
    <xs:complexType>
      <!-- types are MIME-types such as "text/xml" -->
      <xs:attribute name="type" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="index">
    <xs:complexType mixed="true">
      <xs:attribute name="set"/> <!-- context-set URI for CQL queries -->
    </xs:complexType>
  </xs:element>

  <xs:element name="rank">
    <xs:complexType mixed="true">
      <!-- The content of rank depends on its type -->
      <xs:attribute name="type" use="required"/>
    </xs:complexType>
  </xs:element>


  <xs:element name="subset">
    <xs:complexType>
      <!-- e.g. "id" for record identifiers, DC for Dublin Core summaries -->
      <xs:attribute name="type" use="required"/>
    </xs:complexType>
  </xs:element>


  <xs:element name="recordFormat">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:schema"/>
      </xs:sequence>
      <!-- MIME-types such as "text/xml" -->
      <xs:attribute name="type" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="schema">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:spec"/>
      </xs:sequence>
```

```
      <xs:attribute name="name" use="required"/> <!-- human-readable -->
      <xs:attribute name="tag" use="required"/> <!-- known to applications -->
    </xs:complexType>
  </xs:element>

  <!-- Each schema may be specified in multiple ways, e.g. as a DTD -->
  <xs:element name="spec">
    <xs:complexType mixed="true">
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NCName">
            <xs:enumeration value="xmlschema"/>
            <xs:enumeration value="dtd"/>
            <xs:enumeration value="relaxng"/>
            <xs:enumeration value="prose"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>


  <xs:element name="statistics">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="peer:statistic"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="statistic">
    <xs:complexType mixed="true">
      <xs:attribute name="type" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

5.2 Schema for Enriched Documents

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: enriched-document.xsd,v 1.5 2006/01/20 17:40:23 mike Exp $ -->
<!-- This Schema prescribes the format of Alvis enriched document records -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
           targetNamespace="http://alvis.info/enriched/"
           xmlns:enriched="http://alvis.info/enriched/">

  <xs:element name="documentCollection">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:documentRecord"/>
      </xs:sequence>
      <xs:attribute name="version" use="required"/>
    </xs:complexType>
  </xs:element>


  <xs:element name="documentRecord">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="enriched:acquisition"/>
        <xs:element minOccurs="0" ref="enriched:linguisticAnalysis"/>
        <xs:element minOccurs="0" ref="enriched:relevance"/>
      </xs:sequence>
      <xs:attribute name="id" use="required"/>
    </xs:complexType>
  </xs:element>


  <xs:element name="acquisition">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="enriched:acquisitionData"/>
        <xs:element minOccurs="0" ref="enriched:originalDocument"/>
        <xs:element ref="enriched:canonicalDocument"/>
        <xs:element minOccurs="0" ref="enriched:metaData"/>
        <xs:element minOccurs="0" ref="enriched:links"/>
        <xs:element minOccurs="0" ref="enriched:analysis"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="acquisitionData">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="enriched:modifiedDate"/>
        <xs:element minOccurs="0" ref="enriched:expiryDate"/>
        <xs:element minOccurs="0" ref="enriched:checkedDate"/>
        <xs:element minOccurs="0" ref="enriched:httpServer"/>
        <xs:element ref="enriched:urls"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="modifiedDate" type="xs:string"/>
  <xs:element name="expiryDate" type="xs:string"/>
  <xs:element name="checkedDate" type="xs:string"/>
  <xs:element name="httpServer" type="xs:string"/>
  <xs:element name="urls">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:url"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="url" type="xs:string"/>

  <xs:element name="originalDocument">
    <xs:complexType mixed="true">
      <xs:attribute name="mimeType" use="required"/> <!-- from IANA's list -->
      <xs:attribute name="charSet" use="required"/> <!-- from IANA's list -->
      <xs:attribute name="compression">
        <xs:simpleType>
          <xs:restriction base="xs:NCName">
            <xs:enumeration value="deflate"/>
            <xs:enumeration value="gzip"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="encoding">
        <xs:simpleType>
          <xs:restriction base="xs:NCName">
            <xs:enumeration value="quoted-printable"/>
            <xs:enumeration value="base64"/>
            <xs:enumeration value="xml"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="canonicalDocument">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:section"/>
      </xs:sequence>
      <xs:attribute name="version"/> <!-- only when this is document element -->
    </xs:complexType>
  </xs:element>
  <xs:element name="section">
```

```
    <xs:complexType mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="enriched:list"/>
        <xs:element ref="enriched:ulink"/>
        <xs:element ref="enriched:section"/>
      </xs:choice>
      <xs:attribute name="title"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="list">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:item"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="item">
    <xs:complexType mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="enriched:list"/>
        <xs:element ref="enriched:ulink"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="ulink">
    <xs:complexType mixed="true">
      <xs:attribute name="url"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="metaData">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:meta"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="meta">
    <xs:complexType mixed="true">
      <xs:attribute name="name" use="required"/> <!-- Dublin Core element -->
    </xs:complexType>
  </xs:element>

  <xs:element name="links">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" ref="enriched:outlinks"/>
        <xs:element minOccurs="0" ref="enriched:inlinks"/>
        <xs:element minOccurs="0" ref="enriched:inlinkHosts"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="outlinks">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:link"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="inlinks">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:link"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="inlinkHosts" type="xs:string"/>
  <xs:element name="link">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" ref="enriched:anchorText"/>
        <xs:element ref="enriched:location"/>
      </xs:sequence>
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NCName">
            <xs:enumeration value="a"/>
            <xs:enumeration value="img"/>
            <xs:enumeration value="frame"/>
            <xs:enumeration value="text"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="anchorText" type="xs:string"/>
  <xs:element name="location">
    <xs:complexType mixed="true">
      <xs:attribute name="documentId"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="analysis">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:property"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:ranking"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:topic"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="property">
    <xs:complexType mixed="true">
      <xs:attribute name="name" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="ranking">
    <xs:complexType mixed="true">
      <xs:attribute name="scheme" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="topic">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="enriched:class"/>
        <xs:element minOccurs="0" ref="enriched:terms"/>
      </xs:sequence>
      <xs:attribute name="absoluteScore" use="required"/>
      <xs:attribute name="relativeScore" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="class" type="xs:string"/>
  <xs:element name="terms" type="xs:string"/>


  <xs:element name="linguisticAnalysis" type="xs:string"/>
  <!-- Details omitted: see Deliverable D5.1 -->


  <xs:element name="relevance">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:scoreset"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="scoreset">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="enriched:score"/>
      </xs:sequence>
      <xs:attribute name="type" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="score">
    <xs:complexType mixed="true">
      <xs:attribute name="topicId" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```