Requirements of „Content based web access“ software project

Requirements analysis

1. Architecture, design and selection of the used software
    - requirement analysis (final version)
    - functional specification (functions of the system, etc.)
    - technical specification (software packages, function test of used software, performance test, etc.)
    - architecture and design of the whole system
    - project plan

2. Maintenance of document bases
    - Number of XML-documents: < 1000 (when saved as single pages then 400.000 small documents)
    - Size of XML-documents: 3KB – 20 MB
    - XML-documents are delivered as XML-documents (with XSD-Schema, RELAX NG, or DTD for TEI) and later with presentation transformation (XSL,CSS or ...). This is done by the „Data Entry Spec Group“.
    rule for XML-documents: graphic element contains relative URLs
    - Web based CRUD-Operations (create, read, update, delete) for XML-documents (xml, dtd, xsl) in different document bases in different languages (english, german, chinese, old latin, ...) with duplicate check and versioning. Each of this CRUD-operation triggers also the same operation for the document base of the used search engine and database system.
    - Web based CRUD-Operations (create, read, update, delete) for inline images of the XML-documents: in file system in a special image directory (level under the XML-document)
    - all documents are accessible by the web server (of Tomcat)
    - Encoding: Unicode (only UTF8), other encodings (ascii, ISO 8859-X, UTF16, etc.) have to be converted to UTF8
    - Metatags of XML-documents: Author, Title, ..., Language, ...

3. Web based access to XML-documents
    - fulltext querying XML-documents: words combined with „logical or“ and „logical and“ and „logical andNot“, wildcard-querying (*, range of characters, ...), fuzzy querying, stemming (language specific), near-querying
    - querying XML-documents by search attributes (author, publication year, language, ...); date range querying
    - structural querying of XML-documents:
        - collection('/db/test')//chapter[contains(., 'XML')]
    - structural querying in one XML-document:
        - doc("/db/shakespeare/plays/hamlet.xml")//chapter[contains(., 'hamlet')]
    - query results: sorted by relevance, sorted by fields (author, ...), links directly to the XML-document and opens it with the desired DTD and XSL (or CSS) and with inline images; a one page view and a complete view of the XML-document is possible
    - XML-document is enriched by links of content words to external ressources (e.g. Pollux dictionary or Wikipedia(?) or wortschatz.uni-leipzig.de) and also by inline images (TEI graphics element)
    - Web-GUI for querying (with translation to the system query language) and

for presentation of query results
- utilities: bash-scripts for special tasks (file converting, ...), Java-XML-Utilities,

4. Establishment as a central service
   - maintenance of the central server (with Tomcat, Lucene or eXist, document bases, backup, ...)
   - deployment of WAR-application files to Tomcat
   - project software is maintained by version control
   - project software will be documented
   - transfer of the test working environment to the productional working environment (in agreement with the library (Urs Schoepflin) and ECHO (Simone Rieger))
   - all software is downloadable as open software

Needed software

For fulfilling the requirements the following software should be used:

1. Maintenance of document bases
ESciDoc could be used as the central document base system (intermediate service could be used: to be examined). All CRUD-Operations for XML-documents (and referenced images) have to trigger the same operation to Lucene and eXist in some way (???).

2. Web based access to XML-documents
Lucene and eXist together are the systems which fulfill the requirements. They both are also open software. Other software such as XML-DB, Tamino, DB2, Postgres could also be examined later.

Requirement	**Lucene**	**eXist**
Customizable (Sources extensible)	relative easy	not easy
Fulltext querying for XML-docs (logical operators, wildcard searching)	Yes	yes (but too slow)
Stemming in different languages	Yes	No
Date range queries	Yes	yes
Fuzzy operator	Yes	yes (NGrams)
Near operator	yes	yes
Case sensitive querying	with filter	?
Searching for XML-documents by attributes	Yes	yes
Structural querying of XML-documents	No	yes
Structural querying in one XML-document	Programmable with Java and XQuery	yes
Different document bases for specific languages	Yes	yes
Content encoding in UTF8 (query and content)	Yes	yes
Query results are customizable	by Java	by XQuery
Query results could be sorted by relevance	Yes	no
Query results could be sorted alphabtically by fields (author, publication year)	yes	yes (?)
Web interface for querying	yes (JSP)	yes (XQuery)
Maximal size of XML-document	> 2 GB, efficiency is no problem	theoretically as big as the maximum size of a file on the file system; but system hangs if it is too big (> 100 MB)
Maximal number of XML-documents	> 1.000.000	2^{31}
Saving of relational data (for specific needs)	No	Yes
...		

...		

Lucene could be the system for fulltext querying (with stemming, etc.). eXist could be the system for structural queries.

Web engine: Tomcat as JSP-Container, WAR-application files

3. Establishment as a central service
Central server maintenance: Backup-software. Installation of Tomcat and Lucene and eXist. Installation of the development software on local workstation (Eclipse Ganymed EE, Tomcat). Documentation of all produced software (Java-Doc, Wiki).

Needed hardware

1. Developer-Client-Machines: Mac OS X computer or notebook with relevant developer software

2. Server: need of a linux server for the document bases and the server software (Tomcat, Lucene or eXist). Later: Tomcat as a plugin into the official Webserver of the MPDL-project

What already works (on my local MacOS computer):

- Tomcat Container
- Eclipse Ganymed EE (as Java development tool and for deploying WAR-files to Tomcat)
- Lucene (with language specific extensions for stemming), Luke (Lucene index viewer)
- Example-Java-Programs for indexing and querying Lucene (on local Lucene index): with language specific indexing and querying (stemming in english, german, chinese)
- Example Lucene-Tomcat application (WAR-File) for querying
- HexEdit 2.20: an Hex-Editor for UTF8-files