**Research Proposal for a Ph.D. thesis**

# Towards Distributed Mathematical Knowledge Management

Vyacheslav Zholudev
April 3, 2008

School of Engineering and Science,
Jacobs University of Bremen,

v.zholudev@jacobs-university.de

**Abstract.** My work is devoted to research in domain of the collaborative databases for Mathematical Knowledge Management (MKM), and in particular for the documents in OMDoc format.

The issue of collaborative editing and sharing mathematical knowledge is very significant. Up till now mathematicians prefer the traditional way of collaboration, with pen and paper, in spite of a technical potential. The goal of my research is to propose a system aimed on the facilitation and enhancement of collaboration between mathematicians working even with huge amounts of mathematical knowledge. The name of such a system - *OMBase*.

# Contents

# 1 Introduction

With the rapid growth of computers and internet resources the communication between humans became much more efficient. The amount of electronic mathematical knowledge is also rapidly growing. If Lomonosov had to wait a number of months before his mathematical letter reached the addressee, now we can exchange electronic letters almost instantly. Or some decades years ago we had to go the library to read about new theories, nowadays we can easily download a pdf file containing relevant information.

But nevertheless many mathematicians prefer collaboration with each other in traditional way, figuratively speaking, using pen a paper. Why is that? Because of an unrealized potential of the web for mathematics, because of a lack of mathematic-specific applications and the absence of distributed storage for mathematical knowledge. We will cover all these items in turn.

Mathematicians need not a surface, but a deep web for mathematics. What does it mean? The surface web is a portion of World Web Wide that could be indexed by conventional search engines. e.g. Google, Yahoo or Live Search. This search is not deep in a sense that it is based on text occurrences, but not on semantic information. When we are talking about the deep web, we imply that data are stored in the databases and the content is generated automatically. We want something similar regarding mathematics. What are the possible benefits of having a deep web for mathematics?

- Search. We want not just a search engine that works similar to one which is based on text search, but we want to be able to seek for a mathematical knowledge that satisfies the general view of the mathematical entity, e.g. formula, definition, theorem. It is very significant to search for a non concrete formula, but for a set of formulae which satisfies common conception of scientific notion, since human tends to remember not every detail, but general view of the mathematical conception.

- Generation of the documents on the fly. As we mentioned before the prerequisite of the deep web is the ability to generate content out of a knowledge database. Regarding mathematics we should be able to retrieve mathematical objects according to the special mathematical requests, e.g. obtaining all definitions from a spring semester of the General Computer Science lecture notes. It will reduce the time and efforts of exploring and editing mathematical material.

- Presentational documents. Mathematical documents might look differently depending on the community preferences. For instance, binomial coefficients may look like: $\binom{n}{k}$ $C_k^n$ $C_n^k$. So the possibility of obtaining representational documents depending on established mathematical notations seems to be worthwhile. Also such possibility includes the support of multilingual documents that facilitates the distribution of the mathematical knowledge.

The invention of OpenMath [BCC+04] and MathML [ABC+07] (and finally the OMDoc format [Koh06]) initiated the process of solving a lot of representational problems, but the mathematical world needs not only the sufficient format for representing

mathematical knowledge, but the applications which are able to deal with such format. For instance, the following applications support OMDoc natively: the math search engine [KŞ06], the collaborative community-based reader panta rhei [Mül07], the semantic wiki SWiM [Lan08], the learning system for mathematics ActiveMath [Act07], the system for the verification of statements about programs VeriFun [ver08], etc. But the implementation and the experimentation of the mathematical knowledge management (MKM) systems are hampered because they have to worry about storage. A lot of MKM directions already have some working prototypes of even fully integrated applications, but there is a way to stimulate the development process of such applications. Let's discuss it further.

Let's mention that the cooperation between such applications sometimes is very important, because we may gain from their composition. For instance, if one service provides the generation of courses and another - mathematical search, we can gain the mathematical search among such courses. So a general solution is to have an universal format which could be used by these applications. The OMDoc format seems to be a reasonable choice since it stands on top of MathML and OpenMath and has a lot of additional capabilities. Moreover OMDoc is the fully standardized, emerging and highly expressive format that can markup mathematics, but not only mathematical formulae.

But unfortunately the main part of OMDoc environment is missing: *the database for the OMDoc format*, which is called *OMBase* later in this proposal.

Of course mathematical knowledge could be stored in a local file system or version control system, but it is not the best way to store and share ideas easily, reliably and efficiently, because such variants do not care about content and therefore can not gain from semantic information contained in the OMDoc collections. Apparently a new approach is needed. A lot of steps towards improving the situation are made so far, but they are not sufficient. So OMBase should not only be able to store OMDoc documents, but do a lot of math-related jobs such as searching, obtaining presentation of documents and efficient querying. This eliminates the necessity to implement such tasks in the different MKM systems and therefore facilitates the interoperability of such systems. Moreover it will allow systems to communicate on a higher level, e.g. not on the level of XML or even byte streams, but on the level of OMDoc. Thus the idea of higher level communication entails the facilitation of development MKM systems, because the developers would focus on application specific ways more, but not much on collaboration. More MKM systems that able to easily communicate with each other we have, then more motivation from mathematicians side to use them we might see.

We were talking about MKM systems so far, but since the OMDoc is an XML markup language, the proposed storage can be adapted for using another scientific XML markup languages, e.g. for physical markup language PhysML [Phy08], chemical markup language CML [MR+07] and other markup languages for the natural sciences. And furthermore, such sciences have some intersections, therefore the development of OMBase would facilitate collaboration not only inside mathematical community and between mathematicians, but also between scientists of the different hard sciences.

Thus, OMBase should stand as a interlink between MKM systems via providing a joint storage. So this paper is devoted to challenge the creation of a new database adjusted

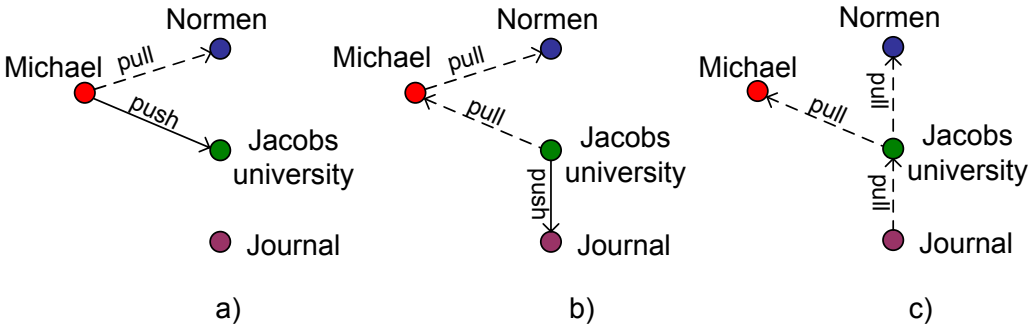


Figure 1: Running example

to the OMDoc format.

## 1.1 A Working Example

In Figure 1 I provide a real scenario from a scientific life that will be used later in this proposal to explain some features.

In Figure 1a Michael started to work on his paper with future intentions to propagate it to Jacobs University. During the creation Normen wants to have a cache copy of a Michael's paper on his computer and look after the changes. From time to time Michael pushes his work to Jacobs University and the corresponding people at Jacobs checks the correctness of the paper. Then assume Figure 1b. When everything is done from Michael's side he wants to pass the rights for primary editing to university and only receive updates from it. Notice that Normen still depends on the Michael's updates. Now Jacobs University propagates its changes of the paper to some Journal and its stuff validates the correctness. Here is the same scenario as with Michael and Jacobs University in Figure 1a. Finally (see Figure 1c) Jacobs University passes the rights for original editing of the paper to a Journal (like Michael did with Jacobs) and Normen decides to switch the source of cached copy to Jacobs since he thinks that Jacobs contains more actual information.

I will illustrate by this example the concepts devoted to the distribution, presentational and aggregated documents, etc.

## 1.2 Structure of this Document

**Related work** This part describes the projects which are related to my research. The kind of relation can differ depending on project. Some of them will be used as a basis for a proposed database (see sections 2.1 and 2.2). Some work are planned to be integrated with OMBase and maybe partially re-implemented (see sections 2.3 and 2.4). The JOMDoc project (see section 2.5) is related to a third type of related work: the project which will be used inside OMBase as an instrument of a higher level manipulation with OMDoc, and probably inside Locutor [loc07] for a client-side processing of OMDoc.

**Vision** The corresponding section tries to answer on the following questions: what is my contribution, what is new in my research, what apart from the implemented system I will give to the science world, what concepts will be elaborated? These questions are very important since they help to plan work and focus on the particular directions, but not deal with purely technical work and integration with the other systems.

**Technical side** In this section we discuss questions which are more specific and more technical than in previous section. It is a right way to have not only abstract vision but prove it in some sense by providing a technical part of a vision even on the early stages. The described at the beginning use cases helps me to make a good feeling about requirements to the system and therefore plan a core and the architecture of OMBase in more or less details. Also some technical open questions are provided at the end of this section, which gives me a better vision on the practical part of my research.

**Approach for getting results** During sections 5, 6 I am discussing domains of applications of OMBase and work that should be done to check preliminary reasonableness of the proposed system. The work schedule is also described and tends to be important since it gives a view on the research process and allows to evaluate the success of the implementation.

## 1.3 Objectives

The objectives of my thesis are:

**O1** Facilitate the mathematicians' collaboration via building the distributed model of the mathematical databases (section 3.3).

**O2** Develop more intelligent, efficient and reliable storage for OMDoc collections of documents then file system or version control systems. What I mean by intelligence will be discussed in the section 3.2.

**O3** Facilitate the interoperation of math-specific application using developed storage for mathematics (see section 4.4). This holds for the potential OMBase clients such as SWiM, Panta Rhei, ActiveMath, etc.

**O4** Try to get rid of notion of files as much as possible. Instead mathematical objects should substitute files and be treated as working units in a content of commons. In particular, we should support versioning on the level of the math objects. The benefits of having this we will discuss in section 3.4.

**O5** Develop mathematical query language (*MathQL* later in this proposal) for retrieving particular parts of documents or collection of documents according to user preferences (e.g. all examples from the lecture slides). More information in section 3.5.

**O6** Integrate management of change in OMBase, i.e. support consistency on the level of collections of documents considering interrelationships between mathematical objects (section 3.7.2).

Listed objectives answer the question "What is new in my ideas?" and "What is my contribution?". Thereby, the beliefs of this thesis work are serious contribution to the collaboration of mathematicians, efficient and reliable storing of OMDoc documents and development of easy-to-use MathQL for effective navigation and retrieving information out of a huge amount of mathematical documents.

# 2 Related Work

The work described in this proposal is built on some existing open source projects. They are described here in brief, we will point out how OMBase will differ from them.

## 2.1 Subversion

Subversion [SVN07] is one of the most popular open-source client-server version control systems. Subversion (SVN) maintains versions and history of documents and directories in a repository. The users check out the working copy of repository to a local working space. They can do various things with local copy: change, update from repository or propagate changes backwards to repository. *Update* does merging of working copy with latest version in repository. In case when automated merging is not solvable, user has to edit conflicting files manually. Afterwards to propagate changes to repository user has to *commit* his changes to repository. Here only basic concepts are mentioned, but that is enough to get a rough conception of SVN.

Subversion has the basic functionality we need for versioning and therefore Subversion becomes a base-line for the proposed system. But unfortunately Subversion does not have the functionality for distribution. Of course there are some distributed version control systems, e.g. GiT [GIT07], but in this case user is obliged to cache the whole repository on his local computer. That is not desirable for us, since it implies storing all history of changes on a client side, what may cause substantial increase of allocated space for the mathematical documents, worse performance and necessity of having *all* content of OMBase. Of course, we do not need all of these.

Instead of pure versioning the proposed system should be able to be aware of content in such repository and therefore efficient do domain-specific tasks. Also for end user the versioning should be presented on the level of object (e.g. mathematical objects), but not the level of files, i.e. user should be able to checkout particular object and make changes with regard to that object, not only to the file containing the object. Furthermore initially the file containing the object could not even exist, e.g. in case when we obtain an object through a stream from another OMBase. Roughly speaking we should get away from file metaphor.

## 2.2 Berkeley DB XML

Berkeley DB XML [Ber07b] is an open-source, XML-native embedded database. Embeddedness means that it is distributed as a library with a number of APIs. This approach does not have an overhead by having surrounding environment like servlets or standalone servers. Also the embeddedness eliminates some database administration costs. Berkeley DB XML (BDB XML) is built on top of Berkeley DB [Ber07a]. Berkeley DB is a open source, embeddable, relational database with zero administration. Thereby BDB XML inherits its advantages and features, e.g. portability, transactions, replications, easy deployment, etc. Also BDB XML has all features which should be expected from XML-native database: XQuery-based [XQu07] access to documents (last version also with support of XQuery Update facility [XQU08]), support of transaction, content-based indexing, hierarchy structure of storing documents, productivity even with huge amount of data, etc. It is planned to build OMBase on top of BDB XML (advantages of such decision we will cover later), but nevertheless the choice of this database is not final, thereby the other possible XML-native database which could be taken into consideration are: eXist [eXi07], X-Hive [X-H07], Sedna XML DBMS [Sed07]. Such preliminary choice is caused by a paper of Ronald Bourett [Bou05], investigations of author and conversations with experienced persons.

Here we should mention importance of not only BDB XML in particular, but of XML-native databases themselves. Since OMDoc is an XML-language, their functionality constitutes the set of XML-specific operations we expect to process OMDoc collections in a right way.

## 2.3 The Locutor Client

Locutor [loc07] is an ontology-driven system for management of change. Currently Locutor uses the Subversion server and substitutes the Subversion client with a Java-based re-implementation. In contrast to the Subversion client, Locutor is based not only on standard diff-, patch-, and merge-algorithms, but implements its own algorithms and also is able compute long-range effects of changes and thereby prevent inconsistency of documents. A serious limitation of such a system is that it uses ordinary Subversion server which does not care about content. So one of the hopes of this thesis is to have not only an server for OMDoc documents, but also integrate it with intelligent client like Locutor. Of course the efforts from client side are required as well. Furthermore OMBase may also stand as a client when it speaks to another OMBase, e.g. propagates some changes or updates the content. Such kind of mentioned collaborations is worthwhile and the aim of the current thesis, because we do not want to reinvent the wheel, but gain all profits from the existence of the relevant projects that allows us to focus on the unexplored features.

This system is being implemented by Normen Müller, who is also a PhD student of KWARC research group, and that will simplify the collaboration.

### 2.4 A Search Engine for Mathematics

A search engine for mathematical formulae [KŞ06] is implemented in the MathWebSearch system. It allows the lookup of math fragments basing not on presentation, but rather on semantic information, i.e. structure and meaning. This allows to search for wider range of mathematical formulae and use a richer set of queries. For instance, the typical situation is when user remembers roughly the formulae (i.e. only general view) and wants to look up for every formula that satisfies his imagination of its form. This easily possible with described search engine.
It should be integrated in OMBase to allow users not only efficiently edit and retrieve different parts of documents, but also search for some pieces of mathematical documents. This part of the research is not first and foremost, but extremely relevant to build deep web for mathematics.

The first version of this project was implemented by former student of KWARC group Ioan Şucan, and now it is evolving by Constantin Jucovschi and Ştefan Anca.

### 2.5 The JOMDoc Project

The JOMDoc project [JOM07] is a Java library that provides a Java API to work with OMDoc documents in more object-oriented way rather than parsing pure XML. This library is able to build a representation of OMDoc documents in memory, work with content like with Java objects, and serialize memory representation back to XML.

Thereby JOMDoc come out like a high-level instrument for dealing with OMDoc. It is planned to integrate such library to OMBase and a client for OMBase. On the server side JOMDoc should do some specific jobs related to retrieving mathematical information. On the client side JOMDoc should be able to perform some actions on received files without communicating with a server.

This project is important for us because it eliminates the necessity to do a lot of tedious XML-specific work and allows us to abstract away from XML elements and concentrate on the OMDoc-specific tasks.

JOMDoc is supported now by internship member of KWARC group Kristina Sojakova and therefore it will be easy to solve possible problems with such a library.

## 3 Vision

### 3.1 Why OMDoc? Why Do We Need a Storage for it?

**OMDoc** Of course, OMDoc format is a core of my research group and there is nothing surprising that my PhD thesis deals with it. But in this section I will try to convince the reader that OMDoc is a right thing to base my work on and my direction of a research is reasonable and beneficial. As was mentioned above OMDoc is a *content-oriented* markup language for *all* mathematics. The content orientation allows us to gain as much as possible from the semantics encoded into OMDoc documents. The more we know about semantics, the more we could benefit out of it. Also OMDoc is able to

cover all mathematics content, therefore we are not restricted to the particular field of mathematics. Moreover, OpenMath and MathML are the subsets of OMDoc, and they are standards for representing mathematical formulae what obviously facilitates interoperability.

But nevertheless OMDoc is only a tool for proving the reasonability of the concepts listed in section 1.3. These concepts like fine-grained versioning, decentralized distribution or having a format-specific query language are worth realizing for the other markup formats as well, whether it is CML or some format for representing printing documents. So the practical part of my thesis should show that with OMDoc things also work, but technically can be used analogously.

**Storage for OMDoc** In section 3.2 we will discuss the aspects why the proposed storage is better than the traditional ways of storing documents like file systems or version control systems. But in this sections we are going to think on the more abstract level about why we need such a storage.

The universal storage will stimulate the development of the math-specific applications since they will have a core to be based on. OMBase will deliver them away from worrying about storage and doing tedious XML-specific jobs, what obviously significantly saves time and efforts. Instead of that developers will focus on a business logic. Furthermore, OMBase will eliminate a lot of negotiations between counterparties about a way of communication, which actually takes a lot of time in a real life.

## 3.2 Intelligence of the Proposed Storage

Now we will discuss the benefits of the proposed storage in comparison to file system or even version control system. The goal of this section is to convince a reader in a reasonability of having such a system. So the implied profits are:

**Efficient storing and querying** Since we know about a format of our documents we can use such information to make storing and querying more efficient. For more information see section 4.2.

**High-level querying** Using file system or version control system we can query only for files and make a text search inside them. Even if we use some XQuery engines to process OMDoc documents we work on the level of XML. But we want a better abstraction. Therefore the proposed system should have a more high-level language and engine to do such work (see section 3.5).

**Versioning on the level of objects** The users want a better level of abstraction and work with OMDoc on the level of the mathematical objects (e.g. theorems, definition, examples, etc.) since such approach is more intuitive and understandable (see section 3.4).

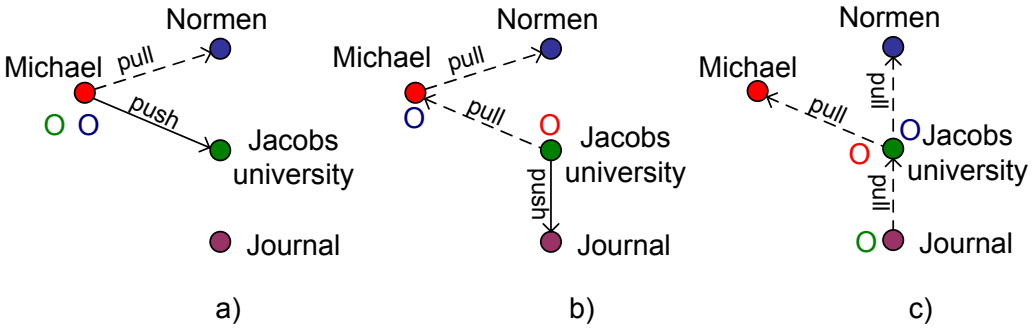


Figure 2: Distribution model

**Generation of the aggregated documents** As was mentioned above, the way of working with the mathematical knowledge on the level of the mathematical objects seems to be more promising. For this purpose OMBase should be able to generate so called *aggregated documents* on a user demand which contains right information. To define a content of a particular aggregated document we should use the high-level query language for mathematics - MathQL. A good example of an aggregated document is a document that contains all definitions with examples from a particular theory. Of course such documents are also exposed to a versioning.

**Integration** OMBase will not only incorporate new features but integrate some projects dealing with OMDoc. For example, to provide a possibility to get the representational documents I want to use the mmlkit project [MMK07] (for details see section 3.7.1). Since OMDoc documents may have interrelations the consistency on the level of the collections of OMDoc documents should be also supported (see section 3.7.2).

**Distribution model** OMBase will support the decentralized distribution model that will help users to easily exchange their ideas and propagate their changes. For more details refer to section 3.3.

**Clients for OMBase** We need both the universal interface with a lighter functionality but a simplicity to operate with and a full functional client which wraps all communication routines and serve the versioning information. The freedom given by presence of such clients should satisfy every potential user whether it is a human or a mathematical-oriented application. Such clients are discussed in section 4.4.

## 3.3 Distribution model

**Original flags** Speaking about a distribution in OMBase we operate with a notion of the original flag. It is a relative notion and means where is the source OMBase for particular documents in current OMBase. Each file in the XML-enabled repository may have no original flag meaning that this OMBase contains original documents (i.e. or it

does not know about original source), or have some original flag which points to some counterpart OMBase. Ideally, we should be able to work with the original flags on the level of the mathematical objects, but not files. The original flag is not only a technical notion that helps us to manage the distribution of the materials, but also a meta-level entity that represents the logical dependencies between documents in a logical network of OMBases. Returning to our working example, Jacobs University may have dozens of papers in a status of a harmonization with the different journals and therefore Jacobs should have a way to manage all of these and know what papers it is responsible for, what papers should be kept up-to-date and what papers could be taken by another OMBases.

**Distribution model** Let's consider Figure 2, which is a slightly more detailed picture of the working example. Also I will repeat the general scenario, but with some particularities.

Assume that Michael started his work on a paper with intentions to share it with Jacobs university later. Normen also wants to see Michael's paper from time to time and possibly add something on his own purpose and not share changes with anybody. When Michael decides to propagate his changes to Jacobs OMBase he can use the *push* method, which is allowed when the destination OMBase does not have original flag of Michael's documents. When Michael pushes his changes to Jacobs OMBase, the latter gets the *original flag* that points to former OMBase. Normen can obtain Michael's paper using the *pull* method, which automatically assigns an original flag that points to Michael's OMBase. In Figure 2 the colored letters $O$ depict that particular OMBase is the original for another OMBase with corresponding color of letter $O$. The above scenario is represented in Figure 2a.

When Michael decides that he finished his work he can move the original to Jacobs OMBase using the *moving original* method. We can see this in Figure 2b. Now Michael can look at the Jacobs changes using the *pull* method. He can not push changes anymore because his OMBase is not original now for his paper. Notice that for Normen original flag still points to Michael OMBase. Also Jacobs university may decide that some journal needs their paper, so they can use the *push* method to propagate the paper. The original for the paper still belongs to Jacobs.

When the Jacobs work is finished, they may decide to give original rights to the journal. It can be done by using the *moving original* method. But for Michael the original remains in the Jacobs OMBase. Then Normen may realize that he wants to update his paper from Jacobs, but not from Michael. So if Normen has rights to access Jacobs OMBase he may change the original source. All these actions are depicted in Figure 2c.

**Reasons of having such a model** So far I proposed the decentralized distribution model for a distribution of the mathematical knowledge. But why do we need exactly this model for OMBase? Why do not we use centralized storage? Let's discuss the benefits of the proposed model:

- OMBase can contain only such knowledge which is needed by an owner. Thus OMBase is not overheaded that certainly beneficially affects the performance.
- Collaboration becomes easier and more transparent when we communicate directly between two OMBases, but not using central storage. Moreover, two OMBases could be offline for a central storage, but still be visible for each other. For example, in the local network without internet access.
- When we want to develop the confidential documents we could have a private OMBase on a local machine and exchange the information only with the particular collaborators. In this case the placement of such information to a central storage might be undesirable despite it is claimed that nobody could see you data without your confirmation.
- Central storage needs to be extremely powerful and involve clustering, replication, etc. For the huge amounts of data it would be very difficult to support such a system.
- If some users want to work with the different versions of the papers, it is more intuitive and transparent to get them from particular OMBases rather than from different branches from a central storage.

**Consistency of the documents** The OMDoc documents may have interrelations and editing of one document may cause inconsistency in the others. This also holds for a process of a distribution since it could change the particular documents but remain the others unchangeable. The solution of preventing the inconsistency is to integrate management of change to OMBase, and this topic is discussed in section 3.7.2.

## 3.4 Fine-grained Versioning and its Benefits

The basis for OMBase versioning is SVN versioning, but the substantial difference between them is that the former should support versioning of the level of the mathematical objects like theorems, examples, definition, etc., but not files.

Let's return back to our working example. Assume that Normen has his own paper and a paper from Michael. Then he wants to edit all examples in those papers. He uses MathQL to retrieve all examples to a single aggregated document, edits it and then commits changes. So the documents from which came the examples will be implicitly changed and versioned afterwards. Thus Normen should not care about which particular documents were affected, OMBase will do it for him.

So the main benefit of having fine-grained versioning is that we can work with multiple mathematical objects which are situated in the different files in the repository and be sure that all corresponding files will be versioned automatically.

## 3.5 Mathematical Query Language (MathQL)

MathQL is implied to be the OMDoc specific query language which allows to abstract from XQuery language and work on the level of mathematical objects. The concrete syntax is not discussed in this paper, but some ideas about it are provided in section 4.5. But in this section we are considering only the basic concepts and a related notion of the aggregated documents. Aggregated document is a document that is obtained from different documents according to some selection criteria. Such documents are often more intuitive and suitable for the particular needs of a user. For example, if someone is interested in the theorems of a particular author he can obtain an aggregated document that contains only them.

Thus MathQL is used for retrieving aggregated documents. The result of MathQL is a list of the mathematical objects, but not of more fine-grained structures like bound variables in formulae or a component of addition.

Also MathQL could be applied to the differences between the versions of the documents. Let's return back to our working example. When Michael pushes changes to Jacobs university, the latter wants to know not all differences, but only what definitions were added. So Jacobs expresses in MathQL a query intended to give all definitions from a difference between revisions. It is worthwhile since it may give us a quick feeling what conceptually was added or changed.

## 3.6 Reasons for High Performance

Since scientific documents may count hundreds of thousands items (e.g. the archive of scientific knowledge [ArX07a]), the question about efficiency arises a lot. According to the special features of OMBase, we should not only retrieve and save documents in an efficient way, but take care about the impactful aggregating, searching, distribution, etc. of the documents. Here we should mention that as one of the key features of the proposed system is the ability to work on the level of the mathematical objects, we should care about efficiency in a more fine-grained way, but not just on the level of files, i.e. be able to operate with the fragments of OMDoc in the implied ways.

## 3.7 Intentions for the Integration

In this section we will mention the most relevant projects which are worth integrating to OMBase.

### 3.7.1 Presentation of Mathematical Documents

Obtaining a representational documents is also important for OMDoc database. This task is shifted to the mmlkit project [MMK07]. It is a Java-based toolkit for building representation engines for content markup formats for mathematics. It uses notation definition as a parameter. The notion of notations is defined in MathML3 working draft [W3C07]. Mmlkit should be integrated to the server part of OMBase as a component. The notation definitions should either be provided externally or be searched in

OMBase. To elucidate this consider the working example. Michael is writing a paper, but Normen and Jacobs university are got used to the different notations. So Michael can provide two notations definitions for his paper, and Normen and Jacobs will choose the most appropriate notation for them.

### 3.7.2 Management of Change

Of course it is not enough just to check a well-formedness of OMDoc to support consistency of the documents. The OMDoc documents contain references to the other documents and therefore the changes in some document may affect related documents and they could become not consistent anymore. So OMBase needs a method for calculating such long-range effects (the violation of consistency) and prevent changes that entail inconsistency. The Locutor project is being developed specially to support consistency in collections of the documents. So the idea is to make the best of it and integrate such a system to OMBase.

# 4 Technical Details

## 4.1 Use Cases

The OMBase system should satisfy a number of heterogeneous requirements. Here different possible use cases are discussed. They will allow to understand us how we see OMBase, what fields I should focus on, what the base-line features are and what the secondary ones are. Also each use case represents the separate direction of my research and it helps to separate the tasks and plan work more concretely.

**U1** Retrieve a document or parts of a document. This should be done according to a new addressing scheme described in [RK08].

**U2** Upload documents to a database into a particular collection of documents. The useful feature would be the possibility to upload some files via a single request.

**U3** Delete document or collection of documents from a database.

**U4** Retrieve an aggregated document obtained from different documents according to MathQL query. For further details of this and above use cases refer to section 4.4.

**U5** All operations should have another dimension, temporal, i.e. versioning should be supported. The user should be able to operate with branches, different versions of the same documents like it can be done in usual version control systems like SVN or CVS [CVS05] (see section 4.2). But versioning should be supported in a more fine-grained way(see section 3.4).

**U6** Query and change documents via XQuery. This level of work seems to be low-level, but can be useful for some kinds of possible clients (see section 4.2).

**U7** Propagate changes to other OMBases according to the distributed model described in the section 3.3.

**U8** OMBase should have user rights administration. Different users might be allowed or not to read, write, administrate OMBase, propagate changes to other OMBases. Also OMBase should be able to store and maintain licensing information and probably not allow to store unlicensed content. I will not cover these issues in the proposal since it implies a high-level of details that does not correspond to an abstract level of this paper.

**U9** Obtain the presentation of OMDoc documents in some format, e.g. in HTML or PDFs (see section 3.7.1).

**U10** Search mathematical documents using special search query language, described in [KŞ06].

**U11** Ability to extend functionality of OMBase by providing the additional modules. For this purpose the plug-in architecture of OMBase should be elaborated. This is not covered here.

**U12** Possibility of querying mathematical knowledge on a higher level and elaborate mathematical query language, for instance for retrieving all slides for a particular course or all definitions and theorems from a set of theories (see section 3.5).

## 4.2 XML-enabled Repository

At the base level OMBase will be a version control system (like Subversion, CVS) which is optimized for storing XML documents. Thus it will allow to support versions of XML documents and folders which contain XML documents as well as do some XML specific jobs like efficient querying and updating via XQuery, indexing, making transactions, etc. The second part of proposed functionality is supported by some XML-native or XML-enabled databases. Straightforward versioning is also supported in temporal databases, but they do not have advanced features like most version control system do, e.g. branching, merging, rich history exploring. Thus the good solutions are either implement manually the own XML-enabled repository which would combine all mentioned capabilities or take some version control system and XML database and try efficiently combine them. The first is not an option for me since it needs much effort and is not a focus of this thesis. For base of version control system Subversion is proposed, for the XML database - the XML-native database management system Berkeley DB XML. Although the choice is quite reasonable, it is not final and I may change it during my PhD program. The explanation is separated into three parts. The first two contain separate explanations of why the concretely these technologies were chosen, and the third contains advantages of being combined together.

### 4.2.1 Subversion

Generally we should choose the basis for versioning in OMBase. Of course it could be implemented from scratch, but there are a lot of technical work that is not the aim of current research and plenty of satisfiable version control system exist. Two general groups of version control systems could be marked out: distributed (e.g the fast version control system GiT) and client-server version control systems (e.g. SVN or CVS). Even though distributed control systems mostly support client-server architecture they are intended for distributed repository architecture. But as we will see later I am going to support distribution in OMBase on another level, therefore distributed version control systems is not an option for OMBase in my vision. The client-server architecture is what we exactly need for a single instance of OMBase since it should serve as a server for storing mathematical documents, and different users of OMBase are clients in terms of version control system clients. Since Subversion is a very popular, reliable and efficient version control system and it is used by the Locutor system (which is also related to my research), I decided to choose exactly the Subversion system. Here it should be noticed that Subversion may use Berkeley DB as one of two underlying storages. This is also important aspect as we will see later.

### 4.2.2 Berkeley DB XML

To work with XML we could mark out two kind of databases: XML-enabled databases and XML-native databases. The first type uses relational databases to store XML in. Such databases are efficient in case of well-structured XML documents, but OMDoc documents are not such. The second type of databases is used to store semi-structured documents and stores XML natively. Since OMDoc documents are semi-structured, I decided to use the XML-native databases. For the sake of a clarity, let's provide some comments about well- and semi-structuredness. *Well-structured* documents imply not only existence of some sort of XML-schema, but the same structure for all documents. But OMDoc documents may completely differ, e.g. some of them contain theorems and proofs, others - definitions and examples. The order of OMDoc elements is also entirely unexpected. Such type of the documents are called *semi-structured*.

Even though there are a lot of solutions on the market of XML-native databases (for an overview cf. [Bou05]), only few of them deserve our attention. Here we will cover only advantages and disadvantages of Berkeley DB XML. The main advantages of such a database are being known as reliable database, with possibility of efficient working even with very huge amount of documents, supporting most of XML-native database related features such as XQuery, indexing, transactions. Also embeddedness of Berkeley DB XML means zero-administration from client-side, high efficiency in comparison to client-server databases since no time spending to communicate with two part of architecture. The consequence of embeddedness is that there are no high-level interfaces provided like the Representational State Transfer (REST) architectural style of communication, XML-RPC [XML08a], XML:DB [XML08b], WebDAV [WEB08] or SOAP [BEK+07]. Of course it is a kind of disadvantage since we have to implement higher-level communication

manually, but in this case only needed parts of functionality could be done and therefore interface might be more light-weight.

Also as was mentioned in section 2.2 the Berkeley DB XML uses Berkeley DB as the underlying storage of XML documents. And this fact together with others is a matter of discussion in the next section.

### 4.2.3 Symbiosis of Two Technologies

Beside of being a good choice, Subversion and Berkeley DB XML have some kind of intersection: they both use Berkeley DB for inner storage of content. This can be treated as an advantage of chosen technologies. Currently I could imagine two ways of XML-enabled repository's creation:

- The first one consists in an alteration of inner work of Berkeley DB XML to force it to read data from Subversion Berkeley DB instead of its own. The difficulties here are that Berkeley DB XML may use different format of storing documents that Subversion does. This question needs thorough investigation.
- The second way is to change Subversion inner workflow to make it work with Berkeley DB XML instead of its Berkeley DB instance. The open question here whether Berkeley DB XML provide an interface that is sufficient to do all things that Subversion does via Berkeley DB interface.

Although the two ways are very difficult to investigate and implement, since it imply going deeply in source code of such products, the approach of combining two tools on low-level seems to be very promising and will allow relatively high performance in comparison to approach when we use separately two products and combine them on a higher-level. Besides benefits with performance we should have all features that we expect from both sides.

In the current research it was decided to follow the second way of implementation since this method is implied by SVN developers, i.e. the source code of SVN is designed with the possibility to add a new databases as a backend for the SVN repository.

## 4.3 Global Architecture

In Figure 3 I show the architecture for OMBase and for the standard client, which will be discussed in section 4.4.2. Here only the server-side is covered.

The core of OMBase is the XML-enabled repository (see section 4.2.3), which is obtained by the substitution of the SVN repository's backend database by Berkeley DB XML. The XML-enabled repository should provide both the standard SVN interface and interface of Berkeley XML DB to allow work in the *SVN*- and *BDB XML*-like ways. Via the SVN interface we should be able to work in the SVN client manner, i.e. we should be able to ckeckout, update, commit, etc. mathematical knowledge. But the considerable difference is that the unit of the working process will be the mathematical objects, but not the OMDoc files itself. This feature will be discussed in section 4.4.2.

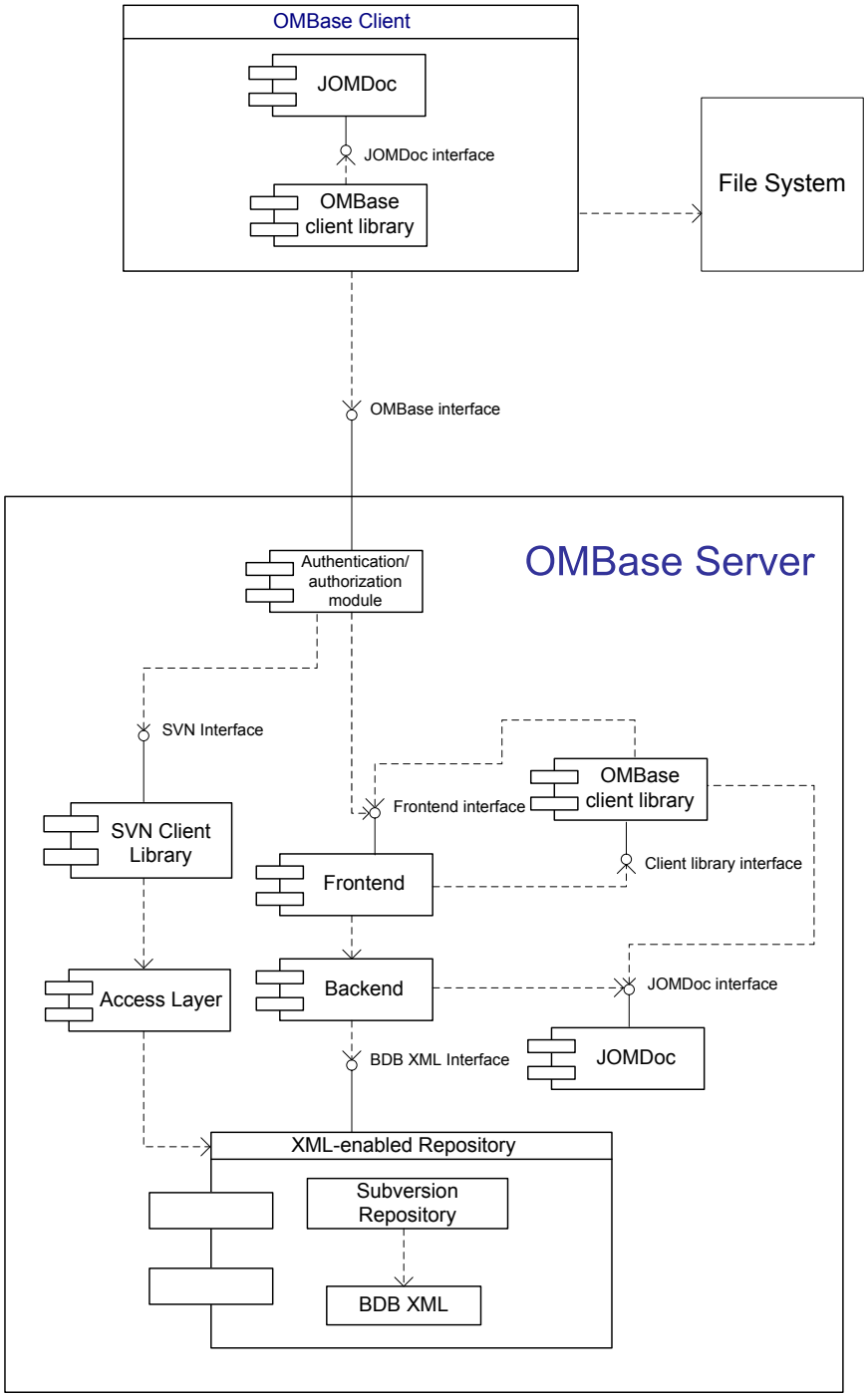


Figure 3: UML component diagram of the OMBase's global architecture

The Berkeley DB XML interface will be not visible to the clients of OMDoc. Instead of this, they will communicate via OMBase's *frontend component*, which is responsible to process RESTful and XML-RPC requests from clients. Here we distinguish two kinds of requests: requests for operating with the current OMBase and requests to communicate with counterpart OMBases.

The former requests are parsed and passed to the OMBase *backend component*. This component is responsible for employment the BDB XML interface, i.e. communicating with XML-enabled repository in XML-native database specific way. In some cases the data received from the XML-enabled repository need to be processed somehow. Here JOMDoc library comes into play. It processes parts of OMDoc documents obtained from XML-enabled repository. Then the results are returned to the frontend component which sends them back to the client.

The requests related to the communication with the counterpart OMBases after parsing by frontend component are passed to the OMBase *client library* which is responsible to take care about pushing/pulling changes to/out the other OMBases. Such a client library in turn may also use frontend component to retrieve necessary data out of XML-enabled repository. Here I should mention that the client library may also handle the JOMDoc library to process OMDoc on a higher-level.

The authentication/authorization module initially processes all requests from a client and redirects them either to the SVN client library or to the frontend component depending on the request type. Also such a module authenticates a user and figures out whether a client has enough rights to make a request. If not, the error message is returned back to the client.

The questions about specific requests and distribution will be covered later in this chapter.

## 4.4 Clients for OMBase

In this section we discuss the vision of what the clients for OMBase should be. The first section is devoted to how to work with OMBase without a client as discussed in section 4.4.2, i.e. which part of the OMBase functionality is available for other application or just every single user. The second section covers the basic ideas about the standard client of OMBase, which is intended to have extra functionality.

### 4.4.1 Simple Interface

The simplest manner to communicate with OMBase is to use HTTP-requests, either in RESTful style or XML-RPC. All requests could have a special parameter *revision* which allows to work with arbitrary revision in the database, but not only with a HEAD revision. RESTful requests serve for simple tasks like getting documents or parts of them, putting or deletion documents, or even querying and changing using XQuery. HTTP protocol has four methods of sending data: GET, POST, PUT and DELETE. Each method finds an communication application in OMBase.

More complicated queries to OMBase can be done via XML-RPC requests. In current

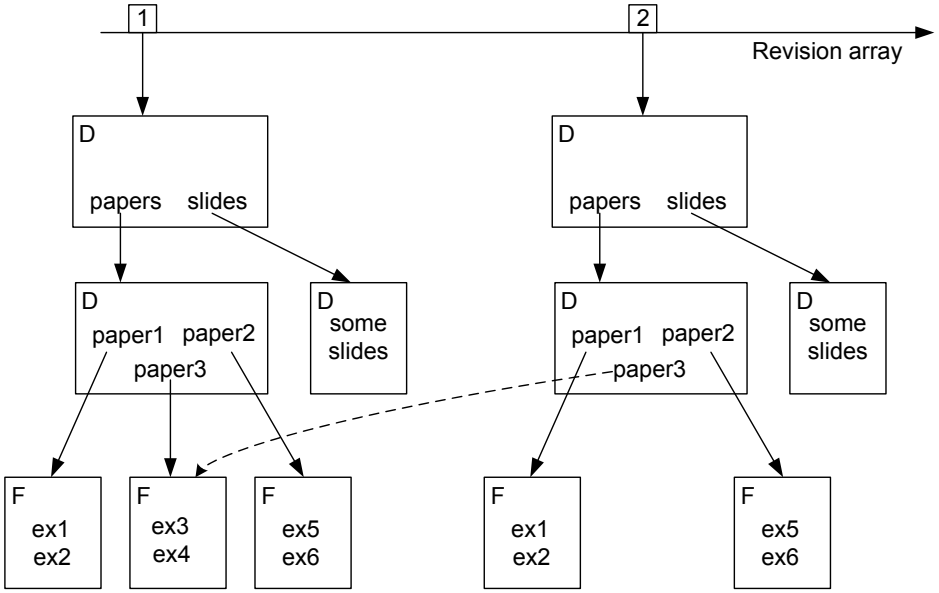


Figure 4: Repository revision array

proposal we will not cover all possible XML-RPC invocations and their signatures since it is not exactly defined what methods we need. The determination of this is a part of the current research. For example, via XML-RPC we should be able to search mathematical knowledge, receive aggregated knowledge according to MathQL queries, receive a representation of documents taking into account different notations.

### 4.4.2 The Standard Client for OMBase

The standard client for OMBase will be a wrapper over HTTP-request and also will support versioning management and distribution of the OMDoc documents. It could be either the set of the command-line utilities or even application with graphical user interface. We do not focus on this and put off the final decision.

The client scheme is presented in Figure 3. The core of the standard client is the OMBase client library component that is located both on the client and server sides of OMBase. First of all such library is responsible to make RESTful and XML-RPC requests to particular OMBase. Such requests were discussed in section 4.4.1 and the standard client allows not to think about the details of fulfillment the HTTP-requests and hides the way of processing them.

Another responsibility of the standard client is to support versioning, i.e. to work with OMBase like with version control system. The SVN client is a good example of how the version control management part of the client should look like. But the substantial difference between the standard client for OMBase and SVN client is that the former

should support versioning of mathematical objects like theorems, examples, definition, etc., but not files.

Let's assume Figure 4. At the top on XML-enabled repository is an array of revision numbers starting from 1 and lasting to infinity. Suppose we have our documents at revision 1 and we have the structure of the mathematical documents presented in Figure 4. In repository documents are stored in files, but OMBase tries to abstract from the notion of files in a way. The letter *D* in the left top corner in the rectangles means *directory* and letter *F* means *file*. Suppose that we made a MathQL query and obtained the document which contains the examples from the different papers: examples *ex1*, *ex3*, *ex5* from files *paper1*, *paper2* and *paper3* correspondingly. Then we edited the examples *ex1* and *ex5* in the generated document and now want to commit changes. The standard client will figure out which original documents in repository should be changed, then change them appropriately what triggers the creation of a new revision in the XML-enabled repository. Notice that since *ex3* was not changed the *paper2* is also unchanged and XML-enabled repository will create a symbolic link for the *paper2* to the first revision. The standard client also can use JOMDoc library to process the mathematical documents in some way instead of making potentially slower requests to the OMBase server.

The third conceptual role of the standard client is to support the distribution of the mathematical documents among OMBases. This question is covered in the section 3.3 in detail.

## 4.5 Mathematical Query Language (MathQL)

Here we will cover more specific details then in section 3.5.

Queries should be executed via XML-RPC requests and MathQL query should be passed as one of the parameter. The query language should have an ability to take into account the following parameters:

- List of the types of the mathematical knowledge, e.g. examples, assertions, theorems, etc.
- Date of the creation or modification elements.
- Owner, editor, co-author, etc.
- Patterns of the names of the mathematical knowledge, e.g. all theorems that contain the word *monoid*.
- Presence of some types of the mathematical objects inside other mathematical objects, e.g. all theorems that contain examples.
- Minimum and maximum number of returned results.

## 4.6 Open Questions to be Solved

**OQ1** The markup language for mathematical formulae OpenMath which can be used inside OMDoc also has its binary encoding which might be useful to save space and more efficient transferring data over the network. The analogue markup language MathML version 3 (but only the content-MathML part) is isomorphic to OpenMath, therefore we also can talk about binary representation of the content part of MathML language.

OMDoc uses both OpenMath and MathML to represent mathematical formulae, furthermore such parts can spend a lot of space in OMDoc document. Therefore it might be worthwhile to store OMDoc documents in the OMBase with binary fragments of OpenMath and MathML elements. But there are a number of upcoming some problems:

- In case we want to query not only upper elements of documents, but OpenMath and MathML elements as well it would be inappropriate to use binary encoding since we are planning to uses database XQuery engines to query documents. So during research it should be found out whether there are some use-cases where we need to query inside formulae.
- Since the Subversion is being planned to be used as a basis for version control system, using a binary format is not an efficient way to deal with versioning. But if the binary format spends not very much space in comparison to usual encoding, such storage might make sense since changing binary parts will not entail huge space consumptions. Efficiency of binary representation is investigated by guided research in KWARC group. The decision will be made depending on the derived results of this investigation.

**OQ2** How to implement search in OMBase? A good promising technique already exist. This technique is called MathWebSearch [KŞ06] and was discussed briefly in section 2.4. But there come the questions about integration: where would we store indexes? how would the search engine work if we store mathematical formulae in binary format? would OMBase and index servers be located on the same machine? Another way to do search is to use XQuery, but in this case we should not use binary representation of formulae. Also the efficiency of two approaches needs to be investigated. All of the above questions are long-term and will be taken into consideration after building the basic functionality of OMBase.

**OQ3** How should we store the information about the original flags? We can do it either inside OMDoc documents or somehow keep track of flags in some auxiliary registry. The former approach allows us to use XQuery to find necessary information concerning original flags, but forces us to change the documents when the original flag moves. The latter approach allows to avoid such problems, but needs supplementary structures to be processed.

**OQ4** How should the standard client preserve versioning information and figure out from which files the parts of the generated document came? This is indispensable

for supporting versioning on the level of mathematical objects. The preliminary decision is to assign unique ids to the every object in the generated document and keep track of sources in meta data of the standard client. Another research question is where should the inverse process of aggregating the document happen, i.e. where should the ascertainment of which original documents were changed happen. We can try to implement it either on the client or on the server and the proposed architecture in Figure 3 gives us such freedom. The question about what tasks should be processed on the client side and which on the server is also under consideration.

# 5 Case Study

Initially two case studies will be undertaken to evaluate the reasonableness of the proposed ideas.

## 5.1 The Connexions System

The Connexions (CNX) system [CNX07] is a developing collection of free educational materials and free open-source software tools for maintaining such materials. This software facilitate the publication of materials, the collaboration between authors, instructors and learners, building and exploring courses as well as related notions and concepts. Connexions currently uses a relational database for storing meta-data and CVS for storing modules which are in XML (or even in MathML or OMDoc) format. Thereby OMBase could be integrated to the Connexions system as a storage for the various modules to enhance the productivity of processing the storage units. In other words OMBase could be used as an intelligent database for Connexions data that eliminates the necessity of doing a lot of XML-specific work.

## 5.2 An Archive of Scientific Knowledge

The archive of scientific knowledge [ArX07a] contains a huge database of indexed scientific papers in LaTeXformat. The disadvantages of this storage is that it contains documents in presentation way, rather then in content. On the contrary, the OMDoc format allows to exploit semantic markup information to effectively process scientific documents.

The arXMLiv project [ArX07b] deals with converting Arxiv's documents to MathML. Therefore OMBase could be used as an intelligent storage for translated documents. This is a great opportunity to test efficiency and stress-tolerance of proposed system since storing converted Arxiv's documents implies the multitude users and the enormous amount of data.

# 6 Preliminary Work and Schedule

Figure 5 represents the schedule of my preliminary work plan for my thesis, which is divided into five major parts:

**Requirements gathering** The first two months were devoted to requirements gathering and realizing what we exactly want from OMBase. To understand it better, the overview of related technologies and papers was done. It helped to understand what has been done so far and what we can benefit from, and what is needed to be implemented manually.

**Analysis** After gathering the requirements there was a two month step towards the analysing of the methods needed to be undertaken.

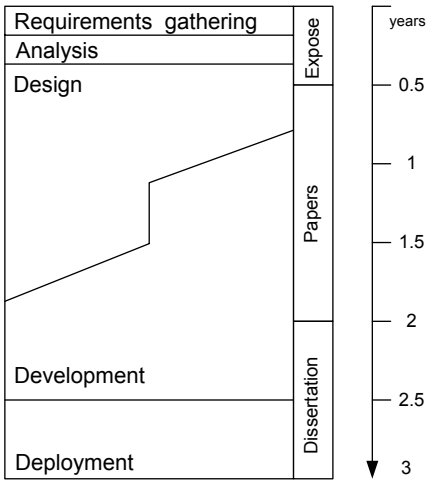


Figure 5: Preliminary schedule

**Design** The design phase is very important for the software development. The well-designed architecture is a key to success. That is why this step took one month only to draw a good preliminary picture towards the OMBase. Furthermore not everything was defined. This step will be repeated in a way during the development. Also this stage is for solving the open questions in section 4.6.

**Development** As was mentioned above this phase intersects with previous very closely. During the development the testing on case studies is necessary to avoid conceptual mistakes which can cause the insuperable obstacles in the future. The main challenge during development is to implement the core of OMBase: the XML-enabled repository. This task in not trivial, but very promising. All key features and power of OMBase are tight to the XML-enabled repository. This stage in combination with the previous one should take all rest time except a half of a year.

**Deployment** Last six months should be devoted to summarize the results and conclude the dissertation basing on the results and papers. Possibly some parts of the PhD thesis will be contained in research papers initiated by developing OMBase.

# 7 References


[ABC+07] Ron Ausbrooks, Bert Bos, Olga Caprotti, David Carlisle, Giorgi Chavchanidze, Ananth Coorg, Stphane Dalmas, Stan Devitt, Sam Dooley, Margaret Hinchcliffe, Patrick Ion, Michael Kohlhase, Azzeddine Lazrek, Dennis Leas, Paul Libbrecht, Manolis Mavrikis, Bruce Miller, Robert Miner, Murray Sargent, Kyle Siegrist, Neil Soiffer, Stephen Watt, and Mohamed Zergaoui. Mathematical Markup Language (MathML) version 3.0. W3C working draft, World Wide Web Consortium, 2007. Available at http://www.w3.org/TR/MathML3.

[Act07] ActiveMath, seen February 2007. available at http://www.activemath.org/.

[ArX07a] arXiv.org e-Print archive, seen December2007. web page at http://www.arxiv.org.

[ArX07b] arXMLiv: Translating the arXiv to XML+MathML, seen December2007. web page at http://kwarc.info/projects/arXMLiv/.

[BCC+04] Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaetano, and Michael Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004. http://www.openmath.org/standard/om20.

[BEK+07] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple object access protocol (soap) 1.2, 2007. available at http://www.w3.org/TR/SOAP.

[Ber07a] Berkeley DB, seen December 2007. available at http://www.oracle.com/technology/products/berkeley-db/index.html.

[Ber07b] Berkeley DB XML, seen December 2007. available at http://www.oracle.com/database/berkeley-db/xml/index.html.

[Bou05] Ronald Bourret. XML and Databases, September 2005. available at http://www.rpbourret.com/xml/XMLAndDatabases.htm.

[CNX07] Connexions. web page at http://cnx.org, seen January2007.

[CVS05] Concurrent Versions System: The open standard for Version Control. Web site at http://www.cvshome.org, seen August 2005.

[eXi07] eXist database, seen December 2007. available at http://exist.sourceforge.net/.

[GIT07] Git - Fast Version Control System, seen September 2007. available at http://git.or.cz/.

[JOM07] JOMDoc Project, seen December 2007. available at http://kwarc.info/students/projects/jomdoc.html.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, 2006.

[KŞ06] Michael Kohlhase and Ioan Şucan. A search engine for mathematical formulae. In Tetsuo Ida, Jacques Calmet, and Dongming Wang, editors, *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*, number 4120 in LNAI, pages 241–253. Springer Verlag, 2006.

[Lan08] Christoph Lange. SWiM: A semantic wiki for mathematical knowledge management. web page at http://kwarc.info/projects/swim/, seen February 2008.

[loc07] *locutor*: An Ontology-Driven Management of Change, seen June 2007. system homepage at http://www.kwarc.info/projects/locutor/.

[MMK07] Christine Müller, Normen Müller, and Michael Kohlhase. mmlkit - a toolkit for handling mathematical documents and mathml3 notation definitions. mmlkit v0.1 at http://kwarc.info/projects/mmlkit, seen November 2007.

[MR+07] Peter Murray-Rust et al. Chemical markup language (CML). http://cml.sourceforge.net/, seen January 2007.

[Mül07] Christine Müller. Panta Rhei. Project Home Page at http://kwarc.info/projects/panta-rhei/, seen August 2007.

[Phy08] PhysML - Capturing the Content of Physics, seen February 2008. available at http://www.omdoc.org/projects/physml/.

[RK08] Florian Rabe and Michael Kohlhase. A web-scalable module system for mathematical theories. Manuscript, to be submitted to the Journal of Symbolic Computation, 2008.

[Sed07] Sedna XML DBMS, seen December 2007. available at http://modis.ispras.ru/sedna/index.htm.

[SVN07] Subversion, seen May 2007. available at http://subversion.tigris.org/.

[ver08] VeriFun: A verifier for functional programs, seen February 2008. system homepage at http://www.verifun.de/.

[W3C07] W3C. Mathematical Markup Language (MathML) Version 3.0 (Third Edition). http://www.w3.org/TR/MathML3/, 2007. Seen November 2007.

[WEB08] WebDAV - Web-based Distributed Authoring and Versioning, seen February 2008. available at http://www.webdav.org/.

[X-H07] X-Hive database, seen December 2007. available at http://www.x-hive.com/products/db/index.html.

[XML08a] XML-PRC, seen February 2008. available at http://www.xmlrpc.com/.

[XML08b] XML:DB - Application Programming Interface for XML Databases, seen February 2008. available at http://xmldb-org.sourceforge.net/xapi/.

[XQu07] XQuery: An XML Query Language, seen December 2007. available at http://www.w3.org/TR/xquery/.

[XQU08] XQUpdate: XQuery Update Facility 1.0, seen February 2008. available at http://www.w3.org/TR/xquery-update-10/.