Building The Czech Digital Mathematics Library upon DSpace System

Vlastimil Krejčíř

Masaryk University, Institute of Computer Science,
Botanická 68a, Brno, Czech Republic
krejcir@ics.muni.cz
URL: http://muni.cz/people/4189?lang=en

Abstract. The paper describes the process of building the Czech Digital Mathematics Library (DML-CZ) upon DSpace System. At first, the DML-CZ will be briefly introduced. Then we will describe DSpace system and its architecture together with Manakin – a system for building user interface above DSpace. The first technical part of the paper will be about mapping DML-CZ structure onto DSpace structures and about our importing tools and the way of managing requested features which are not supported by the default version of DSpace. The second technical part of the paper is about building a web user interface using Manakin – implementing new functionality and creating a new design for DML-CZ.

1 About the DML-CZ

The aim of the Czech Digital Mathematics Library (DML-CZ) project[1] is to digitize and present the relevant mathematical literature which has been published throughout history in the Czech lands [1, 2].

The complete workflow of the DML-CZ project involves several steps:

- choosing the mathematical literature,
- digitization (scanning, enhancing scanned images, OCR, ...) [3–5],
- getting[2] metadata,
- integration of the scanned data and metadata (Metadata editor [8]),
- presenting the results to the end users.

This paper describes how to present the results to the end users – how to build a mathematics digital library. There are at least two options: to develop completely a new digital library system, or to use and customize some of the existing digital library systems.

The most important advantage of building a new system is that you can develop it exactly according to your needs. However, the development consumes a lot of time. The other option (using some existing systems) consumes less

[1] Project 1ET200190513 – funded by the Academy of Sciences of the Czech Republic Programme "Information Society" (National Research Programme, 2005-2009)

[2] From referential databases ZentrallBlath [6] and Mathematical Reviews [7].

time, brings many interesting features and the future development of the system is guaranteed. The most significant disadvantage of the latter is that the system does not necessarily comply to your needs and has to be modified.

We have chosen the second option – to customize DSpace system. DSpace has many useful functions and features already included and supports many widely-used standards. Also the DSpace's data structures are suitable for the DML-CZ.

DSpace provides the tool Manakin for building custom user interfaces. Using Manakin allows to customize the system without doing any modifications in DSpace core – Manakin is independent on DSpace so migrating to a higher version of DSpace is relatively easy. Manakin is one of the most important reasons which led us to choose DSpace.

The next reasons for DSpace is that we are quite familiar and experienced with DSpace and that we have wanted to test DSpace and (especially) Manakin capabilities on a large project.

Next, we will introduce DSpace and Manakin and describe the most important and most interesting steps in the process of building DML-CZ using DSpace. The final DML-CZ web can be found at `http://dml.cz/`.

2 A Gentle Introduction to DSpace and Manakin

DSpace [9] is a free open source software for building institutional repositories and digital libraries. Nowadays, the development of the system runs worldwide, however, the main developers are supported by the Massachusetts Institute of Technology and Hewlett Packard. DSpace core (repository) is built on Java platform and PostgreSQL or Oracle database engines. There are two user web interfaces (UI) available within DSpace: Java Server Pages (JSP) UI and Manakin XML UI.

2.1 DSpace Features

DSpace can store various types of data. But some types of data are supported better (text documents, images) than the others (streaming video). Besides storing data, DSpace provides many useful functions and features. DSpace uses Apache Lucene as a search engine which allows to search in metadata and also fulltext documents (there is support for Adobe PDF and Microsoft DOC formats). DSpace also provides various browsing options (by structure, by titles, by authors, by date). All the digital objects stored in DSpace have their own persistent identifier based on the Handle.net [10] system. There is also OAI-PMH [11] server included and OpenURL [12] supported. As a metadata schema Dublin Core is used [13] (but different schemas can be added).

DSpace has its own system of authentication (which can be simply replaced by LDAP or any custom authentication system) and authorization. There is also adequate support for ingesting new data together with the sophisticated ingesting workflow system (dedicated users can accept/reject user submissions

before they are made public). However, we do not use the features described in this paragraph for DML-CZ currently (there is no possibility to ingest by hand and also there is no authenticated area in DML-CZ).

2.2 DSpace Data Structure

The main building stone of DSpace is a digital object – in DSpace terminology called Item. Item consists of metadata and data. Data is represented by files called Bitstreams which are compounded in named Bundles. Bundles are used to classify/separate Bitstreams according to their type and way of use.

Items are stored in Collections. Each item must be stored at least in one Collection. Above the Collections there is a hierarchical tree structure of Communities (Communities under a certain Community are called Subcommunities). Every Collection must be stored under just one Community. Communities are intended to represent a community at an institution (for example a faculty or a department) but it is usual that Communities are used differently and for diverse purposes.

Communities, Collections and Items are identified by a unique identifier (which can be, and for DML-CZ also is, Handle.net identifier). The URL for a certain Community, Collection or Item contains this identifier.

2.3 Manakin

Manakin is a brand new system for building custom user interfaces above the DSpace core. Compared to the default JSP web user interface, Manakin has a new approach to creating user interface. JSP is closely tightened to DSpace core (to servlets) and the customizing leads to modifying the servlets Java code directly. This causes difficulties when upgrading to a higher version of DSpace (because it is necessary to do complete re-engineering of servlets code to implement custom changes from the previous version). However, Manakin is strictly separated from the DSpace core, avoids the difficulties of JSP UI and has new features. To sum up, Manakin separates the logic and the presentation layers.

Manakin generates a user interface in two steps. In the first step there are Java classes called Aspects. Every Aspect is, in fact, a connector to DSpace core code and prepares data for the next step. Aspects get necessary information from the DSpace core using the DSpace API and transform it into an XML document called DRI document (**D**igital **R**epository **I**nterface). Manakin also provides methods for easily creating valid DRI documents. Aspects are run in chain according to web page they are preparing information for. So every Aspect can add its piece of information into DRI document. For example there are Aspects for generating the head of the page, footer, menu, navigation etc. Chaining the Aspects is done by certain rules and can be affected by developer needs.

In the DRI document there is all necessary information for the second step – transforming a DRI document via XSLT into a final HTML document (with CSS style attached). Manakin provides default templates for transforming a DRI document. A developer can easily redesign every template and create new content

of the final web page. We can set which XSLT style (and also CSS style) will be used for the whole DSpace, for certain collection or community. Every part of the DSpace can have different design then. The designs are called Themes.

To sum up the process of creating a web page with Manakin – a user requests a certain URL, Manakin gets the URL and runs a chain of Aspects according to the rules for this URL. After the Aspects are finished Manakin puts the generated DRI document for XSL transformation (according to a defined Theme). XSLT then produces an HTML document which is sent to a web browser (to a user).

3 Mapping DML-CZ Structures to DSpace

There are three types of data in DML-CZ: journals, monographs and proceedings. Each of them follows their own logic structure that has to be mapped to DSpace structures – Communities, Collections and Items – before we can start the import.

The classic journal structure is *Journal* → *Volume* → *Issue* → *Article*. In DSpace terminology this corresponds to *Community* → *Community* → *Collection* → *Item*. Communities, Collections and Items have certain attributes and behavior which do not always suit our needs. These, however, can be changed with help of Manakin (see Sect. 6).

Usual monograph structure is *Monograph collection* → *Monograph* → *Chapter*. This is equal to *Community* → *Collection* → *Item* in DSpace.

For proceedings there is *Proceeding series* → *Proceeding issue* → *Article* structure. This is mapped to *Community* → *Collection* → *Item*.

As you can see, journals, monographs and proceedings are Communities – there is no information about the Community's type (whether it is a journal, monograph or proceeding). This kind of information is not relevant for DSpace core and we sort the Communities according to their types at a presentation level (via XSLT, see Sect. 6.3).

DSpace uses Dublin Core as the default metadata schema and a basic metadata field set (includes author, title, date issued, etc.). The metadata for DML-CZ is a bit richer. To fulfill our needs we have added several new fields to the default schema (e.g. the identifiers to Zentralblatt Math and Mathematical Reviews, fields to store transliterated titles, references, ...).

4 Importing DML-CZ into DSpace

The whole process of preparing metadata and data before their publication consists of several steps (see Sect. 1). In the paper, we are interested only in the last one of them when everything is nearly prepared by Metadata editor for publishing on the web. The philosophy we have chosen is to let all the tools, such as the Metadata editor, to do as much work as possible so that the import consumes less time. The importing time is not so interesting for the initial import but it is really important in case we want to do periodical updating.

DSpace has its own importing tool which is obviously not compatible with the Metadata editor data format. We have found the DSpace import tool a little bit clumsy because it only allows importing Items into a certain Collection, does not support managing Bundles nor creating Communities and Collections. Although the most processing is done before the import, we need to do some work during the import too. That is why we have decided to create our own importing tools (using Java and DSpace API).

Because our import tools work similarly for journals, monographs and proceedings, next we will describe them only for journals.

4.1 Main Import

There are several scripts which do the import into DSpace, however, only one of them can be considered the most important. This script is written in Java and is designed to import a single Item into a given journal (and is called – according to the data directory structures created by the Metadata editor – by a special unix shell script for every single article in the given journal).

The Java import script checks at first if there is the volume and the issue already created and, if not, then it creates them. Now the import script knows the Collection that represents the issue and can load (or if the article already exists, update) the article. This includes processing the XML file with metadata and making necessary work on them – loading author's alternative names (from Metadata editor database) and normalizing titles. Normalizing titles means to cut off diacritics according to the title's language ("Analýza" → "Analyza")[3] and to do transliteration for German titles ("Lösung" → "Loesung"). This work is made for indexing purposes for the search engine.

The metadata should be filled as best as possible – everything in the Dublin Core metadata record is then used by the OAI server.

After the metadata are loaded, the import script processes the references – there is a special Bundle created where the XML file with references is loaded (for later processing in Manakin) and also each reference is stored in the Dublin Core metadata field (for OAI).

At the end of the import process the thumbnail of the first page of the article and the article itself (in PDF) are ingested.

4.2 Identifiers

DSpace behaves as an identifier generator – whenever a new object is ingested, DSpace generates a unique identifier. However, we need to know the identifier before the object is ingested. This is because the stored Item is referenced by its identifier in a URL and this URL should be included in the pregenerated article PDF file (which is loaded during the import). That is why we have created our own independent unique id generator (written in Perl). Before the import script creates/ingests a new object (identified in the way the Metadata editor uses)

[3] ICU4J [14] library is used.

this id generator is called to get a new unique identifier. Once the identifier is generated it is persistent – id generating is an idempotent operation – so the generator can be called repeatly (e.g. when updating).

DSpace supports assigning pregenerated identifiers for Items only. To be consistent we have settled that the identifier will be assigned by our id generator also for Communities and Collections. This decision requires the adding of support for this into the DSpace core. And this is the only change to DSpace core we have made.

5 Added Features

There are some features requested by mathematicians which DSpace does not provide. These features include support for links between articles, support for MSC codes and articles count by authors and MSC codes.

The links between articles are provided (in XML) by Metadata Editor for every article and are simply loaded into the article's metadata record.

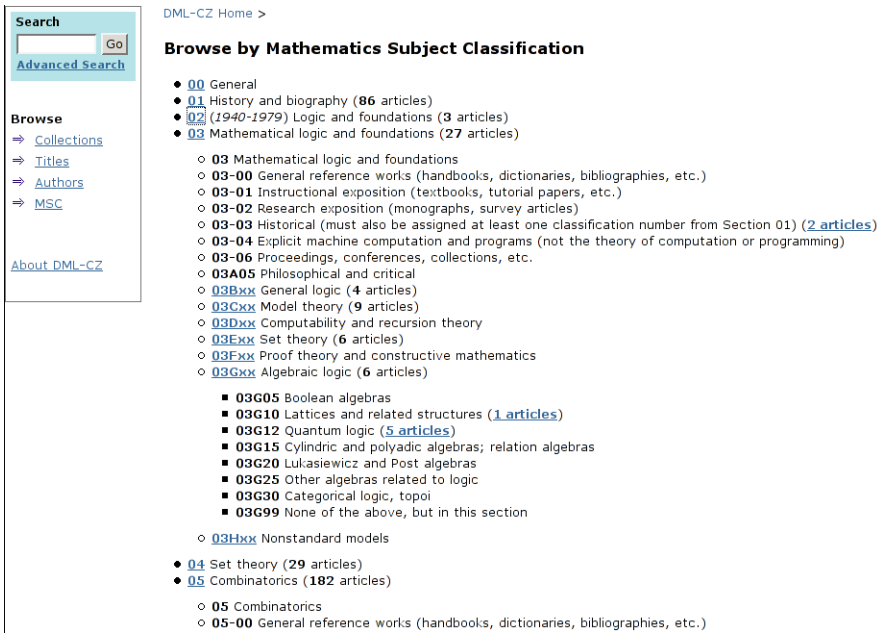

Fig. 1. MSC Codes Tree

Because DSpace supports listing Items by subject we store MSC codes in Item's metadata record in the field "subject". We only need to build up the MSC codes tree to provide comfortable navigation in MSC codes (see Fig. 1). The tree is stored in an extra database which is separated from the DSpace

database – because the best practice how to add new features is to change as little of DSpace code (or db schema) as possible. Except for the MSC codes tree itself (a code, its description and parent code) we also store how many articles the code has and how many articles are under the code's subtree. The counts of articles are generated in a batch mode by a special script.

The articles counts for authors are done the same way as MSC codes are. There is a table (in the same database as MSC) for authors where pairs of author and count are stored. The script that fills the table runs in a batch mode too. DSpace supports the context sensitive browsing – one can browse authors only in a certain journal or issue. Generating counts for all contexts would be really tough and that is why we have counts only for the general context (which is the whole DSpace) now.

6 User Interface

In this section we will describe how to use Manakin to present the imported articles to the end users. As we described in Sect. 2.3 Manakin works in two steps. If we want to implement a new feature, a new behavior or a design, we need to know what kind of Manakin tools would be most suitable. For example changing colors, icons, font sizes etc. can be done via CSS while adding navigation links has to be done at the Aspects level. The general instructions for working with Manakin are these:

- change the CSS style,
- if all information you need are in a DRI document then use XSLT,
- if there is not enough information in a DRI document then create Aspect (which adds the information to a DRI).

A lot of modifications of the user interface have been done. Almost all of them have been done on the same principles. Further, we will describe the most interesting and the most "tricky" modifications.

6.1 Journal's Homepage

There should be enough information for the comfortable navigation through journal on the journal's homepage. There should be listed all the volumes together with all issues within the volume (see Fig. 2). Manakin (default Aspects and templates) does not support this. Manakin only lists Subcommunities within Community. Journal is in fact the Community and so we can get only the volumes (Subcommunities) within the journal and not the issues (Collections) within the volumes. Next, we will describe how to achieve that.

First, we need to create a new Aspect that adds a list of all issues (Collections) to every volume. Our Aspect will go through the volumes and for each of them will call DSpace API method which returns the issues (Collections) within the volume. Then we can add these issues into the DRI document. Once the Aspect is created we have to put it on the right place into the Aspect's chain. Now there

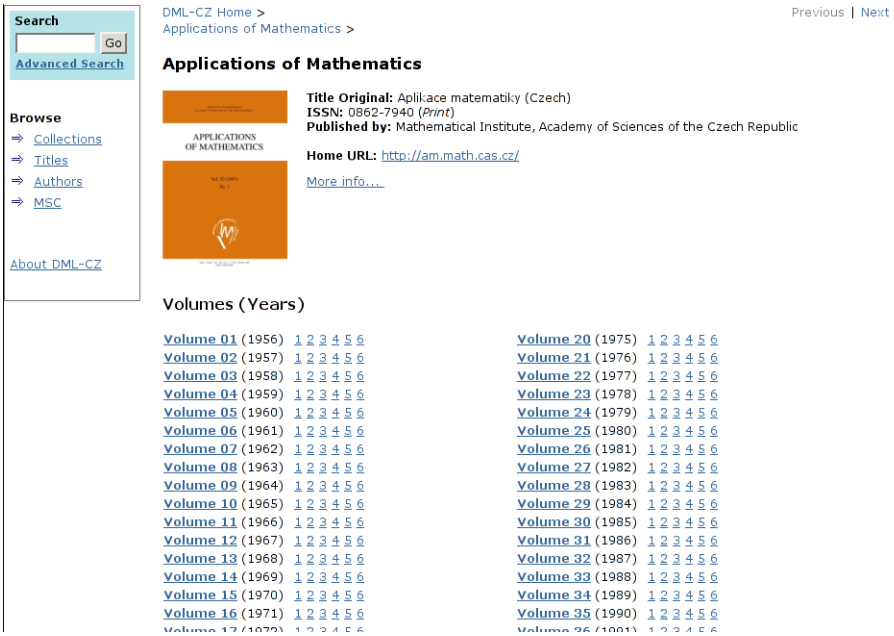

Fig. 2. Journal's Homepage

are all issues (within volumes) listed in the DRI document so we can modify an appropriate XSL template to transform this piece of information into the final HTML page.

6.2 Journals and Volumes

There is no difference between Communities in DSpace core. Both journals and volumes are Communities and we need to know which is which for presenting them to the end users. Because the journal's structure is known we can check "the level" of any Community. "The level" means how deep we are in the Communities tree structure and is provided in the DRI document. The "journal" Community is designed to be the top Community (1^{st} level) while the "volume" Community is on the 2^{nd} level. Redesigned XSL template for displaying Community can take "the level" and according to the information it can call appropriate XSL templates for displaying the journal or the volume.

6.3 Journals, Proceedings, Monographs

The decision if the Community is Journal, Monograph or Proceeding is also done during XSLT transformation. We set (by hand) three global variables called `journals`, `proceedings` and `monographs`. These variables hold identifiers of the Communities according to their types. Because XSL does not support complex

types, the identifiers are stored as a String type and separated by commas. Every template can access these variables and get (with help of XPath function `contains($identifier, String)`) the type of the Community and can do the appropriate transformation (or appropriate template call) then. The result might look like Fig. 3.

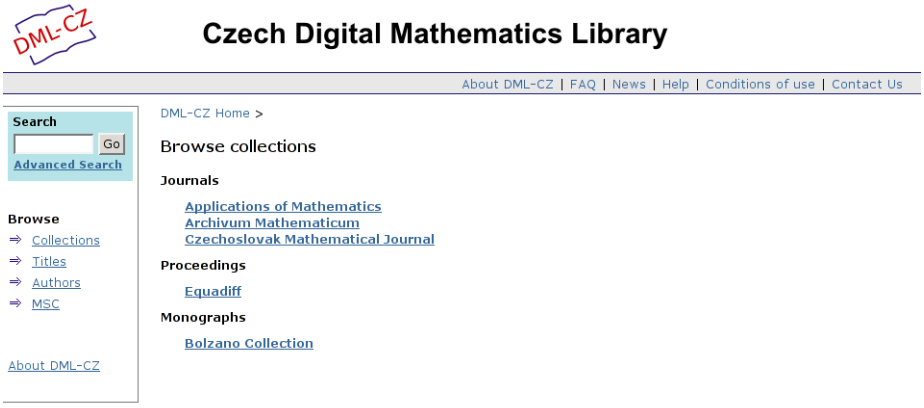

Fig. 3. Journals, proceedings and monographs

6.4 Items Counts by Authors and MSC Codes

The default DSpace does not know anything about counting items by authors or MSC codes. We have already populated the special database (see Sect. 5) with such information. We need to create a new Aspect that connects to this database, gets the information about counts and puts it into a DRI document. Once the information about Items count is in the DRI document, any XSL template can render it.

6.5 Summary

Most of the design and functions of DML-CZ web user interface is done by the set of XSL templates and Aspects similar to the ones described in the sections above. Except for the described there are more then 120 new or redesigned XSL templates and about six new Aspects. The most complex templates are especially for rendering an Item page, changing the search page and for the MSC Codes tree page.

7 Conclusion

We have found that DSpace and Manakin are the proper tools for building specialized digital libraries like the DML-CZ. Although there are some difficulties,

the advantages prevail. We would like to implement new features like usage statistics, users discussions and subscriptions and to use a DSpace authentication and authorization system. The development of DSpace and Manakin is still in progress so we can expect less difficulties and many new features in the future.

References

1. The Czech Digital Mathematics Library Project. http://project.dml.cz/
2. Bartošek, M., Lhoták, M., Rákosník, J., Sojka, P., Šárfy, M.: *DML-CZ: The Objectives and the First Steps*. In Rocha, E.M., ed.: CMDE 2006: Communicating Mathematics in the Digital Era. A.K. Peters, MA, USA (2008) 69-79
3. Sojka, P.: *From Scanned Image to Knowledge Sharing*. In Tochtermann, K., Maurer, H., eds.: Proceedings of I-KNOW '05: Fifth International Conference on Knowledge Management, Graz, Austria, Know-Center in coop. with Graz Uni, Joanneum Research and Springer Pub. Co. (2005) 664-672
4. Sojka, P., Panák, R., Mudrák, T.: *Optical Character Recognition of Mathematical Texts in the DML-CZ Project*. Technical report (2006) presented at CMDE 2006 conference in Aveiro, Portugal.
5. Sojka, P., Řehůřek, R.: *Classification of Multilingual Mathematical Papers in DML-CZ*. In: Proceedings of Recent Advances in Slavonic Natural Language Processing-RASLAN 2007, Karlova Studnka, Czech Republic, Masaryk University, Brno (2007) 89-96
6. Zentralblatt MATH. http://www.zentralblatt-math.org/zmath/en/
7. MathSciNet Mathematical Reviews. http://www.ams.org/mathscinet/
8. Bartošek M., Kovář P., Šárfy M.: *DML-CZ Metadata Editor: Content Creation System for Digital Libraries*. (Submitted to the workshop "Towards Digital Mathematics Library 2008").
9. DSpace. http://dspace.org/
10. Handle.net System. http://handle.net/
11. Open Archives Initiative – Protocol for Metadata Harvesting (OAI-PMH). http://http://www.openarchives.org/pmh/
12. OpenURL. http://en.wikipedia.org/wiki/OpenURL
13. Dublin Core. http://dublincore.org/
14. International Components for Unicode. http://www.icu-project.org/