Building WebPAC for Faculty of Philosophy Libraries - experiences and lessons learned

Abstract

In this article, we'll try to answer the question "how" we implemented our WebPAC, what were our goals and how they were changing as we were progressing and adjusting them to fluctuating conditions. We will describe our environment, consisting of 19 different libraries with an undefined organisational structure, problems with floor space, insufficient communications between library staff and the board, and many other problems.
Another important question we want to answer is why we decided to solve our problem going step by step and using the existing Open Source solutions, no matter how complete or inadequate they were, instead of waiting for a single "perfect" solution that would solve all our problems in one integrated package.
Finally, we will share with you what we have learned in this process and how our new knowledge affected creation of new requirements, as well as our future plans.

Short biography of authors

Marijana Glavica is employed at Faculty of Philosophy in Zagreb, in the Department of Psychology Library. Currently she is involved in providing on-line resources to users of all Faculty Libraries. She also maintains the existing electronic library catalogue, and coordinates publishing of the catalogue and library related web pages on Internet.

Dobrica Pavlinusic, self-proclaimed Unix addict and Internet consultant, graduated at The University of Zagreb Faculty of Organization and Informatics Varazdin from where he received B.Sc. in Information Science.
His fields of interest include Internet technologies, Unix, Open Source movement, Free software and Linux.
Member of supervisory board of HULK - Croatian Linux User Group and member of executive committee of HrOpen - Croatian Open Systems Users Forum.
Currently he is working for the largest pharmaceutical company in this part of Europe, where he performs system administration of Unix servers and implements Internet technologies based on Open Source.

Introduction

Today, there is no special need to prove that IT technology has to be introduced into standard library operations. IT is simply irreplaceable as a support to librarians in performing a number of tasks necessary for normal library operations, as well as to users in search of needed information. IT revolutionised libraries, changing their role in society and turning them into information centres.

Electronic services are no longer just additional services which a library can offer to the user. Rather, if a library, particularly one which provides support for academic and research purposes, cannot offer these services to the user, it cannot be considered as an institution which fulfils its function.

We will try to describe how and why we started developing our own system through the integration of components – both existing and those that are yet to be built, and why we believe that we choose a good approach.

This is also a story about how we realised what is an information system.

Here, we would like to show you how we built WebPAC, and to share with you the experiences we gained in process. The goal of this text is to explore these relations and to focus on interaction between human, organisational and technological components of the system which should be a support to people and their community.

In the first part, we will talk about:

- critical prerequisites necessary to start using IT
- our working environment
- decision to make online catalogue – and how and why it was reached
- approach to the problem and why we decided to use Open Source technologies
- what we have learned and what are our plans for the future

The second part will present the technology used in creating online catalogue for the Faculty of Philosophy Libraries, also known as WebPAC. Those who are more interested in the practical part might consider skipping straight to the chapter "WebPAC as Technology Challenge".

Brief History of How IT Entered the Faculty of Philosophy Libraries[1]

The following story describes the stages through which FF Libraries went in their efforts to introduce IT technology in their environment. It was necessary to pass through a number of stages in order to demonstrate to the end users that the new technology will bring them substantial benefits. We believe that it is imperative to show this development as it reflects on the momentum we are using now and to show that this projects is resting on old but solid foundations. Therefore, we won't skip any of the steps we walked in fulfilling the prerequisites necessary for the current project.

The entire nineties were marked with waiting – everyone was waiting for a change in informatisation, the need for which was discussed more and more often. However, some people did not accept passive roles and started using IT on their own. At the beginning of nineties, prof. Aleksandra Horvat from Faculty of Philosophy Chair of Librarianship, and Marija Laszlo developed an application of a free UNESCO program CDS/ISIS for teaching cataloguisation, in order to demonstrate modern librarian tools to their students. Prof. Horvat, who was very proficient with required standards (UNIMARC) and cataloguisation rules, and Marija Laszlo, who was well versed in informatisation, combined their knowledge to create a product which was used not only for teaching but also in library practice. Some librarians started creating catalogues of their materials using CDS/ISIS as early as 1991., and authors of the application provided support and education for them. This proved to be a good choice, since this application of CDS/ISIS respected baseline standards.

The first shipment of computers destined for libraries arrived in 1995. These computers came through a donation without a particular purpose, and they were more of a symbol of things to come than tools for serious use. There was no organised education at the FF level, nor was there any thought on the organised informatisation of the libraries.
At that time, Internet access was already present in Croatia and FF has been connected to CARNet. The Network was slow and unreliable, but it was enough to start demonstrating its abilities to some individuals. The Library Department started considering its use in libraries.

A new shippment of computers arrived in 1999. This time it was followed with courses in Windows and Word basics, and the author of CDS/ISIS application was engaged to install the application on all computers, to teach librarians how to use it, to maintain it, and to develop improvements. This period brought great expectations, but also great disappointments. The co-operation was smaller than today, but the most important consequence turned out to be that most libraries started making records in the same format, meaning that it will be easy to eventually unify them.

The need to use the information technology in libraries also increased because of the increased popularity of Internet. FF was building a communication infrastructure (which has become modern and powerful), thus creating prerequisites to connect the libraries. Some librarians saw the importance of Internet for libraries and started gathering and using electronic sources of information, advising their users how to use them, and Web publishing the key information together with links

---

1 Although the title states "brief", this overview turned out to be rather long, perhaps even boring. However, both adjectives reflect the period perfectly – it lasted too long and it was boring.

to useful sources. The librarians felt a strong need to educate themselves and to explore their new roles. This was the moment when more serious projects became feasible.

In May 2001, we came in charge of subsystem for humanities within the structure of the project Croatian System of Scientific Information initiated and funded by Croatian Ministry of Science and Technology[2]. We will call that project by it's Croatian acronym SZI. SZI brought first strong encouragement – we received another shipment of computers for all libraries, as well as the money to engage people to enter data (these were picked among librarian students). Most of the money from the project was used for entering data in order to create digital record of our fund which will, finally, be shown on Internet. Through this project and thanks to the FF support, we established foundation for permanent expert education of librarians. SZI provided necessary funding, but WebPAC didn't result as a direct consequence of these assets.

September 2002 – We publicly present WebPAC – an online library catalogue, a prototype solution[3] which demonstrates our position on informatisation problem.

At the moment, we are still entering data in our database, and the rate of this activity should be increased. There is also a number of unsolved problems, and we are deciding about our future steps.

Critical prerequisites for successful acceptance of IT

We were aware that we badly needed IT, and we were helplessly listening to others and wondering "why are we not getting it?" We were mostly listening to shallow promises and waiting for solutions, but in the meantime we were also learning about usage of computers and becoming more and more aware of its possibilities.

People can accept IT only if they are motivated, informed in its use, and ready to cooperate, if they have well equipped information and communication support and want to understand their work and realise that IT is offering them something useful. Although we were listening to complaints that there was nothing going on in the FF Libraries regarding the IT improvements and that part-time efforts resulted in partial results, the whole period created a common ground for future. Today, most of the librarians are using computers, they can find their way in word processing programs, they know how to search catalogues using computers, how to use e-mail, search other libraries using on-line catalogues and acquire books over the Internet using on-line services. Few months before entering the SZI project, we started our internal mailing list at groups.yahoo.com which allowed more open communication about problems. Finally, there were the new IT support department and new network, both of which significantly increasing work comfort.

After the arrival of the shipment from the SZI project, we became, for the first time, motivated to do something. Finally, we were able to communicate and emphasize the importance of our project, and we received strong support from FF. We began to believe that we were the ones who needed, and more importantly could, provide initiative and vision about necessary services.

---

[2] SZI goals are: to build the information system for scientists and turn libraries in information centers with services to satisfy scientific and academic activities. You can find more information (in Croatian) on the SZI website at http://www.szi.hr/

[3] In the technology chapter, we will explain why we consider the current solution to be a prototype.

SZI joined us. It also partially substituted the non-existing central service; through it, we were able to realise a number of important jobs for all libraries. SZI also provided a cooperative, caring environment where issues were solved through dialogue instead of hierarchical directives.

People and Environment

We learned that the development of IT solutions was impossible without knowing the orgranisation and people who will use that solution. We reasoned: "the purpose of the information system is to support people in a community." Our community was growing and became aware of the fact that there it needed to learn much. As we were progressing in our efforts to use IT in our environment, we had more and more problems that needed dealing with. We started to develop new services while at the same time we still worked in bad conditions and with space problems[4]. We had more and more users coming and the libraries had to function simultaneously in both ways – the old way, unadapted to the new environment, and the new one which librarians were just learning. People were and still are working under a lot of stress because of the changing environment. The librarians are affected with IT in more than one way, because their job consists of dealing with information. IT brought increase in the information production and new kinds of media for storing that information[5] and librarians are expected to deal with it.

The librarians used to be concerned only with their own library, doing all necessary jobs (acquisition, cataloguing, description, communication with users etc.) But all the libraries belong to the same institution[6], and they share similar needs and problems. There was a growing need for collaboration and communication. We knew that we had to share our tasks, but we didn't know exactly what these tasks were and how to distribute them.

Also, we had no experience in joint projects and poor overall experience in working on projects in general, since our jobs were to support someone else's projects. Planning was also incidental, short-term and inconsistent and it never involved all libraries.

The development of the IT solution alone would take around one month. However, we spent the next six months implementing it, because we were questioning and solving problems in each step of the process.

How Did We Decide to Implement Online Catalogue aka WebPAC?

---

[4] Librarians might comment: "What would be the use of IT in helping me making my collection visible to users, if I'm unable to find the requested item because books and other materials are scattered all over the place, and a lot of materials are actually inaccessible because the teaching staff is keeping them in their rooms."

[5] Ten years ago, in our case, librarians had only paper materials. Now they have to deal with on-line publications, search engines, multimedia CDs along with paper.

[6] There are 19 libraries on FF, including a central reading room which is treated as a separate library. These libraries belong to individual departments. There are no common procedures for dealing with issues that are of mutual or shared interest. Libraries depend on their departments and on FF itself. Individual librarians have fixed annual amounts for spending on books and nothing else.

Libraries always communicated with users through catalogues. A library catalogue provides a user with information about library collection and availability. So, it is not strange that libraries want to increase their presence on Internet by publishing library catalogues.

We already had records in CDS/SISI and we realized that we had to and could put them on Internet rather quickly. We knew that there where free solutions available, but we didn't know how to implement them.

One can wonder why we chose to implement WebPAC first, and not to create authority files or to make more detailed records. The answer is simple. We wanted to provide users with information as soon as possible. Even if a record was not as detailed as library standards suggest, it would still give the user fast and complete answer to a simple question. We were fed up with repeating the same answers to the same questions for a hundredth time, and we were tired with managing our own lists. "Do you have this book? Can I borrow it? Which books are recommended by professor X, or for this subject? Which books do you have from this author? Do you have any books on this topic? Where can I find this article?"

We couldn't start with some of the other jobs earlier because there was virtually no interest for them. However, since communications improved as a result of SZI project, the librarians organised themselves and formed small task-oriented teams. We where hoping that we'll be able to solve problems, which we cannot solve alone, in teams. Our goal is to lend books, so keeping our books locked and uncataloguised is counterproductive. Because of that, we were aware how precious is the information coming from WebPAC which helps users in navigating through library materials.

Another good reason to take WebPAC implementation as the first step was that there were people who wanted to do it[7].

As we progressed through our journey of the WebPAC implementation, we were glader and glader that we took this approach. In the process, we learned how other elements of our system should interoperate and we are now ready to move on.

Considerations

For start, we at least knew what we didn't want. Right in the beginning, when we decided we wanted WebPAC as soon as possible, we rejected one of the possible approaches to informatisation of a number of small libraries belonging to a large institution – waiting for the large and slow institution to find gather all data, analyze requests in detail, consider incompatibilities, weigh demands and possibilities and put them in agreement with goals, fulfill the lists of functional requests, discuss every detail several times, wait for a few deadlines to pass, make public tender, choose one of the applicants, give a lot of money to the chosen one and hope that he will take care of the libraries. This might have been a good solution for most of the libraries, because they wouldn't have to worry about a job they didn't know anything about. We wouldn't have had anything against it either, provided we received a truly good and flexible software together with support and without license restrictions that could make us dependent on one supplier. No library application is perfect (just like any other software), which is normal

[7] Although this seems simple and logical, there are many projects which failed because of insufficient motivation among their developers.

because it constantly follows demands of humans and the environment, both of which are constantly changing their requests and demands.

Even if a perfect solution like that existed and we were prepared to pay for it, inexperienced as we were, at that moment we would have been unable to choose what we really needed.

Now we had to consider the other two approaches we had on our minds. If we didn't have money for an integrated solution, we could just buy a complete WebPAC solution. However, there was no satisfactory localized product, products were too expensive and demanded adaptation to our needs. That would have created additional problems with localization, education, data conversion and maintenance for us.

So if we can't get complete solution, why not just WebPAC for ISIS? We had exactly 12,000 Kuna (or €1,600), and that happened to be exactly the price for one local software solution which worked with data from CDS/ISIS. The whole deal seemed a little bit too expensive, especially considering that the price didn't include any customisations which would have had to be made by the original author at an extra charge. We found several free foreign solutions developed by CDS/ISIS community, but those solutions where hard to implement and proprietary[8] so we didn't think it would be smart to work on their improvement.

We choose the third approach: a combination between the existing Open Source solutions and development of custom made parts for our specific needs. Using this approach allowed us to work on our problems one at a time, without limiting future possibilities and fixing ourselves to firm requirements. The benefits also come from the fact that we were learning through constant changes, experimenting and modifying our requirements in consideration of new experiences and it was all rather easy to do – as long as we continued using open standards.

Open Source in Libraries

Soon we discovered a growing community[9] of people advocating Open Source systems in libraries, and found our colleagues who were thinking and exploring in the same way as we were.

We saw that we, librarians, and Open Source community shared similar ideals and principles. We respected open standards, knowledge sharing, and informal

---

[8] CDS/ISIS is not an Open Source solution. It is just free (as beer). To base something on the code which is not free (in terms of a speech) has its downsides. For more information, see http://www.gnu.org/philosophy/shouldbefree.html

[9] Community with its own web sites, with reference to articles and books about Open Source in libraries (you can find a good review of main ideas in Special Issue of the journal Information Technology and Libraries with topic Open Source Software).

development processes. The fact that it provided us with full usage of existing resources[10] was also a benefit.

We were fortunate to be involved in process of building and implementing an application for our needs, and we where finally invited to express our needs and given the time to explore them.

We were also sure that we wanted Open Source because we worked and lived in the environment which supported learning and exploration. Our work is based on cooperation, rather than on competition.

[10] We were promised that all kinds of records, no matter their format, would be usable as long as data is publicly available and machine readable (we had lists of journals written in MS Word and user address books in MS Excel, e.g.)

WebPAC as Technological Challenge

For those of you who jumped straight to this chapter, we would you like to know that there was a moment earlier on when we had to consider which technologies we would use for our WebPAC. Since we already had CDS/ISIS for cataloguing (with an on-going effort to enter even more data in it), we needed some way to connect our new system to the existing CDS/ISIS. Fortunately, as we were to find out later on, we were not the only ones who did not know where to look.

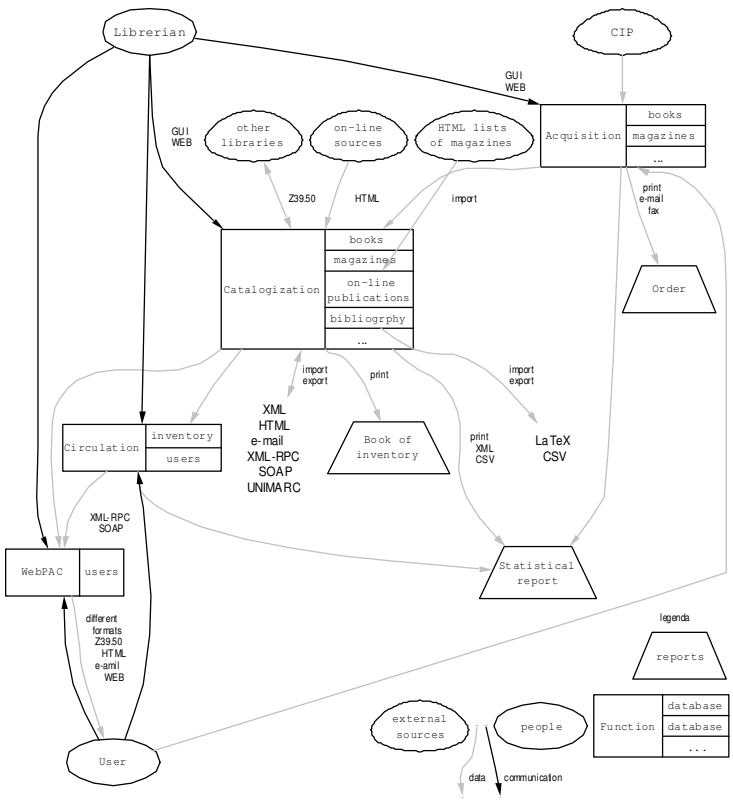
We wanted our solution to be based on Open Source because of numerous reasons which are best summarized[11] in this definition from www.opensource.org :

> "Open source promotes software reliability and quality by supporting independent peer review and rapid evolution of source code. To be certified as open source, the license of a program must guarantee the right to read, redistribute, modify, and use it freely."

Component Architecture

It's not easy to choose the correct solution when you are trying to implement a library IT solution for the first time. We anticipated that the best solution would be *component-based*, because that would allow us to change any component if needed (or replace any component with a current solution, e.g. like using Excel for library acquisitions).

Since an implementation based on components tends to have quite a significant overhead (development time, code size and complexity) compared to a monolith[12] implementation, we had to be sure that it was the right approach to our problem. Our reasoning for the component approach was simple: we wanted flexibility to change or improve any given component (in our case, search engine and circulation) without needing to change any other element in our system.



The same component-based approach, together

[11] Parties interested in Open Source are kindly requested to examine some of the resources available about that topic, some of which will be referenced throughout this paper.

[12] Sometimes monolith implementations are called *one-shot*. Some of us know why: you often end up being shot in the leg.

with definitions of available formats and procedures for interchange between components, is recommended by one of the teams *for NISKA project* which deals with networking of all libraries in Croatia. You can see all that on the picture attached.

Pack Your Components in a Box

When we became aware of all the advantages offered by the component approach, we had to choose which components to take. Since some of our components were fixed (we use the existing CDS/ISIS for cataloguing) and some were planned for later (like circulation), we opted for integration of individual parts as opposed to one solution that covers all functionality.

That's exactly the opposite from Koha project that is a full catalogue, OPAC, circulation and acquisitions system. We tested Koha (along with many other solutions which promised to fulfil our needs), but decided to standardize interfaces between components and pick the best component for each job. Or, indeed, write our own component if necessary.

This approach (starting with the existing CDS/ISIS for cataloguing and adding web OPAC to it) enabled us to progress in an incremental fashion, as opposed to a single gigantic release. So, after the planning we started to search for a perfect search engine...

How to Choose Search Engine

First problem we encountered was to decide which indexer to choose for our indexing. We had some previous experience[13] which taught us one thing: we didn't want to implement a search engine that would use a relational database management system (RDBMS). Basically, the reasons could be summed up like this: RDBMS is great for searching of structured data. However, if it is used for text, it's slow and hard both to use and to program.

On the other hand, we wanted to use an existing technology in order to avoid reinventing the wheel and to possibly improve some project to the benefit of the whole society (or at least Open Source and Library).

Sure, we where concerned about performance, but our main concern was human-computer-interaction (HCI) or Web interface. Fast search engines can't help a lot if your users can't find their ways. However, if user interface (UI) is too simple, not only would power-users feel left out, but it would also prevent making multi-criteria searches. At the beginning, we were considering if we should have a completely different user interface (e.g. normal Windows GUI) for librarians and Web interface for other users. But later we decided that it would suffice if our interface had the advanced search option.

As with all Open Source project quests, first we examined a great Open Source index called freashmet.net. This turned out to be a no-go, and for a while we where thinking about browsing through sourceforge.org (which is not really a good

---

[13] This is not the first text indexer implemented by Dobrica Pavlinušić. The first one was a search engine for Narodne novine with special support for various forms of words in Croatian language. Lession learned in that project helped in forming the decision that indexing software is much faster and better suted for searching of text than RDBMS (relational database management systems). For more inforation visit http://nn.rot13.org/.

way to find useful projects, because most projects on SourceForge are in early stages of development).

However, a few good words entered in www.google.com revealed site that we where looking for: www.oss4lib.org (open source systems for libraries).

> "… [oss4lib] mission is to cultivate the collaborative power of open source software engineering to build better and free systems for use in libraries. Toward this end, we maintain a listing of free software and systems designed for libraries (the physical, books-on-shelves kind), and we track news about project updates or related issues of interest.
>
> oss4lib started at the Yale Medical Library in early February 1999 thinking there were probably other folks like us out there who might be working on free library software or looking for same. Our reasons for wanting to see Open Source take off in the library software arena are straightforward, fairly typical…" (from oss4lib website, about)

There, under the section projects, Dan Chudnov (thanks, Dan!) had a list of 74 projects that were somewhat related to libraries. After a lot of surfing we were getting a better picture about what is out there, but we still didn't have any real decision or hint about the direction we should take.

However, it seems that we were not alone in our search for text indexers and free or Open Source library programs. Eric Lease Morgan wrote a great article entitled "Comparing Open Source Indexers", the information from which served as a starting point in our assessment of various search engines.

Eric covered the following Open Source indexers:

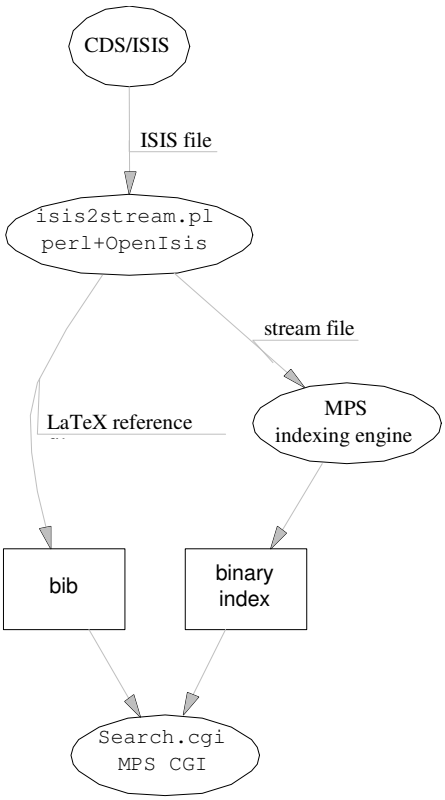
1. freeWAIS-sf
2. Harvest
3. Ht://Dig
4. Isite/Isearch
5. MPS
6. SWISH
7. WebGlimpse
8. Yaz/Zebra

Since a detailed description of each indexer is not the scope of this document, we will just emphasize that MPS and SWISH are the two indexers which received top marks for both performance and features. While each of them seemed like a good choice for our project, we selected MPS because it comes with a CGI Web interface which was already prepared for library use (although that perl CGI script will prove to be a bad choice, as it will be soon revealed).

MPS Indexer

So, now we had an indexer that would solve all our problems. Or did we? One of the frequent problems with Open Source software is the lack of documentation. Let's face it: programmers don't like documentation. They are not librarians. However, MPS is not a free (as in not-paying, volunteer effort) Open Source effort. It's a commercial product by *FS Consulting, Inc*, a consulting firm specializing in the development, implementation and installation of information retrieval systems solutions. So, the lack of documentation wasn't a mere laziness. It was a deliberate

decision made by *FS Consulting*. After a lot of *greping*[14] and moving things around[15], we managed to install it anyway. Installation of its CGI script was a challenge of its own: the CGI script never returned a meaningful error message, and most of the time it was just producing "500 Internal Server Error".



However, after the installation and configuration, example databases (thanks God for them!) looked as good as the on-line demo. So, we where pleased with the interface (which reminded me very much of ProQuest UI with its ability to save user searches for later) and proceeded towards inserting our data from ISIS. However, that's a bit too fast. Let's first examine OpenIsis.

OpenIsis

OpenIsis is a project located at OpenIsis.org. As authors say on the site:

> "...a software to access isis databases [...which...] is and will be freely available for everybody with full source code, no fee, no restrictions."

Since we wanted to implement our WebPAC in languages which where familiar to us (which include C, perl and php) OpenIsis fit in nicely (with it's perl bindings). It was also fortunate that it already came with perl support (otherwise we would have to use SWIG to provide functionality of a C library in Perl). Later on in the project, we realized that using first versions of Open Source software isn't always good: we had to change most of the perl code with regard to OpenIsis after one major revision. However, functionality that was added with that revision was well worth the change.

Writing perl code…

Perl, as it is often called *Pathologically Eclectic Rubbish Lister*, proved to be a perfect middleware between CDS/ISIS on one side and search engine on the other, as can be seen on picture.

First thing that we had to do was to create a perl script (called isis2stream.pl) that would produce a stream file (in format which is expected by the MPS indexing engine) and a LaTeX reference file called bib file on the picture. The stream file is basically a list of words that can be searched through, while the bib file serves for displaying of additional information about the book. That design seemed reasonable, so we accepted it. It had an additional benefit for us: we could remove

---

[14] usage of grep command in Unix shell. It allows user to see all occurrence of a given word through all files in a directory

[15] for some strange reason, it is easiest (or possible, as opposed to extremely hard) to install MPS in the directory /local (which is very non-standard)

all Croatian special characters from the stream file: that way, users could enter either Croatian special characters (čšćđž) or their equivalents without carrons (cscdz).

Next thing we had to do was to write the user management functions. MPS CGI interface supported saving of user searches and organizing them in folders. However, in order to do that, it used HTTP authentification. That involved .htaccess files (together with .htusers file) which had to be modified by the administrator. Since we wanted to run a non-restricted search engine we wanted solution which would enable users to open their own accounts as needed. For that we used the perl module HTTPD::UserAdmin which is available on CPAN[16]. Using CPAN module enabled a quick development and users got the ability to save searches in folders for later.

Downsides of MPS

It wasn't long before something struck us as odd: Search.cgi used just one perl module. And that module was MPS module, which is basically just an API to C functions provided by MPS. On the other hand, perl code turned out to be hard to modify. Often, the problem with perl is that code is written using the "there is more than one way to do it" method (which usually results in the code that is unreadable to the coder himself after 15 minutes). However, this code was clean and used strict directive (which forces all variables to be declared, enabling perl interpreter to check for them and programmer to catch errors in variable names).

After a while (and quite a bit of modifications in Search.cgi) it became obvious what the problem was: it was written as if it was a C code. That means it had no CPAN modules[17] but it had wired structure. At one moment, this lead to a sudden choice whether to drop everything and re-write the whole code (that decision was accepted later because of other positive side-effects) or to just lose more hair.

That wasn't the only thing wrong with Search.cgi: all html output was hidden somewhere in in-between code. That is an easy way to produce output, but using templates has so many advantages (for us, one of the most important ones was the ease of localization) that I won't even try to list them all here[18].

We redesigned UI a bit for our needs: the use of CSS and JavaScript enabled us to provide select all/select none functionality and nice buttons (as opposed to normal HTML form buttons).

---

[16] CPAN is Comprehensive Perl Archive Network – collection of perl modules available at www.cpan.org. Often, CPAN is cited as one of the largest advantages of perl programming language: you can find modules for almost anything: sorting through arbitrary alphabet, database independent modules (DBI), various web and GUI interfaces and so on…

[17] Yes, perl is famous for "there is more than one way to do it". However, you can have "too much of the good stuff" sometimes. This time it means that it's perfectly possible to write perl code as if it's C, but that makes some constructs hard to watch (C doesn't have unless or if at end of statement and other visually pleasing [for programmers at least] aspects of perl). It also makes code hard to modify by "average perl person" (author assumed in this case).

[18] But I could note some references here for those of you programmers, who still doubt that templates are "good thing to do". You can read http://html-template.sourceforge.net/article.html for gentle introduction or http://www.stonehenge.com/merlyn/LinuxMag/col38.html for real-life story.

In the end, we spent most of the time modifying format of the stream and bib files that were created using OpenIsis from CDS/ISIS database. We also stumbled upon some bugs that were related to the MPS indexer or search engine (we are still not sure which): one of them was particularly annoying, forcing us to add asterisk (*) at the end of each ISBN number so that we could search using ISBN as a criteria.

Free Speech Differs from Free Beer

Actually, I will talk here about freedom, not free beer. Since we opted to implement the first version of our search engine based on an Open Source product, we choose the MPS engine and web GUI. We did not have to pay any money in order to obtain MPS source code. It was free (as beer). However, since MPS is actually a commercial product and licensing terms are somewhat undefined (and so our right to use it without license is somewhat questionable), our plans are to replace the MPS indexing engine with SWISH-E, so that we can enjoy real freedom. Since most of the web CGI interface is already rewritten, the transition should not be as much trouble as it might seem at first.

SWISH-E

SWISH-E is a great indexer, best described on its web site:

> "Swish-e is not a "turn-key" indexing and searching solution. The Swish-e distribution contains most of the parts to create such a system, but you need to put the parts together as best meets your needs. You will need to configure Swish-e to index your documents, create an index by running Swish-e, and setup an interface such as a CGI script (a script is included). Swish uses helper programs to index documents of types that Swish-e cannot natively index[19]."

It's also interesting to note the long and interesting SWISH (and later SWISH-E) history:

> SWISH was created by Kevin Hughes to fill the need of the growing number of Web administrators on the Internet - many of the indexing systems were not well documented, were hard to use and install, and were too complex for their own good. The system was widely used for several years, long enough to collect some bug fixes and requests for enhancements.
>
> In Fall 1996, The Library of UC Berkeley received permission from Kevin Hughes to implement bug fixes and enhancements to the original binary. The result is Swish-enhanced or Swish-e, brought to you by the Swish-e Development Team." (from SWISH-E web site)

So, future versions of WebPAC will have a free (as in speech as well as in beer terms) indexing engine and web CGI interface.

Development tools

Since I don't like to fly without a security net, early in the process we choose to use CVS (concurrent versioning system) to store source code. Benefits are obvious:

---

[19] In our case, the perl script which will read ISIS data using OpenIsis and export XML for SWISH-E to process

we can always roll-back to any old version (hopefully working) of our code without any problem. On the other hand, CVS also enables us to have one central repository from which we can distribute source code to multiple locations (distribution of source code over Internet comes as a first idea, but it's actually even more useful to have two versions of the site: one for production (latest stable version) and the other one for development).

CVS will also allow us to review all changes in Source.cgi (MPS CGI script which displays results from the indexer) to make sure that we re-implemented the complete source code, making it free from any license restrictions (and still protected by GPL license).

Where Are We Today?

Since the publication of this paper is a moving target, we'll freeze and review our status as of August 2002. We are planning and developing the circulation module (written in perl, using PostgreSQL -- RDBMS isn't always a bad idea. It makes perfect sense to put data on library users, as well as data on library inventory, in a relational database) which will communicate with the rest of our system using XML-RPC calls.

We are also evaluating if a terminal interface (available via telnet on obsolete 286 PC's) and/or Windows GUI interface would be a good and useful thing to do. That decision (whatever it turns out to be) won't change the circulation module; it would just add another component that is user interface, which will communicate with circulation module using XML-RPC.

Next step will be to implement a single common configuration file, which should be easy changeable by librarians. That configuration file will describe conversations that take place from ISIS data to formats for indexing (stream file; also used to display search results) and full book details (bib file; used when user want to see all data about particular book – somewhat like display of records in CDS/ISIS).

Also, we are planning to replace the indexing engine with SWISH-E (as mentioned earlier), publish our project (which won't be clouded by some obscure license; we already have a website for at WebPAC.sourceforge.net) and continue integration into current and new systems in use at Faculty of Philosophy.

Conclusion with doubts

So far, this project was a load of fun, but we feel that we must emphasize some doubts that often follow projects like this one.

Doubt 1. S*oftware and hardware are most important elements of information system. It is important to invest in expensive equipment to make a start*[20].

The fact is that equipment becomes obsolete very fast, and software is constantly upgrading. You don't have to buy equipment which suits our needs right away, you wouldn't want powerful computers to serve as typing machines, would you? People and their community always comes first, so maybe it is good idea to ask people what are their needs are, and give them time to explore them.

Doubt 2. Successful IT solution is only integrated solution with support for all librarian's tasks and needs.

We believe that perfect software doesn't exist[21], and that integrated solutions may be good choice for some people and institutions. Problem with integrated solutions is that they have to be universal: they provide only solutions to most common problems in one or only few ways.

Doubt 3. When you buy a software solution, it is important to buy it from well known and expensive vendor. That way you will get lifetime support for that product, together with improvements suited to your needs.

We believe that open source give us more insight in what we are going through while trying to persuade information technology to help us. Within our cooperative environment, we like to promote open access to information, not allowing some proprietary rights to harm the right of the user.

During the project we managed to define our jobs better, to handle our responsibilities towards users and enjoy working.

Having said all that, we are looking forward to new challenges that this project will impose on us. We are quite confident that we can deal with them, and make our librarians, users and community happy.

---

[20] Often, practice seems to be: first buy an expensive server and then try to figure out with to do with it.

[21] We are even quite sure that it doesn't. We could try to find some scientific reference to prove that, but anybody using computers would agree with us even without that.

References

1. **Special Issue: Open Source Software**; *Information Technology and Libraries, 21(1) March 2002;* available also on-line; http://www.lita.org/ital/ital2101.html
2. **Open Source Systems for Libraries**; *on-line site;* http://www.oss4lib.org/
3. Dan Chudnov: **Open Source Library Systems: Getting Started**; *August 1, 1999 issue of Library Journal*, but original version on-line; http://oss4lib.org/readings/oss4lib-getting-started.php
4. Eric Lease Morgan: **Comparing Open Source Indexers**; *on-line*; http://www.infomotions.com/musings/opensource-indexers/
5. Owen de Kretser, Alistair Moffat: **Needles and Haystacks: A Search Engine for Personal Information Collections**; *Proc. 23nd Australasian Computer Science Conference, Canberra, Australia, February 2000, 58-65.*; http://www.cs.mu.oz.au/~alistair/abstracts/dm00:acsc.html
6. FS Consulting Inc.: **The MPS Information Server**; *on-line*; http://www.fsconsult.com/products/mps-server.html
7. Frank Hecker: **Setting Up Shop: The Business of Open-Source Software**; *originally published May 1998, revised 20 June 2000, revision 0.8, draft; on-line*; http://www.hecker.org/writings/setting-up-shop.html
8. K M Elisabe Murray: **Caught in the Web of Words: James A. H. Murray & the "Oxford English Dictionary";** *Yale University Press, 1995.*
9. Simon St. Laurent, Joe Johnston, Edd Dumbill: **Programming Web Services with XML-RPC**; O'Reilly & Associates, Inc., June 2001.
10. WebPAC development web site; *on-line resource;* http://WebPAC.sourceforge.net/