

Improving the Formatting Tools of CDS Invenio

Jérôme CAFFARO

CERN-THESIS-2009-057
20/10/2006



Supervisors:

Jean-Yves LE MEUR Dr. Pearl PU FALTINGS
IT-UDS-CDS, CERN I&C-IIF-HCI, EPFL

October 1, 2006

Abstract

CDS Invenio is the web-based integrated digital library system developed at CERN. It is a strategical tool that supports the archival and open dissemination of documents produced by CERN researchers. This paper reports on my Master's thesis work done on BibFormat, a module in CDS Invenio, which formats document meta-data.

The goal of this project was to implement a completely new formatting module for CDS Invenio. In this report a strong emphasis is put on the user-centered design of the new BibFormat. The bibliographic formatting process and its requirements are discussed. The task analysis and its resulting interaction model are detailed.

The document also shows the implemented user interface of BibFormat and gives the results of the user evaluation of this interface.

Finally the results of a small usability study of the formats included in CDS Invenio are discussed.

“Good tools obviate bad processes.”

Dr. Michael B. Johnson, Pixar Animation Studios,
Apple WWDC, August 2006

Contents

1	Introduction	1
1.1	Context	1
1.1.1	CERN	1
1.1.2	CDS Invenio	1
1.1.3	BibFormat	2
1.2	The Project	4
1.2.1	Motivations	4
1.2.2	Objectives	4
1.2.3	Activities	5
2	Analysis	7
2.1	Task Analysis	7
2.1.1	Product Statement	7
2.1.2	User Population Analysis	7
2.1.3	Personas	11
2.1.4	User Needs	17
2.1.5	User Scenarios	18
2.1.6	Task Tables and Task Map	19
2.1.7	Usability Goals	24
2.2	Comparative Analysis	24
2.2.1	Previous Formatting Module	25
2.2.2	Competitors	30
2.2.3	Synthesis	34
3	Design	37
3.1	Specifications	37
3.2	Prototypes	40
3.3	Final UI Design	44
3.4	Formatting Engine Design	51
4	Evaluation	57
4.1	User Evaluation of the BibFormat Admin User Interface	57
4.2	Usability Study of the Formats	58
4.2.1	Goals of the Study	59
4.2.2	Experimental Conditions	60
4.2.3	Results	61
4.3	Further Improvements of BibFormat	62

5 Conclusions	65
Appendices	
A UML Use Cases	69
B BibFormat Developers APIs	73
Bibliography	79

Chapter 1

Introduction

1.1 Context

1.1.1 CERN

CERN (European Organization for Nuclear Research) is the world's largest particle physics center[1]. Located near Geneva on the Franco-Swiss border, it employs 3000 persons to operate, maintain and develop the complex facilities that allows scientists to discover the building blocks of matter.

Founded in 1954, the laboratory was one of Europe's first joint ventures and includes now 20 Member States. Some 6500 visiting scientists, half of the world's particle physicists, come to CERN for their research. They represent 500 universities and over 80 nationalities.

Among the important discoveries for which many CERN scientists have received prestigious awards, including Nobel prizes, is the Web. The Web was first thought by Tim Berners-Lee in 1989 as a solution to the problem of loss of information due to the high turnover of people at CERN. His original proposal[2] suggested to use Hypertext to link information systems accross network such that anyone could have access to any important documents produced at CERN.

The web has now extended worldwide and has become a primary component of IT. Unfortunately some concepts proposed by Tim Berners-Lee have not made their way into the world wide web: there is for example no typed links (\simeq Semantic web) between information systems, and there is still not always a clear separation between the data and its visual representation.

1.1.2 CDS Invenio

CDS Invenio¹ is the integrated digital library system developed and used at CERN[3]. It is in some ways the implementation of the original idea of Tim Berners-Lee for scientific documents.

As CERN researchers issue about 2000 publications per year, CDS Invenio is an essential tool to capture all of this information and turn it into shared knowledge. At CERN CDS Invenio currently has a database of more than 800'000 bibliographic references, including 360'000 fulltext documents[4]. These documents are organized in more than 500 collections. In additions to the documents

¹Formerly known as CDSware

produced at CERN, CDS Invenio harvests about 100'000 documents from external sources. CDS Invenio uses standards at its core. It is OAI-compliant[5] to support the open dissemination of these documents. CDS Invenio also uses MARC 21[6] (and its XML derivative MARCXML) to store and process bibliographic meta-data.

The server is accessible from a web-based interface, which offers both a fast search of documents and a browsable tree-like structure. Collections of documents can have customizable “portals” to support community building. Additionally, CDS Invenio provides collaborative tools such as baskets or alerts for new specific documents [7].

CDS Invenio is a free, open source application (under the terms of the GNU General Public License). It is developed at CERN by the CERN Document Server (CDS) section, in the User and Document Services (UDS) group. CDS Invenio is now being used by many scientific institutions worldwide, including EPFL[8]. The software complements other librarians’ tools such as Aleph 500[9], with which CDS Invenio can synchronize.

1.1.3 BibFormat

BibFormat is one of the many modules that compose CDS Invenio. Its purpose is to format the records of the database by applying customizable templates. The BibFormat module is almost invisible to end-users of CDS Invenio, in the sense that users only see the formatted output produced by BibFormat, but do not directly interact with the module. Figure 1.1 shows where BibFormat sits in the case of a record access by a user. This workflow is only a conceptual view of

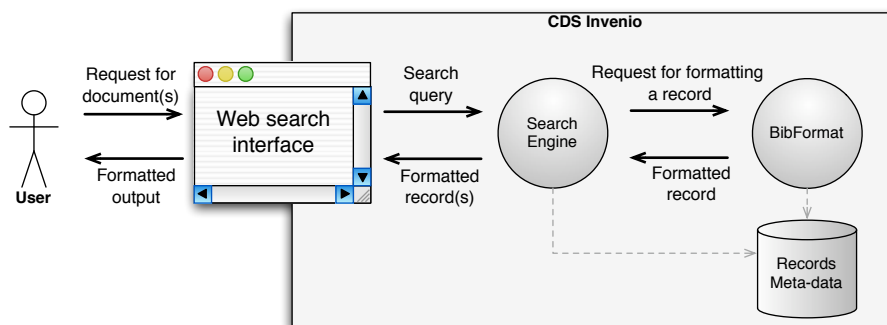


Figure 1.1: Simplified scenario of typical access to a record by end-user.

the process, as other mechanisms such as caching of pre-formatted records are involved. This scenario repeats for each search request of the user as BibFormat is also responsible for formatting each individual notice of the search results list. Figure 1.2 shows the output produced by BibFormat in the case of a search results displaying and in the case of a detailed view of a record. Notice that BibFormat only produces the red surrounded areas, while other components are laid out by CDS Invenio itself.

1.1. CONTEXT

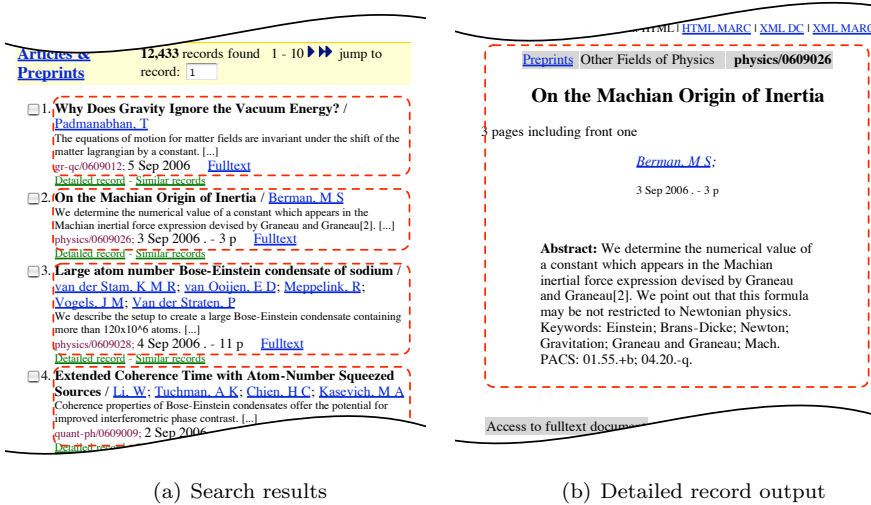


Figure 1.2: Samples of output produced by BibFormat (red surrounded areas) and laid out by CDS Invenio

The formatting challenge Although end-users do not directly interact with BibFormat, administrators of CDS Invenio do have to configure it to produce the desired output. Typically not all records will require the same formatting: there are cases where a field (or attribute) of a record is present in a collection, but not in others, or worse, have a different meaning. For example a technical report and a multimedia document do not share the same meta-data and BibFormat should certainly produce different kinds of output for each of these document types. Therefore the customization of outputs is necessary in BibFormat. Considering the high number of different collections at CERN, this makes the management of formats a particularly difficult task.

The problem becomes even more complex when the different variants of a format are considered: for each formatting style you might need to have a detailed version, a brief version for search results, or even a BibTeX version such that people can directly insert a reference to a document in their LaTeX file.

Another particularity of CDS Invenio which makes the formatting difficult is its internal weakly-typed semi-structured data. No constraints are put on the values entered by the users and everything is saved as text, such that it is very easy to harvest documents from different sources using different conventions, but also difficult to provide uniform output values for meta-data. It is for example common to have different abbreviated names for the same journal or author. A module in CDS Invenio takes care of normalizing meta-data of documents, but there are some cases where it cannot apply or where it is more desirable to normalize at formatting time. One might want to change the way a date is displayed, or link a journal its website even if the URL is not saved inside the meta-data. This is why BibFormat must have the capabilities to normalize and enrich documents meta-data.

1.2 The Project

1.2.1 Motivations

Over the last few years CDS Invenio code base has moved from the PHP programming language to Python. BibFormat was the last module still written in PHP. This has put additional constraints on the installation requirements of CDS Invenio, which already has a long list of dependencies on other software. Integration with other Python modules was also limited given the difference of languages. Finally the already existing BibFormat had some serious usability issues that made the edition and management of formats a long and difficult task. The goal of the project was to design and implement a new version of BibFormat.

1.2.2 Objectives

The initial objectives of the project were the following:

1. Implement a fully working Python version of BibFormat.
2. Reach higher usability goals than the previous BibFormat.
3. Design a formatting syntax that is easier to read and understand.
4. Reach at least equivalent expressiveness as with previous BibFormat.
5. Implement requested features.

Implement a fully working Python version of BibFormat The first objective is almost trivial. It has been decided that the developed product would not be a prototype, but a module ready for production. This implied a huge testing and integration work at the end of the project. It also meant that the design specifications might have needed to be scaled down to fit in the five months project's timeframe. Note that it was also decided that the new version would not be just a code translation of the old one, but a completely redesigned product.

Reach higher usability goals than previous BibFormat Shortly after at the beginning of the project, concerns about the usability of the module were raised by the users of the previous BibFormat. Some serious usability issues in the old PHP BibFormat were the source of frustrations. These were pointed out immediately when establishing the requirements of the product. This second point was also motivated by my own interest in interaction design. This is why this report focuses on the user-centered analysis and design of the project.

Design a formatting syntax that is easier to read and understand The third objective is similar to the second one, but more power-users oriented: in addition to improving the user interaction with the module, we wanted to provide a clearer syntax for the template configuration files, even if those were not to be directly manipulated by the novice and intermediate users. The previous BibFormat had quite a complex syntax, that looked like some subset of PHP, and that was not much appreciated by the users of BibFormat. This switch to

1.2. THE PROJECT

a new syntax also meant a loss of backward compatibility that needed to be addressed.

Reach at least equivalent expressiveness as with previous BibFormat

The last main objective of the project was to get the same level of expressiveness with the new BibFormat as with the previous release. This means that every output generated by the old version could be generated by the new one. However this point does not mean that every feature of the previous BibFormat would have to be implemented in the new one. In fact, a lot of features have been dropped when transitioning to the new BibFormat because they were complicating the module with concepts that could easily be simulated by others.

Implement requested features While some features have been dropped, other features requested by users of BibFormat have been added. The most requested feature was support for internationalization of formats. Although CDS Invenio's user interface has already been translated into 16 languages, BibFormat could not be adapted to a user's language settings.

What the project was not about Finally we can talk about one point that was not an objective of the work: writing new formats. It was clear that this project was not about writing new formats for CDS Invenio. This is a task that would clearly have needed five more months to get done. However some improvements for these formats have been studied, even if they were not initially planned.

Personal objectives On a more personal point of view, this project was also the occasion for me to apply principles and methodologies of human-computer interaction and software engineering on a real project.

1.2.3 Activities

This project covered 5 months of development at CERN and 1 month of report redaction. Five major phases occurred in these 6 months. The next figure summarizes these phases, their corresponding activities and approximate duration. This schema differs a bit from the initial project timeframe, as some activities had to be dropped or added, and some reordering occurred. Also note that activities were not always carried in the same order as shown here and sometimes overlapped.

Phases	Activities	Duration
Analysis	Getting to know CDS Invenio & Python	3 days
	Interviews	3 days
	User population analysis	1 week
	Competitive analysis	2 days
	Scenarios and tasks model	1 week
Design	Low-Fi prototypes	1 week
	Module Architecture Design	1 week
	Call for feature requests	3 days
Implementation	Coding	2 months
	Preliminary User Evaluation	2 days
Testing and Integration	Code debugging and Unit tests development	1 month
	User Evaluation	4 days
	Formats Usability Testing	2 weeks
Report Redaction		1 month

The analysis and design phases are discussed in more details in chapters 2 and 3. The user evaluation of the interface and the formats usability study are to be found in chapter 4.

Chapter 2

Analysis

2.1 Task Analysis

2.1.1 Product Statement

“BibFormat is a module of CDS Invenio that formats the records of the database for displaying in reader’s web browser. The formatting is based on templates defined by administrators.”

2.1.2 User Population Analysis

The sources for the population analysis were the CDS member themselves: they are mainly developing CDS Invenio for internal use and for similar institutions, such that they have become familiar with their users over the years. Of course we focus here on users that are concerned by the BibFormat module.

4 main types of users are considered by the CDS Invenio development team. They are known as:

Users They are the CERN users who browse on the CDS website, looking for bibliographic references or full texts.

Internal customers They are the groups (departments, laboratories, etc.) at CERN who have asked to use CDS Invenio for their publications. They can manage their set of documents, known as collections. Collections in CDS Invenio are organized as a tree structure (a collection belongs to a super-collection, and contains other subcollections) that users can browse. Depending on the collection type, internal customers can customize how their collection looks like. A collection is then some kind of mini-website or portal, with regular readers and some kind of moderator or webmaster.

External customers Institute/Company/University who have decided to use CDS Invenio as bibliographic tool. They install and administrate the software by themselves. These people can also have their own internal customers that manage collections.

Collaborators People external to CERN who contribute to the development of CDS Invenio. Mainly developers. These people are usually also external clients.

We now need to analyse the user population in more details. We cannot keep the above categories as such: for example, the external customer is not limited to the people who install the system, but also includes users who browse the library.

The way CDS Invenio is distributed (free, opensource, without intensive marketing or advertising) and the fact that it has been developed for internal use (while trying to fulfill more generic needs than only internal ones) has made that CDS Invenio is dominantly used in universities. The trend in these universities is to replace or complement the old library system with a more modern one, in particular which offers standardized metadata, interoperability with other libraries and that can make their documents freely available on the web.

It results that (end-) *users* of the system are mostly students and workers with a university education level and a background in science (social science, physics, etc.). They consult the CDS Invenio database in order to document their works. We will hence refer to these users as **readers** (even though they might also publish papers, but this does not concern BibFormat). Among these users we can find beginners (typically students) who might not be used to online libraries and only have very basic knowledge of computer tools, limited to web browsing and office software. The other type of end users are scientists, who almost daily deals with library system and computer productivity tools to write their reports, do their experiments, etc. They are hence familiar with bibliographic search. BibFormat should have a limited or indirect impact on these users: although it formats the results they will read, the choice of the formatting is let to the managers of the various collections, who should be the best experts to make such a decision, as we will see. It is however the biggest population of our analysis, hence the importance for Bibformat to provide adequate customization tools.

Now let's discuss the *internal customers* category. They are the groups that have asked to set up a collection for their needs. Most of the persons in this category are also the reader users we have just seen: they just want to submit their papers and consult others'. However there is one user in this category who is different from the others: it is the administrator of a collection. It chooses how the bibliographic references will look like in the collection it manages. For example, the administrator of some photos collection might decide not to display the date tag of the images in the list of search results. We will see different kinds of administrators. The first one belongs to this category of external customers, and can be called the **domain expert administrator**: this person is one of the readers who has been given the responsibility to manage the collection of his group. He is expert in the domain he has to manage, but has no or few knowledge in HTML or in programming language. In the current version of CDS Invenio this user has usually not administration rights to modify the formatting of his collection. He does usually ask the CDS Invenio development team to write a custom format for the collection he manages, and does not have to care about the management of the collection after it has been set up. However in the future CDS Invenio might allows this user to edit and create his own formats.

The biggest internal customer at CERN, who has to deal with many collections, is totally different from those we have seen. It is the people of the library, who directly manage most of the documents at CERN. The library staff have had a special training in bibliographic systems. They know about the bibliographic

2.1. TASK ANALYSIS

standard MARC 21. They are not particularly familiar with any programming language. We call them the **librarian administrators**. Currently defining the formatting of a collection requires a good knowledge of programming. Although they have been shown how to write custom formats, they have never really been able to do it because of the complexity of the previous BibFormat. This is why the administrators that we have seen most often relies on a third one to write the formatting part. This third administrator is simply a programmer (**developer administrator**), usually someone from the CDS Invenio development team who offers support for this kind of tasks.

The *external customers* category needs refinement: it includes many categories that we have already discovered, like readers and librarian administrators. One new user belongs to the external customers: it is the **system administrator**. It is the person who installs and maintains a running copy of CDS Invenio for his institute. He is very like the developer administrator, but he does not necessarily know about Python and does not want to change the source code of the application to customize CDS Invenio. We could also decide to consider another external customer, who is in charge of deciding if he will use CDS Invenio or a concurrent product (This person might be different from the system administrator). The capabilities provided by BibFormat to have custom formats for the references can have a considerable weight in the decision process. As there is no strong will to “sell” the product and that the development is strongly oriented for internal needs, we do not have to focus on this user. Moreover the choice of this decision maker of using or not CDS Invenio should be based (concerning BibFormat) on the fulfillment of the needs of the users we have considered above.

The *collaborators* category has somehow already been discussed through the previous points. They are made of the same readers and administrators users as at CERN.

Now let’s summarize the categories of users that we will consider. They are ordered according to their knowledge in the field where BibFormat applies:

- Readers
- Domain expert administrator
- Librarian administrator
- System administrator
- Developer administrator

As readers will not deal with the same part of the software as the others, we must consider them separately. Although we said that they were not direct users of BibFormat, we can still analyze this population, as it will tell us which default format we should provide in the CDS Invenio installation package.

Readers

Here is the user profile of the reader users of CDS Invenio. I have compiled the statistics of CERN (Statistics of 2004 [10]), EPFL (2005 [11]) and RERO (2005 [12]) a swiss online library.

70% of the CDS Invenio server traffic comes from outside CERN. It is difficult to analyze the user population accessing to documents from outside CERN in details.

The remaining 30% readers at CERN are mostly scientists (Research physicists, various scientific and engineering work). 90% of them are CERN employees aged from 25 to 65 years (average: 50 years). The remaining 10% are students aged from 18 to 25 years. 43% of the persons come from France, the remaining come from various countries (mostly europeans). About 5% of the scientific staff are women. As scientists, they master scientific tools and report editing software, but do not necessarily have knowledge in computer science (System installation, maintenance and dedicated software development are managed by the IT department). The majority of the members know at least two languages, including English which is the official language at CERN.

EPFL has about 6500 students, from 18 to 25 years old. 25% of the students are women. The CDS Invenio installation at EPFL mostly targets researchers, doctoral students and students.

RERO is a network of 200 swiss French libraries. It has 150'000 readers, including 35'000 students. The library manages a collection of more than 3'000'000 documents. Only 14% of the documents are in the field of exact science. Other concerns history, literature, arts, etc. 37% of the documents are in French, others are in English (23%), German (21%), Italian (6%) and Spanish (2%). RERO targets university students, teachers and researchers.

Here follows an estimate of the readers population:

5% female students 18-25 y/o	15% male students 18-25 y/o
4% female researchers > 25 y/o	76% male researchers > 25 y/o

Administrators

Domain expert administrator The domain expert administrator is typically not experienced in programming languages. He might know a little bit of HTML. He should typically be a reader too, and belong to the category of researchers. His knowledge in bibliographic notation is limited (he only reads them in reports and write them for his own reports). There are about 30 custom formatting formats that needed to be developed for this user at CERN. As CDS Invenio is installed in about 15 institutes worldwide and that they are potentially 500 collections at CERN that could be managed by this user, this gives more than 100 users of that kind.

Librarian administrator The CERN librarian are mostly women, aged from 20 to 40. They are experienced in administration, bibliographic notation (MARC 21) and bibliographic tools (like ALEPH). There are about 5 librarians at CERN working on CDS Invenio. Given the installed base of CDS Invenio, we can give a rough estimate of about 40 librarians using CDS Invenio.

2.1. TASK ANALYSIS

System administrator The system administrator will install and maintains CDS Invenio. As CDS Invenio is not easy to install we can consider that he has knowledge in UNIX administration, command line tools and web server deployment. He does not have any experience in bibliographic notations and tools. He knows a bit about HTML and XML, but don't have to deal daily with these languages. We can estimate that there are about 15 system administrators.

Developer administrator The developer team working at CERN on CDS Invenio is the typical developer population. They are also the most important developer administrators for CDS Invenio. They must know Python as programming language and understand MARCXML to contribute to the development of CDS Invenio. The population is typically under 40 years old. A large part of the CDS Invenio development team at CERN are students, working on the project for less than 1 year. Most of them know at least Java when joining the team. It has never be a problem for them to learn Python. BibFormat formats are nowadays maintained by a full-time member of the CDS Invenio team. There are about 10 developers for CDS Invenio at CERN including 5 who offer support to internal customers. About 5 external contributors complete the number of developers.

Administrators synthesis While the numbers show that formatting should be adapted to non computer literate people, it would be strategically risky to develop BibFormat exclusively for them. Firstly because the edition of formats is a task that requires resources, and the allocation of a budget for this task to librarians instead of developers is a decision that is beyond the scope of this project. We cannot develop a software exclusively for a category of people if in the end they do not have the resources to use it. Secondly the edition of format is a task that can become quite difficult for complex formats. Managers of CD Invenio were not ready to trade off expressiveness of formats for ease of writing.

We can however afford giving non computer literate persons the tools to specify how the formatting should look like and also anticipate future developments. Still we must balance their needs with the needs of the CDS Invenio developers, who will remain for a while the only persons with the permissions and resources to edit formats.

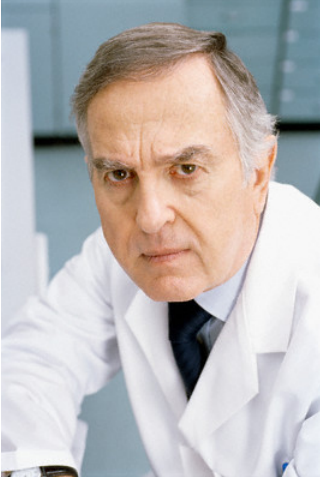
10% System administrator	40% Developer
20% Domain expert	30% Librarian

2.1.3 Personas

Five personas representing the main categories of our user population were created. These personas were especially of a great help to prioritize the needs of our broad user population.

A complete profile has been attached to each persona, to better concretize the users for whom I was developing, and enrich feedback from my co-workers.

The first persona from our end-user population is Silvio Pedone, a CERN researcher:

	<p>Name Silvio Achile Pedone Age 56 y/o Job Physicist Nationality Italian Languages Italian, English, French Work hours 8.30am to 5pm Education Doctor in physics Location St-Genis, France Technology Unix system, latex, matlab. Web browsing and email. Income 8000 CHFr/month Disabilities Wear glasses Family Widower. Has 2 sons. Hobbies Reading science magazines. Walking in the mountains and ski. Goals Make science progress. Make his research recognized by the scientific community in particle physics.</p>
---	--

Silvio Pedone has always been interested in science. He has been encouraged very soon by his parents to think by himself and study hard to reach his goals. He studied physics at University of Padova, where he got his phd in physics. Soon after the end of his studies he found a job as researcher in nuclear physics at CERN. He moved to Geneva, where he met his wife Mireille. They got 2 children. Silvio has seen the evolution of technologies with great hopes for science.

Silvio works in the theoretical physics laboratory on string theory with 15 other persons. He regularly publishes articles about his researches in scientific magazines. He also publishes them on CDS Invenio so that his work is easily accessible from the web to anyone.

2.1. TASK ANALYSIS


Stephanie represents a second reader from our user-population:

	<p>Name Stephanie Jarlet Age 23 y/o Job Biology student Nationality Swiss Languages French, English and German Work hours Flexible. Usually 6 hours a day at university. Works 2 hours at home in the evening. Education Bachelor degree Location Lausanne, Switzerland Technology Windows PC and Microsoft office. Web browsing, emails. Income None Disabilities She sometimes wear glasses to read. Family Has a younger brother, who lives at home with her parents, in Neuchatel. She lives currently with 2 friends in a student lodging. Hobbies Cinema and shopping Goals She would like to easily find results of studies that have been done to document the lab reports she has to hand out to her professors.</p>
---	---

Stephanie is a 3rd year student at EPFL. She studies biology. She had first thought to become mathematician and has been studying maths for one year. She preferred biology because it was less abstract. She really does not regret her choice.

Stephanie has practical labs at school. She has to hand out about one report per week. She must document herself for each of them. The professors have recommended the students some books at the beginning of the semester, and the assistants are always here to help. She also uses google to find paper references.

Alain represents the user that is managing the formats as a full-time job in the old PHP BibFormat, but who will finally be able to spend more time on other tasks once the new BibFormat is released, when management of formats has become easier. He keeps in touch with his customers for establishing the requirements of the formats. Alain represents the advanced user of BibFormat.

	Name Alain Boilat
	Age 42 y/o
	Job CDS Invenio Developer at CERN
	Nationality French
	Languages French, English
	Work hours From 8.30 am to 5 pm
	Education Master degree
	Location France
	Technology Unix, Windows, Java, C++, Python, XML, etc
	Income 6000 CHFr
	Disabilities None
	Family Married, 2 children
	Hobbies Football, and sports in general
Goals Enjoy his work. Make CDS Invenio fills the need of CERN users.	


Alain Boilat has been working on CDS Invenio for more that 6 years. He has become one of the few to fully understand the different modules of CDS Invenio. He implements about 2 major new features per year on CDS Invenio. A large part of his work is to maintain the existing CDS Invenio installation at CERN, and provide all kind of support to its users: correct bugs, assist new users and write new custom formats for the collections that are created. He regularly receive a request to write one new formats. He wished that it could take less time on his work, either by letting the users do it, or by having more adequate tools that he could use to write and manage formats more easily.


Alain works on making CDS Invenio meets its customers needs. From time to time he gets a request to modify/correct the formatting of a given collection. Most of the time these request comes from a librarian. He can do this very easily, as he knows almost by heart where the faulty format lies. It is only a matter of minutes for him to fix any problem. He knows that it would take the people at the library much more time. To edit a format, Alain uses the web interface. This can become a pain, as it does involves programming, but without the usual tools that a developer needs, like syntax highlighting, debugger or even keyboard shortcuts.

Alain lives near CERN, with his wife and his two girls. The youngest of his girl his very proud of her father and want to become developer too.

2.1. TASK ANALYSIS

The librarian user was very important to take into account: some CERN librarians have shown a great interest in being able to edit format themselves, for doing small modifications. With the new BibFormat they should be able to do the regular maintenance of formats, and ask assistance to CDS members for complex projects. Taking into account the librarian user will also lead to a design compatible with persons who are only familiar with basic computer usage.

	<p>Name Marisa Borboen Age 32 y/o Job Librarian at CERN Nationality French Languages French, English, Spanish Work hours From 8.30 am to 5 pm Education High school Location Ferney-Voltaire Technology Windows, Microsoft Office, Aleph, CDS Invenio. Web browsing. Income 4000 CHFr Disabilities Wear glasses Family Single Hobbies Reading people magazines and polars. She likes TV sitcoms. Goals Offer high quality service to the readers of the library. Hope to get married with a rich man.</p>
<p>Marisa has been working for the CERN library for 5 years. Previously she was secretary in a bank, near Annecy. She has been fired due to budget cuts. She jumped on the occasion to do a one year long formation of librarian. She feels very lucky to have found a job at CERN.</p> <p>She now lives in a flat in Ferney-Voltaire, 5 minutes from CERN by car. She likes its quiet surroundings, which allows her to peacefully reads books outside on public bench.</p> <p>Marisa is quite excited to work for the CERN library. She feels that CDS Invenio is one of the indispensable tools for the users of the library, and therefore tries to maintain the bibliographic references as clean and updated as possible. As a librarian, she feels that the web could be a wonderful way to discover new writers and books, may it be fiction or scientific books, but that it still lacks some tools for this purpose. Marisa maintains a small blogs, where she can share her passion for some books and TV shows. She has learned some HTML, but did never really have to use it. However she would also like to help for some parts of the CDS Invenio website, as long as she does not have to deal with technical problems.</p>	

	<p>Name Patrick Teneau Age 21 y/o Job Computer Science student, internship at CERN Nationality French Languages French, English Work hours From 8.30 am to 5 pm Education Bachelor degree Location Annemasse, France Technology Unix, Windows, Java, C++, HTML, XML Income 1200 CHFr Disabilities Wear glasses Family Single. Lives by his parents. Hobbies Computer science and Sci-fi reading. Plays online role based games. Goals Finish his studies with a great CV.</p>
---	--

Patrick has been studying computer science at IUT (Savoie University, France). He has to do an internship during his studies, and has had the possibility to do it at CERN, working on CDS Invenio. Patrick did not know Python before starting his internship, but he quickly learned it. Patrick still lives by his parents, in Annemasse. He travels everyday to CERN by car.

During his short internship Patrick has been asked to develop new formats for BibFormat. The first step involved to read the BibFormat documentation. Understanding how to write formats was difficult as it almost required to understand the underlying architecture, and to learn a new language specific to BibFormat. The web interface allowed him to create a new format. The writing had to be done in a standard web text field, which was awful to use. Patrick fooled the system by using an external HTML editor to write the skeleton, then copy paste the results to the bibformat field. He could then benefit of all the nice tools of his editor, and then preview the result in CDS Invenio. The only problem is that he had to do modification after each copy-paste, to adapt the raw HTML to the BibFormat syntax.

The persona of Patrick was invalidated by members of the CDS team very soon in the analysis phase, as he did not correspond to a role distributed there. Currently one full-time member of the CDS team is responsible for managing all the formats, and this task is not assigned to temporary members of the team. There are two main reasons which make that this job must be done by a full-time member: firstly learning BibFormat and writing formats is hard, and secondly the work must be supervised by someone who knows the requirements of the formats. Still a big part of the work is spent on actually writing the code of the formats, while it would be better spent in the other tasks of the work, like establishing the requirements of the formats or designing the formats.

Nevertheless introducing a user like Patrick, who represents the novice user that was critically missing in the design of the previous BibFormat, will help to reduce the time required to learn BibFormat and write formats. This might eventually leads to distribute this job to other resources, such as temporary

2.1. TASK ANALYSIS

students. This novice user (but still computer literate), is also necessary to prevent the issue discussed in section 2.2.1, which makes of BibFormat users almost perpetual novices.

Still this persona has evolved a bit in the course of this analysis, to better reflect the current situation in the CDS team. He is now working closely with Marisa and Patrick to establish requirements of the formats. He keeps tracks of the requirements in an Excel spreadsheet to which he can refer when he has to modify a format. He also works on other modules of CDS Invenio from time to time.

Prioritization of personas The most important persona that will interact with BibFormat is Patrick. It is mainly for him that BibFormat is going to be designed.

Our secondary user is Alain, as he only use BibFormat from time to time.

Our third user is Marisa, although she would ideally be our primary user: Marisa knows the requirement of the formats, but as explained in the user population analysis it is strategically risky to design in priority for Marisa.

The two last personas do not directly interact with BibFormat. We do not design an interaction model for them, but we must keep in mind that our three main personas will develop formats for them.

2.1.4 User Needs

The high-level needs of the personas are the following: Patrick's needs:

- Create new formats for new collections (totally new or duplicate from existing one)
- Modify the look of existing formats (Colors, fonts, layout, etc.) according to librarians' directives.
- Modify the information displayed by an existing format (which fields are displayed) according to librarians' directives.
- Check the quality of his modifications on formats.
- Assign a format to a new collection.

Alain's needs:

- Modify the look of a format (Colors, fonts, layout, etc.) according to librarians' directives.
- Modify the information displayed by an existing format (which fields are displayed) according to librarians' directives.
- Sets the format for a given collection
- Keep track of requirements for each format.
- Make sure that everything is working right with the formats in production.

- Define formats that require complex business logic that other users cannot or do not want to deal with.
- Define new kinds of outputs (brief output, portfolio output, Excel output, etc)

Alain and Patrick's needs are almost identical. They mostly differ in the fact that Alain takes care of all modules of CDS Invenio and only punctually uses BibFormat, while Patrick uses it almost every day. They will also differ in the manner to execute their tasks.

Marisa's needs:

- Modify the look of a format (Colors, fonts, layout, etc.)
- Modify the information displayed by an existing format (which fields are displayed) according to modification made to meta-data of library's items.
- Modify references list such as list of journal's names and websites, or collection names, which are not saved directly in records.

2.1.5 User Scenarios

We discuss here typical scenarios for the three main personas of our analysis.

Patrick's scenarios Patrick is responsible of the formats used at CERN for CDS Invenio. When a new collection is added, Patrick has to create a new format, which usually is only a slightly modified version of an already existing format. He mostly needs to add or remove displayed fields, to match the meta-data of the new collection. He modifies the labels, and sometimes has to provide a default value for a field in case it can be empty for some records. Patrick follows the requirements of the collection owners (mainly librarians) regarding the fields that have to be displayed, their ordering and their labels, but is most of the time free to choose the appearance of the page (fonts, colours, general layout). He tries to keep a uniform look across collections, but gives priority to the wishes of collection owners.

For each new collection, Patrick should design different formats: one detailed HTML format, one brief format (for the search results), and other formats for a BibTeX output, Excel output etc. However most of the time only a new HTML detailed format is written, unless there are specific needs to adapt other output formats.

Patrick likes to try different designs, and therefore wants to be able to quickly see the results of different attempts.

Given that records of the databases are entered by various users who use different conventions and harvested from different sources, Patrick has to adapt formats to produce the most uniform output as possible. He wants for example that a journal name abbreviated in two different ways is always displayed in the same way. To do so Patrick maintains a list of mappings that the formatter can use to normalize output.

2.1. TASK ANALYSIS

Alain's scenarios Alain is responsible of the complex and low-level tasks of the maintenance of the CERN documents server. He has to ensure that the server is always available to users. He backups the system, fixes bugs, etc. When he receives a request to add a new collection, he defines the requirements with the client regarding the meta-data that will be collected, the look of the collection and the submission process. Alain delegates the task of implementing the submission page and the formatting of the collection to his colleagues, while he prepares the database to support the new collection.

From time to time Alain receives a request for a small modifications in the collections, which might require minor adaptations of a format. In that case Alain opens the format file and do the modification by himself.

It also happens that Alain takes care of writing the complex business logic behind a format. In these cases he prefers to use all the power of his UNIX tools and scripting languages than a web interface.

Alain sometimes cleans the existing templates. He tries to eliminate duplicate or very similar ones, and delete those who are not in use.

While formats are collection-dependants, users wants to be able to select custom outputs (like BibTeX or detailed output) which apply to all records. Therefore Alain must be able to define new categories of output which will take care of selecting the right template for the right record given the selected output.

Marisa's scenarios Marisa is responsible about one third of her time of the loans at the users desk, one third of the time registering new books in the Aleph system and one third of the time preparing the various CERN periodicals. When she inputs new records in the Aleph system, she knows that they will be available to users from the CDS Invenio website. Marisa must create custom formats for the different publications at CERN. She needs the tools to create formats with a low-level complexity, simply displaying some fields in tabular environments. She does this by using Dreamweaver the HTML editor distributed at CERN, and for which training session are offered to CERN employees.

Sometimes she notices that an existing format does not display the right field or that a label needs to be clarified. She opens the corresponding format and tries to modify it by herself before calling the CDS support to do it if the task is too difficult for her. One of the big tasks of Marisa is to keep up-to-date list of mappings, which define the normalized version of a field for different values. For example if some records refers to Phys. Rev A or Phys. R. A, Marisa must let the BibFormat know that they refer to Physical review, A. journal, by mapping the different abbreviations to the full journal name. She also does this for authors' names and journals' websites. She just sets these mappings in knowledge bases.

Other scenarios You can find other more detailed and formalized scenarios for these users in appendix A.

2.1.6 Task Tables and Task Map

Patrick's task table is show in table 2.1.6, Alain's task table in table 2.1.6 and Marisa's task table is in table 2.1.6.

The task vs frequency of usage table is in table 2.1.6.

Task	Importance	Frequency	Details
Add format template	Medium	Low	Patrick creates a new format template, or copy of existing one, using web interface
Retrieve a format template	Medium	Medium	Patrick finds a format template by its name and description in the list of templates
Edit look of format template	High	High	Edits fonts, colors, layout, ordering of elements using WYSIWYG editor
Edit displayed information of format template	High	Medium	Modifies displayed fields of the record according to requirements
Check format template validity	High	High	Patrick wants to be sure that he has made no error, after his modification
Preview format template	High	High	Patrick can see a preview of his work with real records
Delete format template	Medium	Medium	Patrick clicks on delete button in the list of format template
Modify format template name	Medium	Low	Patrick clicks on “modify attributes” in the format template editor
Add new category of output	Medium	Very low	Patrick adds new output format (Excel, XML, etc), using web interface
Retrieve a category of output	Medium	Low	Patrick finds an output format by its name in the list of output formats
Delete category of output	Medium	Very low	Patrick goes the list of output formats and click on “delete” button
Assign a template to a collection	High	Low	Patrick adds a new rule for format template in corresponding output format
Remove the template of a collection	Low	Very Low	Patrick removes the corresponding rule in desired output format
Modify name of category of output	Medium	Medium	Patrick changes the name of the output format in the output format editor

Table 2.1: Patrick’s task table

2.1. TASK ANALYSIS

Task	Importance	Frequency	Details
Add format template	Medium	Low	Alain creates a new format template file directly from his terminal
Retrieve a format template	Medium	Medium	Alain finds a format template by its filename in the list of templates
Edit look of format template	Medium	Low	Edits fonts, colors, layout, ordering of elements using his text editor
Edit displayed information of format template	High	Low	Modifies displayed fields of the record according to requirements, from his text editor
Check format template validity	High	High	Alain checks the correctness of all formats from his terminal, at any time
Preview format template	High	High	Alain can see a preview of his work with real records
Check format template dependencies	High	High	Alain track the dependencies of all formats from his terminal, at any time
Delete format template	Low	Low	Alain remove the file from his terminal
Modify format template name	Low	Low	Alain renames the format template file right from his terminal
Add new category of output	Medium	Low	Alain adds a new output format file right from his terminal
Retrieve a category of output	Medium	Low	Alain finds an output format by its filename in the list of output formats
Delete category of output	Medium	Low	Alain deletes an output format file right from his terminal
Assign a template to a collection	High	Medium	Alain adds a rule in output format file using his text editor
Remove the template of a collection	Low	Low	Alain removes a rule in output format file using his text editor
Add complex format	High	Medium	Alain writes a new script in Python, using his preferred tools
Check validity of output rules	High	High	Alain checks the correctness of all formats from his terminal, at any time
Check dependencies of output categories	High	High	Alain track the dependencies of all outputs from his terminal, at any time

Table 2.2: Alain's task table

Task	Importance	Frequency	Details
Add format template	Low	Low	Marisa creates a new format template, or copy of existing one, using web interface
Retrieve a format template	Medium	Medium	Marias finds a format template by its name and description in the list of templates
Edit look of format template	High	High	Edits fonts, colors, layout, ordering of elements using WYSIWYG editor
Preview format template	High	High	Marisa can see a preview of his work with real records
Edit displayed information of format template	High	High	Modifies displayed fields of the record
Check format template validity	Medium	Medium	Marisa wants to be sure that he has made no error, after her modifications
Assign a template to a collection	Medium	Low	Marisa adds a new rule for format template in corresponding output format
Adds a new journal name (or equivalent)	High	High	Marisa adds a new entry in the journal (or equivalent) knowledge base

Table 2.3: Marisa's task table

2.1. TASK ANALYSIS

Task	Patrick	Alain	Marisa
Add format template	5%	1%	5%
Retrieve a format template	10%	3%	10%
Edit look of format template	15%	1%	20%
Edit displayed information of format template	10%	1%	10%
Preview format template	15%	7%	10%
Check format template validity	10%	15%	5%
Check format template dependencies	5%	1%	5%
Delete format template	3%	1%	0%
Modify format template name	2%	1%	0%
Add complex format	0%	15%	0%
Add new category of output	2%	3%	0%
Retrieve a category of output	5%	5%	0%
Delete category of output	2%	2%	0%
Assign a template to a collection	5%	15%	5%
Remove the template of a collection	2%	2%	0%
Modify name of category of output	2%	2%	0%
Check validity of output rules	5%	15%	0%
Check dependencies of output categories	2%	10%	0%
Adds a new journal name (or equivalent)	0%	0%	30%

Table 2.4: Task vs frequency of usage table

For Patrick, it is very important that :

1. he can modify templates and see the results immediately
2. he can play with the system to learn how it works (without reading manuals, but learn by looking at provided sample format templates)
3. he can easily see all existing templates and retrieve a particular one
4. he can quickly move to more advanced administration tasks

For Alain, it is very important that :

1. he can administrate BibFormat without the web interface, but using his preferred UNIX tools.
2. he can immediately check the status of all formats
3. he can easily see all existing formats and retrieve a particular one
4. he can assign a format to a sets of records
5. he does not have to remember how BibFormat works each time he wants to use it (reduce need of syntactic knowledge of the system)

For Marisa, it is very important that :

1. everything is well documented in case she has a problem (Contextual Help)

2. she can use her HTML editor to edit formats
3. she can do minor modifications without having to contact Alain or Patrick

A summarizing task map is shown in figure 2.1

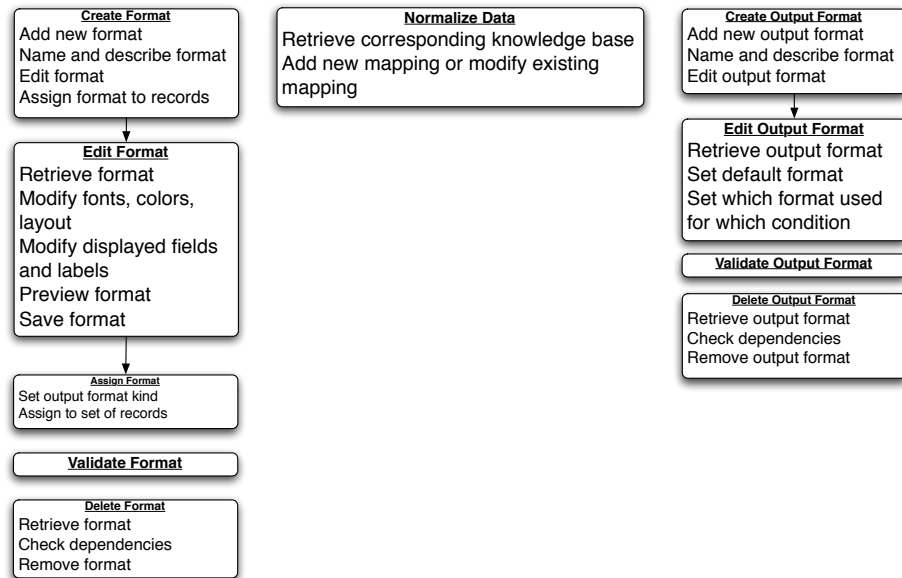


Figure 2.1: Task Map.

2.1.7 Usability Goals

1. Short learning phase (Rely on concepts and tools already known and appreciated by users).
2. Let users try/play with the edition of format, with their own tool or online WYSWYG editor.
3. Easily retrieve an existing format and edit it.
4. Provide preview of the formats.
5. Provide awareness of the status of formats (correctness, dependencies).

2.2 Comparative Analysis

In this section we study the different formatting solutions employed by direct competitors of CDS Invenio and products whose goal is to format bibliographic references, in order to get a glimpse of viable design solutions. Given the short amount of time allocated to the project, this analysis helped avoid spending time prototyping inefficient solutions.

We also describe the current formatting solution available in CDS Invenio previous to this project. This section also gives an idea of the different requirements of the formatting process.

This analysis was done shortly after the tasks analysis had begun, in order to evaluate the different solutions against our own requirements. Still both analysis continued in parallel given that this comparative analysis could also expand the task analysis with interesting elements.

2.2.1 Previous Formatting Module

The PHP version of BibFormat is a complex software that has been included as a formatting module in CDS Invenio until now. It has a web configuration interface. An extract of the main page is shown in figure 2.2. Excepted for the complexity of this page, it is a typical entry page of the CDS Invenio's modules: the main page links to the administrator guide and to the different tools or concepts of the module. The main page has the following links:

Behaviours Define the rules that will decide which format must be applied to a given record.

Extraction Rules Define how the metadata tags from the database are mapped into internal BibFormat variable names. .

Link Rules Define rules for automated creation of URI links from mapped internal variables.

File Formats Define file format types based on file extensions. This will be used when proposing various fulltext services.

User Defined Functions (UDFs) Define functions that can be reused when creating formats. This enables to do complex formatting without ever touching the BibFormat core code.

Formats Define the formatting of CDS Invenio records.

Knowledge Bases (KBs) Define one or more knowledge bases that allow to transform various forms of input data values into a unique standard form on the output. Example: Specify that Phys Rev D and Physical Review D are both the same journal and that these names should be standardized to Phys Rev : D.

Execution Test Test formats with a sample data file.

The most important concepts for the formatting process are *Formats* and *Behaviours*. The *Formats* link leads to a list of formats, as shown in figure 2.3. This page allows to view, edit or delete formats. It also let users insert a new format right from this page. Each format has a name, a description and a code that define the formatting. When clicking on the *Modify* link of a format in the list of formats, an editor for this format is loaded in another window, as shown in figure 2.4 The code of the format is written in EL (Evaluation Language), a custom language invented specifically for this BibFormat. EL is a subset of PHP, with loop and conditional statements, but without variable assignment. Interesting things to note about this language is that EL allows to :

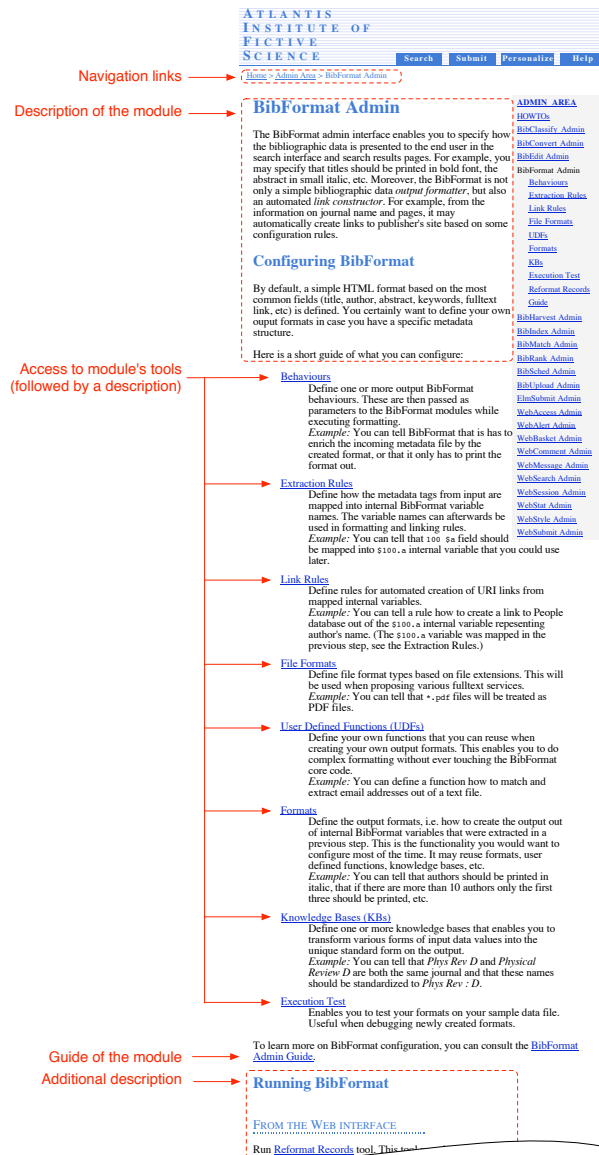


Figure 2.2: Main BibFormat administration page (Old PHP BibFormat).

1. call another format from inside a format, such that formats can be structured in small reusable components (but can also have complex dependencies, or even insoluble circular dependencies).
2. use custom functions, also written in EL, and saved in the *User Defined Functions* section.

2.2. COMPARATIVE ANALYSIS

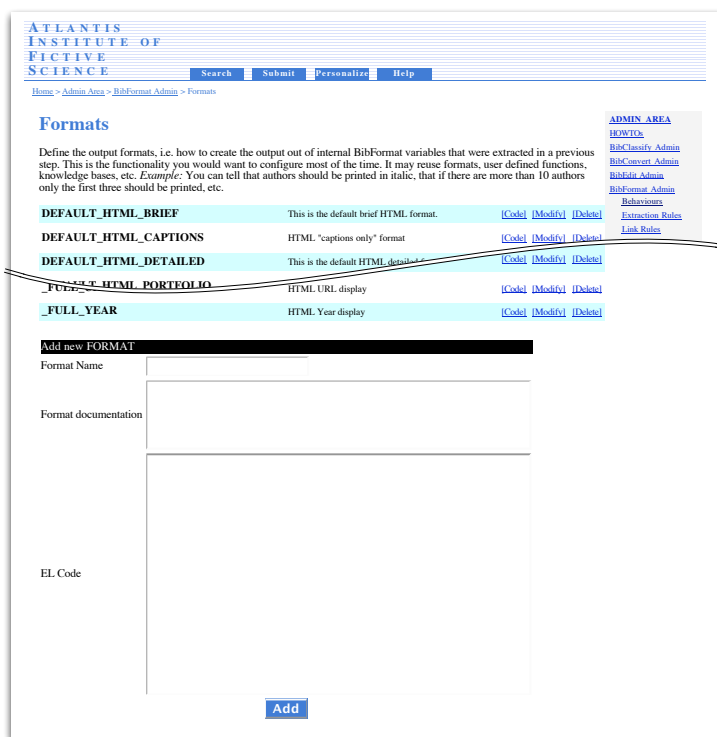


Figure 2.3: Extract of the list of formats administration page (Old PHP BibFormat). Bottom of the page offers controls to insert a new format.

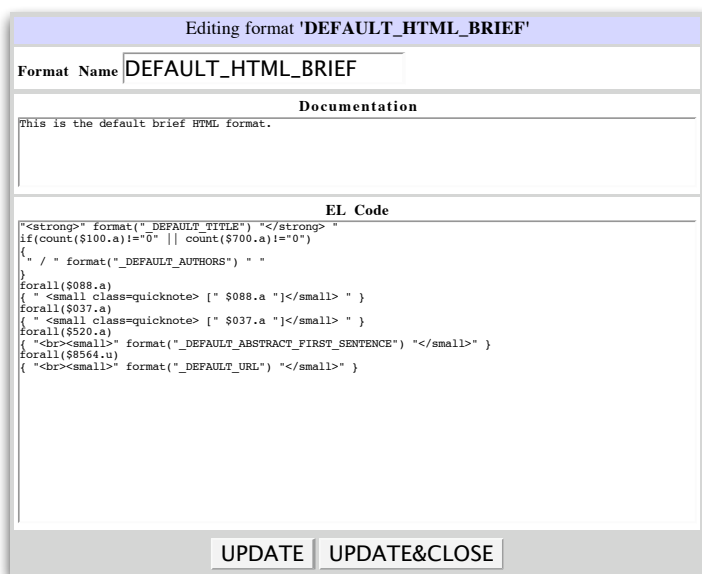


Figure 2.4: Format editor (Old PHP BibFormat).

3. get the value defined for a key in a *Knowledge Base*.
4. create a link to some resource using the *Link Rules*.

This is how formats are managed. Now let's see how these formats are applied. To specify which format is applied to which record, users have to follow the *Behaviours* link of the main page, which leads to a list of behaviours (Figure 2.5). These behaviours can be managed in the same way as formats. A behaviour

Behaviours

Define one or more output BibFormat behaviours. These are then passed as parameters to the BibFormat modules while executing formatting. *Example:* You can tell BibFormat that it has to enrich the incoming metadata file by the created format, or that it only has to print the format out.

Name	Type	Documentation	
DEFAULT	ENRICH	Creates DEFAULT formats and includes them in the input XML OAI MARC record in the "FMT" element	[Details]
HB	NORMAL	Produces HTML brief format. Useful for reformatting records existing in the database.	[Details]
HC	NORMAL	PRODUCES HTML CAPTIONS FORMAT. USEFUL FOR REFORMATTING RECORDS EXISTING IN THE DATABASE.	[Details]
HD	NORMAL	PRODUCES HTML DETAILED FORMAT. USEFUL FOR REFORMATTING RECORDS EXISTING IN THE DATABASE.	[Details]
HP	NORMAL	PRODUCES HTML PORTFOLIO FORMAT. USEFUL FOR REFORMATTING RECORDS EXISTING IN THE DATABASE.	[Details]
HX	NORMAL	PRODUCES HTML BibTeX format.	[Details]

Add new OUTPUT TYPE

Output Type Name:

Behavior Type:

Documentation:

[Add output type](#)

ADMIN AREA

- [HOME](#)
- [BIBClassify Admin](#)
- [BIBConvert Admin](#)
- [BIBEdit Admin](#)
- [BIBFormat Admin](#)
- [Behaviours](#)
- [Extraction Rules](#)
- [Link Rules](#)
- [File Formats](#)
- [URIs](#)
- [Formats](#)
- [KBs](#)
- [Execution Test](#)
- [Reformat Records](#)
- [Links](#)
- [BIBHarvest Admin](#)
- [BIBLinks Admin](#)
- [BIBMatch Admin](#)
- [BIBRank Admin](#)
- [BIBSched Admin](#)
- [BIBUpload Admin](#)
- [FileSubmit Admin](#)
- [WebAlert Admin](#)
- [WebBasket Admin](#)
- [WebComment Admin](#)
- [WebMessage Admin](#)
- [WebSearch Admin](#)
- [WebSession Admin](#)
- [WebStat Admin](#)
- [WebStyle Admin](#)
- [WebSubmit Admin](#)

Figure 2.5: List of behaviours administration page, the rules that define which format is applied to which record (Old PHP BibFormat).

works as a rule-based decision system, which given a condition on the record to format, executes a given portion of code. Typically a behaviour will define a list of rules with conditions on the value of one field of the record (for example that field 980.a is equal to value "PICTURE" or "PREPRINT") and depending on the result of the evaluation of this condition (True or False), execute the corresponding code of the rule (most of the time the code simply calls a format to apply on the record).

The condition and the associated code are written in the same EL language as formats. When clicking on the *Details* link of a behaviour, its editor opens in a new window. This editor lets users add and edit conditions and rules, as shown in figure 2.6

Critique of the old PHP formatting solution

There are some serious issues in the solution provided until now. I think that the most important one is the long learning process required before being able to do anything with the BibFormat:

- Overflow of concepts and information on the main page
- Impossible to learn step-by-step

2.2. COMPARATIVE ANALYSIS

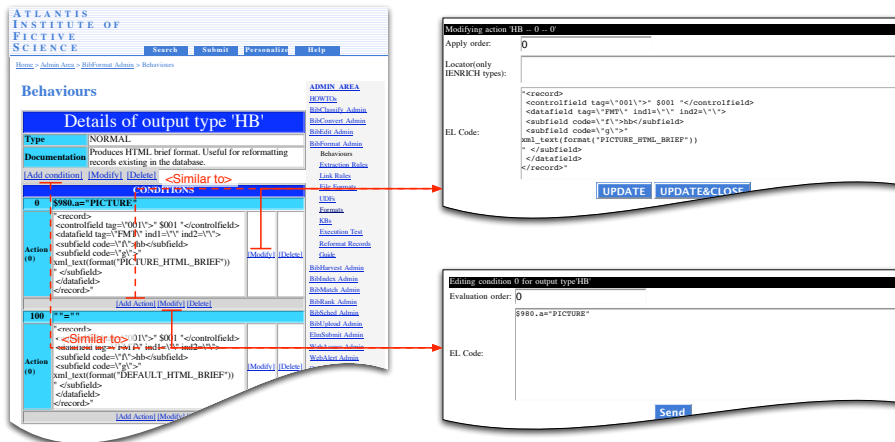


Figure 2.6: Behaviour editor (Old PHP BibFormat).

- Impossible to “play” with BibFormat tools
- User must learn the EL language
- EL language not really suitable as formatting language

The overflow of concepts on the main page might be daunting for a first time user. The provided information labels are too long and make this page look like a manual (and nobody reads manuals). Moreover it is not possible for a user to hope learning BibFormat step-by-step by clicking on each of the links of the page: these tools are mostly interdependent and require a global view of BibFormat. Users also cannot just try each of the tools, as they mostly provide no feedback of the result of the modifications. Another issue is that it is mandatory for users to learn the EL language, which has been designed for, and used only in, BibFormat. All of these reasons make that BibFormat is totally inadequate for novice users.

The EL language also makes BibFormat inadequate to intermediate users: in most situations, formats will be modified only punctually, such that it will be very difficult for users to remember EL between each time they will need it. They always will have to go back to the documentation or read sample code before editing a format.

Moreover even if the EL language has been designed for the sole use of BibFormat, it fails to help users define formats easily. One of the biggest complaints about this language is that is that it requires quoting every outputted text: for example to output **Hello World**, users must write "**Hello World**". In that basic case, this looks pretty simple. In cases where " (double-quote character) is part of the output, every double-quote must be escaped with a / (slash character). As the double-quote is frequent in HTML, it is very difficult not to make an error. When there is an error the output is random, and users have to track manually for the missing or additional double-quote in their code.

This problem occurs mainly because EL is more like a programming language than a formatting language, such that outputted text is not a first-class citizen of EL although it should.

There are also other usability issues:

- Layout of behaviours editor follows no traditional design pattern
- No tool to help users
- Frustrating experience in general
- Controls and links labels are often misleading
- No feedback of user's actions
- No use of CDS Invenio interface guidelines

One thing worth to note is the problem in the behaviours editor: although the mental model of the assignment of a format to a record under the form of a rule-based system seemed totally natural for most users, the interaction model provided by the editor is totally flawed because of its layout and the fact that the rules are coded in EL. It is interesting to see that it would be possible to use the same mental model, and just build an adequate interaction model.

There is usually a lot of formats (more than one hundred in the production server at CERN), and as format can depends on each other, it becomes difficult to track the dependencies when a format has to be modified. BibFormat provides almost no tool to help users manage these formats. In fact even the formats and behaviours editors are simply input fields for the EL code.

Users of the previous BibFormat expressed a lot of frustration when using the web interface: some actions, like opening the editor of a format, would open a new window, but the window would stay behind all others, such that was impossible to know that the action has been executed, or difficult to figure out which window to get back on top. It also happened to some users to loose all of their work when trying to save their modifications.

2.2.2 Competitors

There are many applications that can handle bibliographic references. I will screen some of them, those that have been suggested to me by my colleagues and those that I found to be useful for this project. The focus is put on their capabilities to output differents formats and how much they allow customization of the ouput format. Some of the applications presented here are not direct competitors of CDS Invenio, but they are considered as long as they provide similar functionalities to BibFormat.

Direct Competitors

DSpace (MIT Libraries and Hewlett-Packard Labs) From the authors' website[13] "DSpace is a groundbreaking digital repository system that captures, stores, indexes, preserves, and redistributes an organization's research data".

2.2. COMPARATIVE ANALYSIS

This application offers pretty much the same general features as CDS Invenio. Formatting is defined using a configuration file. The system administrator textually specifies in the configuration file which metadata appears, and in which order. For example, the line `dc.title, date.issued(date), identifier.uri(link), description.*` will show the title, the issue date (rendered as a date), the identifier (rendered as a link) and the DC description metadata. Whenever one of those fields does not exist for a record, it is not displayed. Labels for each of these fields are defined in another global UI dictionary file and automatically added. A third configuration file is used to map the field name to the actual meta-data of the record.

The displayed fields can also be customized for individual collections. This is done by overriding the default line of the configuration file with a new similar line, which specifies the name of the display “style”. Then all collections using this style must be listed textually in another line of the configuration file. This method to customize formatting does not allow to change the style and layout of the page. These modifications must be done by customizing the JSP (Java Server Pages) files that produce the HTML code of each page. This is how the layout of the page can be modified. The colours, fonts, etc. are mostly described in CSS (Cascading Style Sheets) files. The system supports different JSP files (for internationalization), but does not allow to have different styles (same style across all collections). Once the style has been customized, DSpace must be rebuilt and reinstalled to take the style into account.

Next screenshot shows how a DSpace record is displayed to the user.

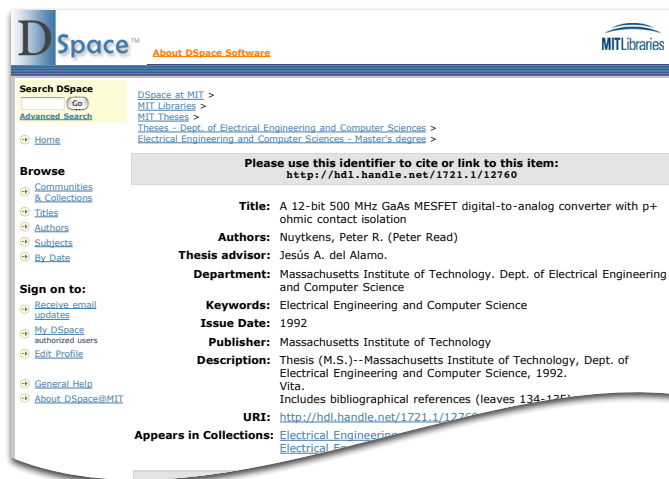


Figure 2.7: DSpace displaying a formatted record

EPrints (EPrints.org Community) EPrints[14] is another competitor of CDS Invenio. There does not seem to be a way to customize bibliographic notices other than programmatically (and it is not documented). Search results formatting can also be customized programmatically.

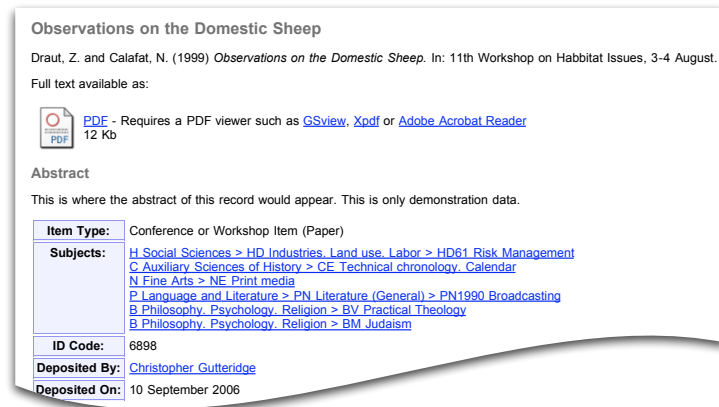


Figure 2.8: Eprints displaying a formatted record

Fedora (Cornell University Information Science and University of Virginia Library) Fedora[15] is a very powerful content management system, whose scope is much larger than the simple dissemination of documents. One could see Fedora as a technology on which systems such as a digital library could be built on. The very abstract “digital objects” Fedora manages support different views and complex business logics. The formatting in Fedora can be defined using XSLT, by accessing the XML internal structure of objects. Fedora hence relies on a standard well-known by webmasters. Fedora developers do not need to create, maintain and document a custom language, and webmasters do not have to learn a new language to format the records.

Applications with similar goals

Pybliographer From the developer website[16] “Pybliographer is a tool for managing bibliographic databases. It can be used for searching, editing, re-formatting, etc”. It is designed to be used as a standalone application, for a single user. It is typically used for storing temporary bibliographic references when writing reports. Figure 2.9 shows the main window of Pybliographer. Pybliographer provides six default output formats: HTML, LaTeX, Raw, Text, Textau and Textnum. There is no GUI for editing new formats. However new formats can be added by dropping format files in a dedicated folder. The formats must be defined using an XML encoding. There is still no documentation on how to write new formats. The developers plan to replace the current formats by an XSL-based mechanism.

Bibioscape (CG Information) From the company website, “the Bibioscape product family is designed to help researchers collect and manage bibliographic data and notes, as well as generate citations and bibliographies for publication”. It is a complete suite of products that is very similar in functionalities to CDS Invenio. The desktop version of the suite allows to edit formats output, as seen on screenshot 2.10.

The interface relies on a list of elements (sequence of predefined bibliographic

2.2. COMPARATIVE ANALYSIS

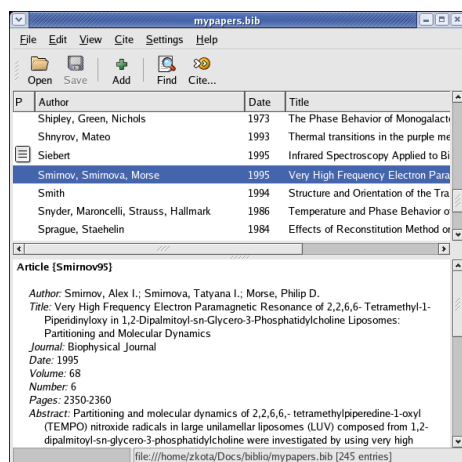


Figure 2.9: Pybliographer main window

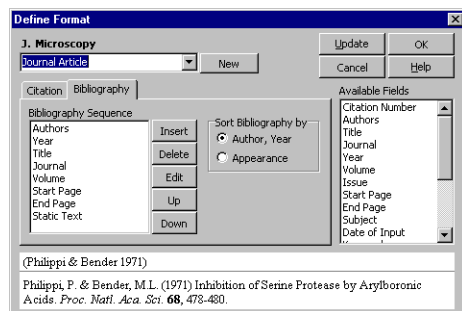


Figure 2.10: Bibloscape, edition of formats.

fields) to specify a format. An edit button allows to refine the format of the selected element. A preview of the format is displayed at the bottom of the window. The preview should be lively updated but an update button is provided in case it does not work. While it seems simple to edit a format, the user manual of the application still provides a support email address to request custom formats. Bibloscape offers hundreds of predefined formats.

BookEnds (Sonny Software) From the company website[17] “Bookends offers a powerful and flexible means of saving, retrieving, and formatting references for bibliographies or footnotes”. While its primary purpose is to be used as a desktop application, it can also be used as a web server.

The application allows to manage and edit formats. Writing a format using the integrated editor is pretty much similar to writing a script. Bibliographic fields are represented using reserved characters. Some controls allow to customize some particular fields, like the authors and editors. The manual of this editor contains than 10 pages, and the creation of styles requires to know the scripting language.

BookEnds provide more that 150 default formats. New formats can be created by reusing the existing base.

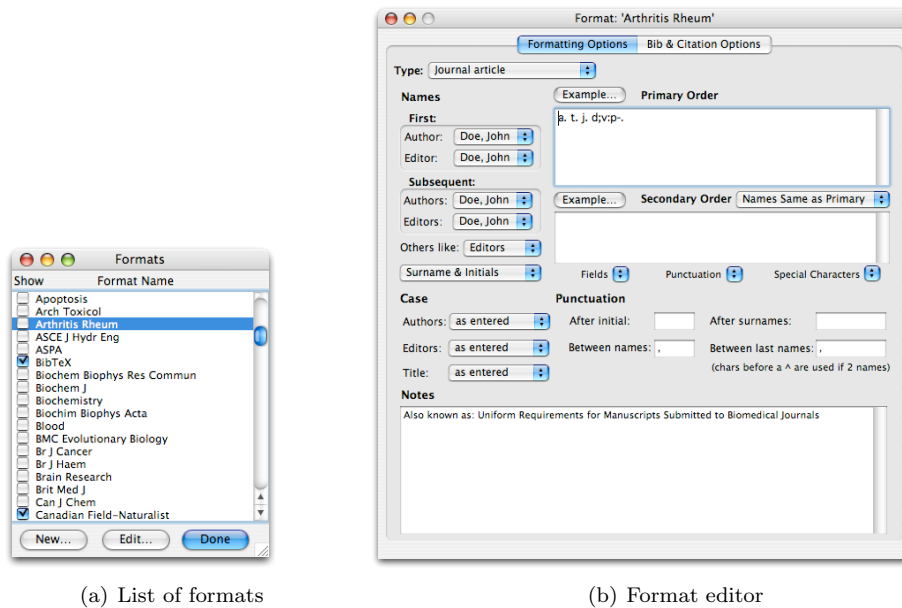


Figure 2.11: Bookends

2.2.3 Synthesis

In table 2.2.3 we review the strengths and weaknesses of the applications we have seen. These applications either provide a scripting language to define formats or a GUI that does basic formatting. Given the user population of BibFormat, we should provide both. Formats management tools are almost non-existent, and maybe useless as these formats have poor expressiveness. For most of these applications the documentation is scarce and lacks sample files.

	DSpace	Eprints	Fedora	Pybliographer	Bibloscape	BookEnds	PHP BibFormat
Ease of learning	4 (need to learn keywords)	1	5 (Lots of tutorials. XSL standard)	2 (Little documentation)	N/A	1	2
Formatting language ease of writing	6 (short english keywords)	N/A	3 (XSL is not especially easy)	4 (easier than XSL, but still XML)	N/A	2 (Lots of special characters)	2 (Lots of text to escape)
Formatting language ease of reading	6 (short english keywords)	N/A	4 (XSL is not especially easy)	5 (easier than XSL, but still XML)	N/A	2 (Lots of special characters)	2 (Lots of text to escape)
Formatting language expressiveness	1 (doesn't allow styles)	?	6	4 (no branches, loops, etc.)	4 (no branches, loops, etc.)	5	6
Management of existing formats	5 (Defined in only one text file. No tool)	?	4 (Defined in individual XSL files. No tool)	4 (Individual XML files. No tool)	6 (GUI to add, edit, remove formats)	6 (GUI to add, edit, remove formats)	2 (Tools difficult to use. Dependencies between formats)
Assigning formats to collections	2 (Time-consuming)	N/A	6 (object-oriented)	N/A	N/A	N/A	3 (Difficult)
GUI editor	N/A (Exclusively text editor)	?	N/A (Exclusively text editor)	N/A (Exclusively text editor)	6	2 (GUI almost of no help)	1 (Basic text input field)
Documentation / Help	3 (Basic. No manual)	1 (No documentation)	4 (Almost no doc, but XSL doc exists)	2 (Almost no doc)	5	5	5

Table 2.5: Synthesis of competitive analysis. Scale go from 1 (weak) to 6 (excellent).

Chapter 3

Design

3.1 Specifications

As a result of the analysis, the following specifications were formalized. The object table 3.1 lists the objects and attributes with which users will interact. The names of the objects have been carefully thought with the users to convey their meaning.

Object	Attributes		Description
Output Format	name		Name of the output format
	description		A description of what the format is for
	content-type		The MIME content-type of the output
	short identifier code		A unique 6 chars long code used to identify the output format
	assignment rules	condition	A condition on the value of a field of formatted record
		format template	The template used for a matching condition
dependencies		Links to format templates used by this output format	
Format Template	name		Name of the template
	description		A description of what the template does
	HTML code	text, images, etc.	Static elements of the formatted output
		format elements	Dynamic element that changes contextually to record, language, etc.
	dependencies		Links to output format that call this template, and format elements used by this template
Format Element	name		Name of the element
	description		A description of what the template does
	parameters		A list of configurable options for the element (varies depending on element)

Knowledge Base	name		Name of the knowledge base
	description		a description of what the knowledge base is used for
	mappings	from	Key of the mapping
		to	Value of the mapping
dependencies		Links to format element using this knowledge base	

Table 3.1: BibFormat object table

Output formats are categories of formatting styles. They can be selected by end-users in the web interface to change how records are displayed. Output format applies to any kind of record: for example the “detailed HTML” output format can be chosen for “publications” as well as for “pictures” documents. For admin-users an output format simply is a set of rules that redirect to a format template matching the type of record being formatted. The output format object more or less corresponds to the *behavior* of the old BibFormat module. The *format templates* simply corresponds to the *formats* of the old BibFormat module, and define how to format a record. It can be modified by all of our personas. A new concept has been introduced, the *format element*. It corresponds to a dynamic brick that can be added into a format template. The format element value will vary according to the record that is being formatted. For example, the *authors* format element will print the value of the authors, and the *title* element will print the title of the record. The object map (Figure 3.1) shows the relationships between these objects.

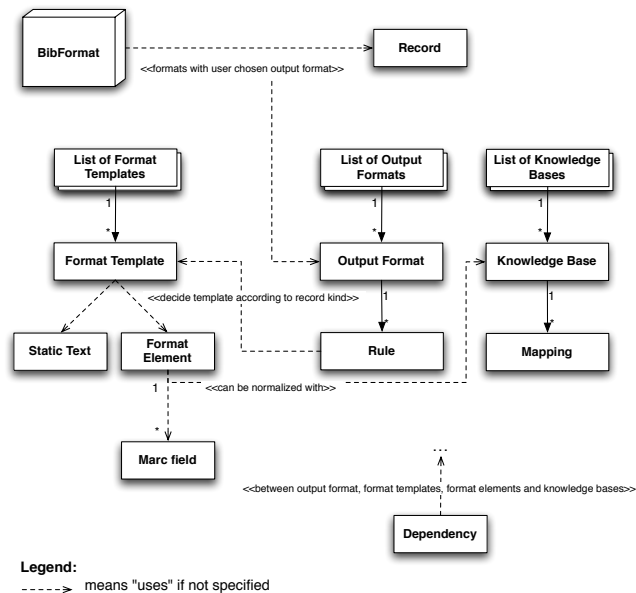


Figure 3.1: BibFormat object map

3.1. SPECIFICATIONS

Figure 3.2 summarizes the structure of the BibFormat administration interface that lets users interact with these objects. It is a mix of action-based and task-based structure.

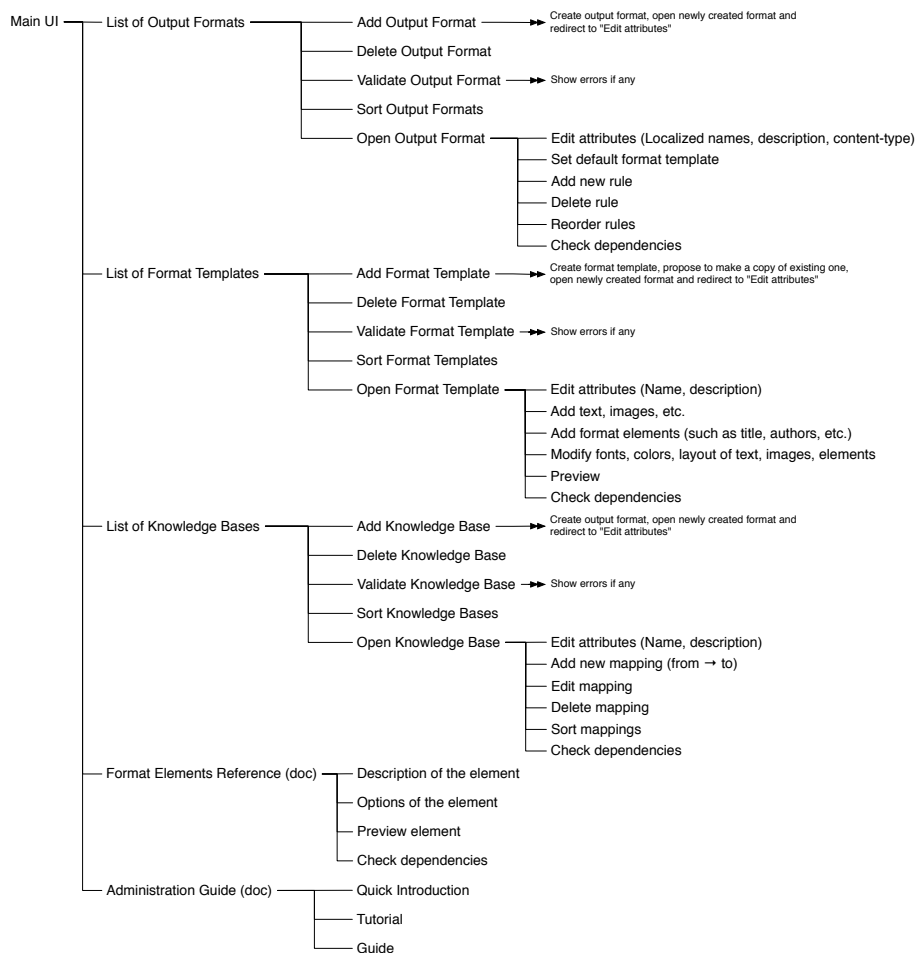


Figure 3.2: BibFormat administration interface structure

Once the specifications were done, an email was sent to all customers subscribed to the CDS Invenio mailing list, calling for feature requests and comments based on the textual description of the specifications. This was a way to validate our model, inform the users of an upcoming version of BibFormat and get them somehow involved in the design process. Feedback was globally positive. Some concerns raised regarding the possibility to use previous formats in the new BibFormat and support for internationalization. Both of these concerns were addressed in the implementation.

3.2 Prototypes

Early paper prototypes were sketched as a means to get a feeling of the possible layouts. Some prototypes were designed even before the analysis was completed, such that they did not fully comply with the specifications. Nevertheless they have provided an interesting basis for the design of the BibFormat user interface.

One of the prototypes shown here (Figure 3.3) is the interface for the management of format templates (called “Stylesheets” in the prototype). It included the possibility to create a template, delete selected templates and edit a template by clicking on the “edit” link of the template. Format templates could be analyzed and checked via a button at the bottom of the page. A column in the list showed the languages for which a format template had been translated.

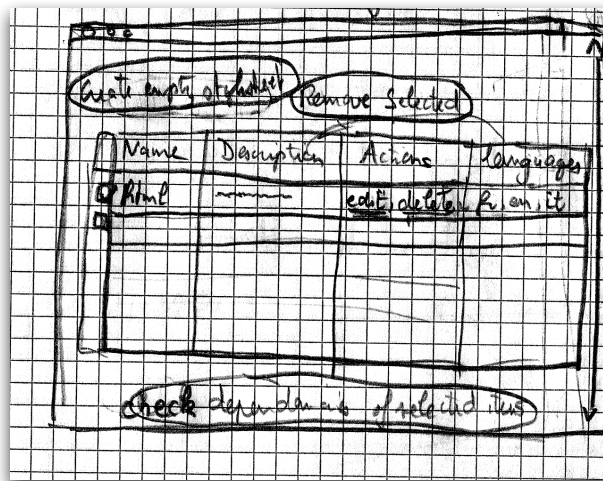


Figure 3.3: Prototype of the format templates management page

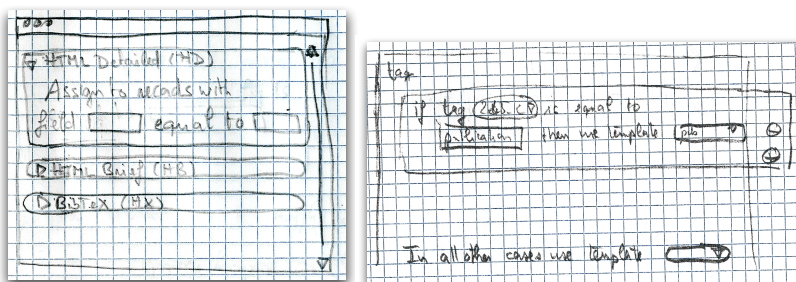
Interfaces in figures 3.4(a) and 3.4(b) allow a format template to be assigned to a set of records. A condition needs to be specified on a field of the record. In a task-based system, this assignment would be done right after the template has been created. This case was studied in Figure 3.4(b). However, as seen in the task analysis, this task was more to be done by a user dealing with the output formats, in a separate interface. Figure 3.4(a) shows the same assignment from this point of view.

Figure 3.5 shows a prototype of the WYSIWYG format template editor, as conceived earlier in the design phase. The idea was to mimic well-known word processors for the edition of format templates.

A button would let a user insert format elements into the template. When clicking on an element, it would expand to let a user modify options, such as a prefix or suffix text that would be printed only if the format element was not empty for the formatted record.

Later in the design phase the prototype was reconsidered in order to better fit in

3.2. PROTOTYPES



(a) Assignment through output formats (b) Assignment through format templates

Figure 3.4: Prototype of the management of the assignment of format templates to records

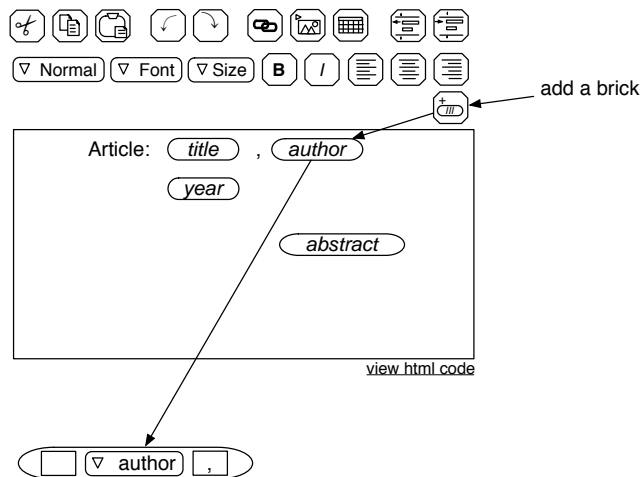


Figure 3.5: Prototype of the format template editor

the project timeframe (Figure 3.8). The WYSIWYG editor has been replaced by a code editor and preview panel. It was also an occasion to give more importance to the format elements documentation, as this was essential to the edition of format templates.

Based on the paper prototypes, an interactive HTML Low-Fi prototype was built. This allowed to test the visual integration with CDS Invenio administration pages and check what controls could fit on a single screen. They were also used to get the interface approved by the CDS Invenio developers.

Figure 3.6 is a Low-Fi version of the paper prototype of figure 3.3 and figure 3.7 shows two different versions of the Low-Fi prototype for the edition of knowledge bases.

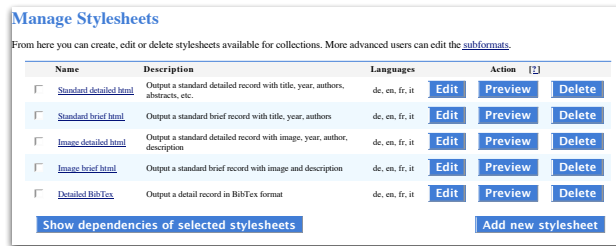


Figure 3.6: Low-Fi prototype of the management of templates

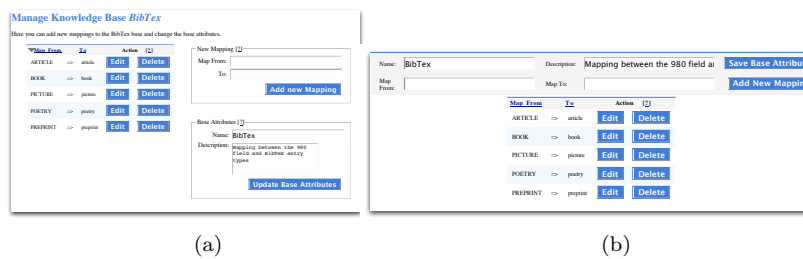


Figure 3.7: Low-Fi prototypes of the knowledge base editor

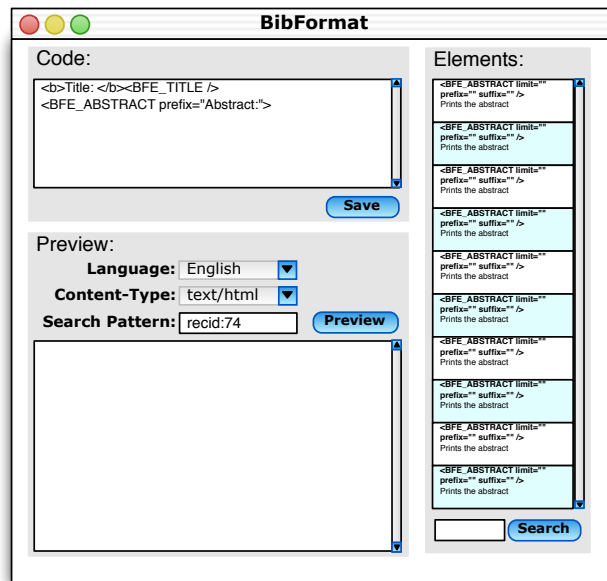


Figure 3.8: Revised rototype of the format template editor

Although unrelated to UI prototyping, it is interesting to discuss the prototypes of languages that have been thought as formatting language. Even if our personas Patrick and Marisa were not to see this language, it was in the interest of Alain to come up with a clear and elegant solution to define formatting.

3.2. PROTOTYPES

The prototypes below show how to output a bold title, followed by the author's name linked to his website (for example), and on next line the year and publication name of the current record.

Language 1

```
<b>$format("title")</b>, $link($AUTHOR_NAME)
<br/>
$260C, $kb($999_t).
```

This language mixes HTML and custom elements starting with a \$ sign. Each custom element is replaced by its value during the formatting process. Fields of the record can either be accessed by their MARC tags or by name. Being able to access fields by name instead of accessing them by MARC tags introduces an abstraction layer which has several benefits: it makes the understanding of the formats easier, allows people unfamiliar with MARC to write formats, and finally dissociates the meaning of a field from its code (which allows the users to redefine a MARC tag for other purposes without having to change all formats).

Language 2

```
<b>$format("title")</b>, $link($100_a)
<br/>
$260C, $kb($999_t).
```

This language is similar to language 1, except that it only allows the user to refer to a field with its MARC tag.

Language 3

```
<b><?py format("title") ?></b>, <?py print link($100_a) ?>
<br/>
<?py print "%s, %s" % ($260C, kb($999_t)) ?>.
```

This language is similar to the PHP programming language: special tags <?py ?> enclose Python code.

Language 4

```
print "<b>";
format("title")
print "</b>, %s <br/>" % link($100_a)
print "%s, %s." % ($260C, kb($999_t))
```

The fourth language is completely standard Python code.

None of these languages were satisfactory, as they either required learning a custom language, or were not suitable for formatting. The software architect of

CDS Invenio has come up with a better idea than the prototyped language: the idea was to use exclusively HTML code. The examples shown above translate to:

Language 5

```
<b><BFE_TITLE /></b>, <BFE_AUTHOR print_link="true" />
<br/>
<BFE_YEAR />, <BFE_PUBLICATION kb="pub"/>
```

In that way we have a language that is HTML valid, and that can be generated by any HTML editor. The dynamic elements (format elements) are tags that start with `BFE`, and are replaced at runtime by values of the record. A format element can take parameters as input, in order to modify the behavior of the element. For example `BFE_AUTHOR` takes `print_link` as parameter, which determines if a link to author's website has to be printed by the element. These format elements are defined in individual Python files provided by the programmers.

See section 3.4 for more information on the different files involved in the formatting process.

3.3 Final UI Design

The implemented web-based administration user interface of BibFormat is shown in the screenshots of this section. The navigation between these pages follows the structure specified in the section 3.1. You can walk through these interfaces (each labeled with a number on their top left corner) by following the orange links printed on the screenshots.

Format templates related interfaces The list of format templates, output format and knowledges bases user interfaces are all similar, and very close to the prototype of section 3.2. Therefore only the format templates list is shown (Figure 3.10).

Format templates and output formats lists interfaces show right in front of each format its status, such that any problem with a format is found immediately. The format templates list has an additional button "Check Format Templates Extensively", which allows errors to be found in the templates that could not be detected by a quick check. When an error is found, the status switches to a red "Not Ok" link, which leads to a description of the problem. Errors can happen for example when an output format refers to a non-existing format template, or when a format template uses a format element which fails to produce an output. Errors should mainly occur when configuration files are modified manually without using the web interface.

A menu (standard in CDS Invenio) that allows the user to quickly switch to other sections of the administration without having to go back to the main menu is placed at the top of the page.

3.3. FINAL UI DESIGN

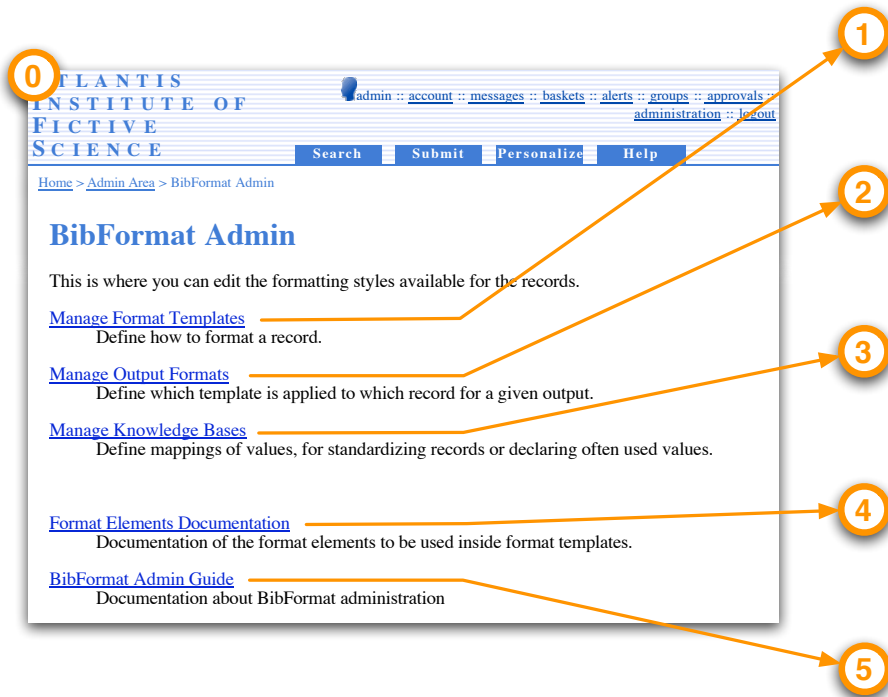
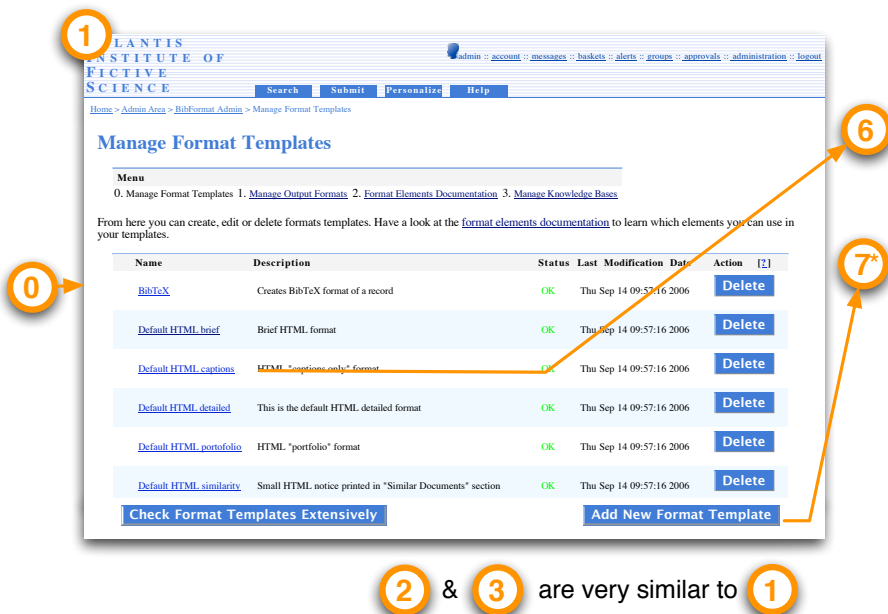


Figure 3.9: BibFormat main menu



2 & **3** are very similar to **1**

Figure 3.10: Format templates list

The format template editor has finally been simplified due to time constraints and technical requirements. The fully WYSIWYG editor has been replaced

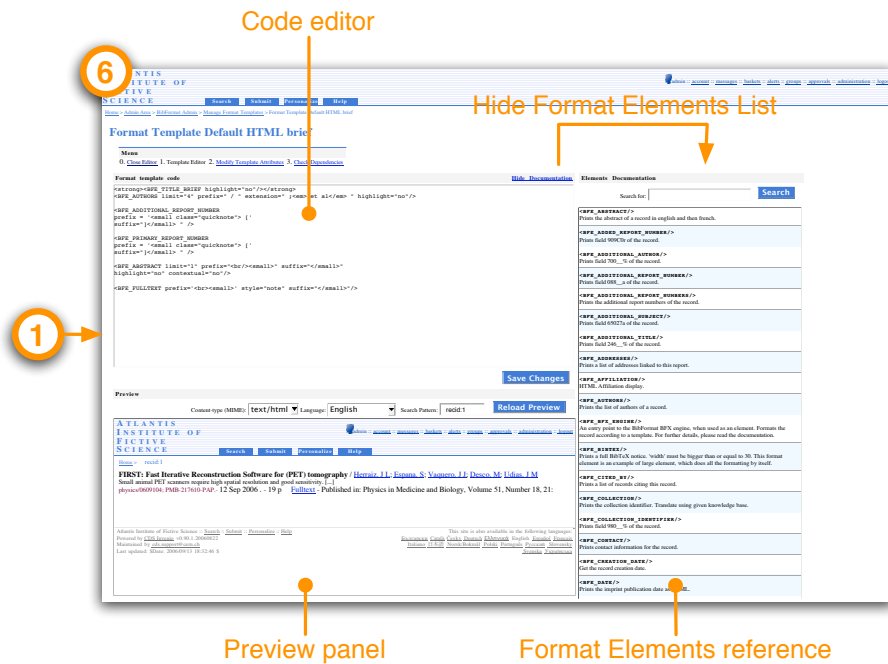


Figure 3.11: Format template editor

by a code editor and a preview panel. Still the underlying design allows the implementation to be upgraded to the initial specifications in a later release of the module. The editor also contains a list of format elements that can be included in the format template. It is now a more important element of the interface than in the prototypes, as it is an essential part of the edition of format. Users can search in the list of format elements by keywords, move their mouse over an element to see its description, and click on an element to include it in their code. This allows users to concentrate on the edition of the template without having to deal with another window containing the documentation of elements. Finally experienced users can close the format elements reference. The final version includes a toolbar to insert HTML tags into the code of the template, and get the documentation of the selected format element in the code. A menu (Figure 3.12) at the top of the editor allows to

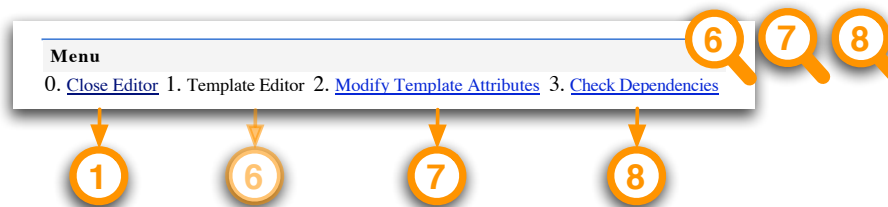


Figure 3.12: Top menu of the format template editor

- close the editor and go back to format templates list (go to figure 3.10).

3.3. FINAL UI DESIGN

- edit the format template code (go to figure 3.11 from other parts of the interface).
- edit the attributes of the templates (go to figure 3.13).
- see the dependencies of the template on other objects (go to figure 3.13).

Similar menus are also available in the knowledge base editor and output format editor.

The edition of the attributes is done in a very basic interface (Figure 3.13).

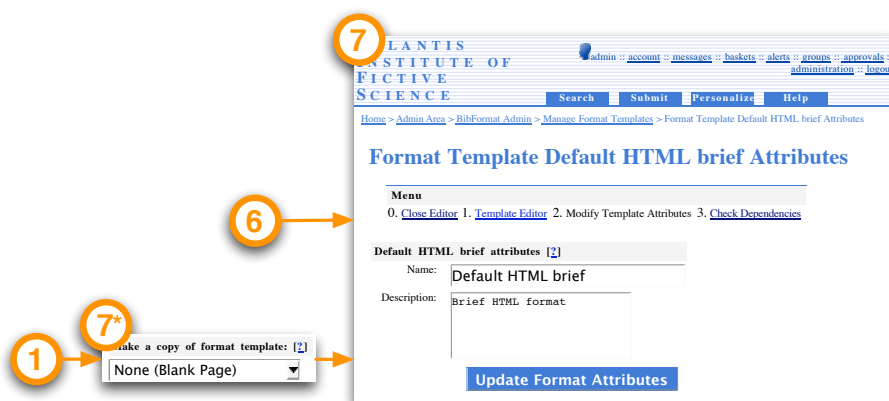


Figure 3.13: Attributes of a format template (Similar to output formats and knowledge bases attributes). When adding a new format template, a duplicate of an existing one can be created.

It is not going to be used very often, excepted for the creation of the format template, but it is essential to be able to set these attributes, in order to make the retrieval of format template possible. When creating a format template, this interface is shown first, and gives the possibility to make a duplicate of an existing format template.

Output formats and knowledge bases have a similar interface. The differences are that the interface for format template attributes allows to make a copy of an existing template at the creation of a format template while other interfaces do not, and that output format attributes contain more fields for the name of the output format, one for each language supported in CDS Invenio, in order to display that name in the main search interface of CDS Invenio.

The dependencies page (Figure 3.14) shows the list of usages of the format template in output formats, and the list of format elements and associated MARC tags used in the format template.

Finally a Dreamweaver floating panel was implemented as a proof of concept of the possibility to edit format template with HTML editors (Figure 3.15). It allows to insert format element in the HTML code and provide access to the format elements documentation. Inserted elements are shown as a blue brick marked as *bfe*.

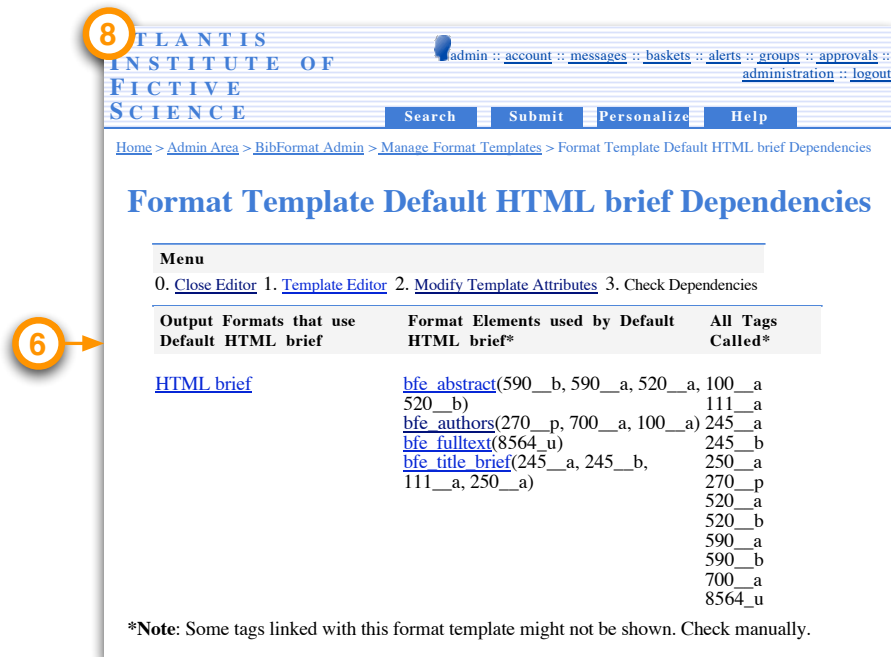


Figure 3.14: Dependencies of a format template (Similar to output formats and knowledge bases dependencies)

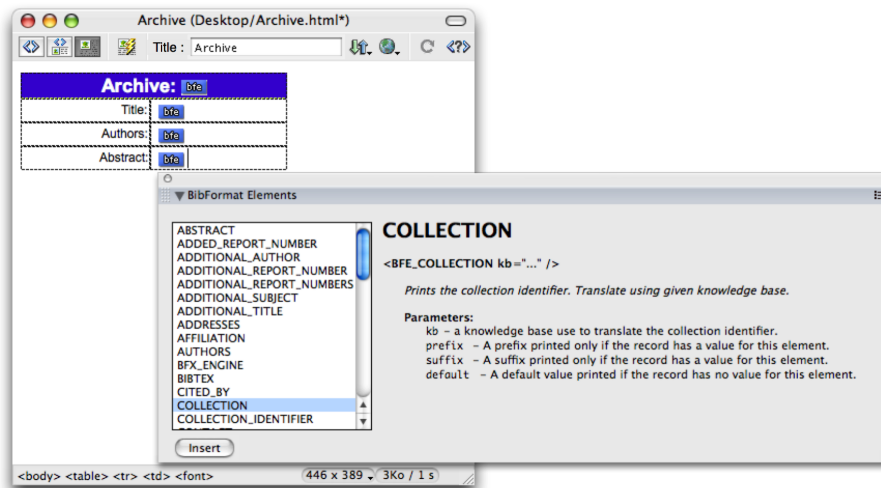


Figure 3.15: Integration of BibFormat within Dreamweaver

Output Formats related interfaces When users click on an output format in the list of output formats, the rules of the output format are displayed (Figure 3.16). They allow to specify which templates must be used when this output format is selected to format a record. Depending on user-specified conditions

3.3. FINAL UI DESIGN

on the values of the fields of the record, one of the template will be chosen for the formatting. Users can reorder the rules (rules are evaluated from top to bottom), add rules and delete rules.

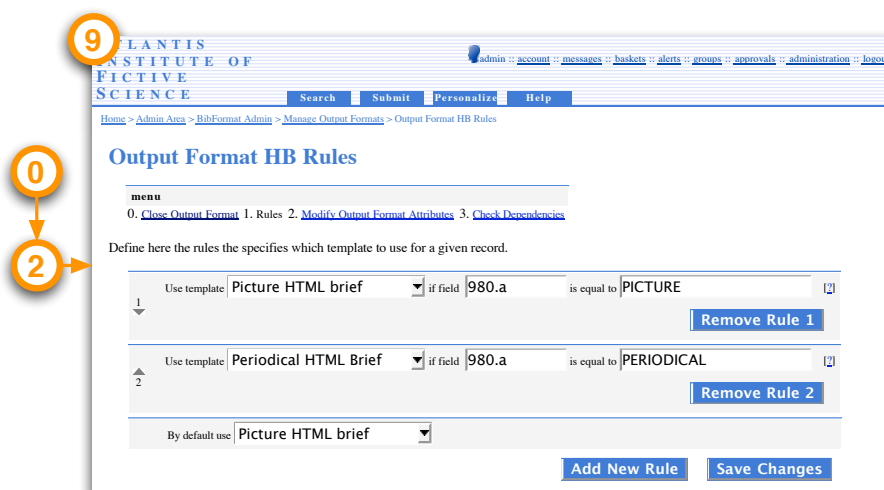


Figure 3.16: Output format rules

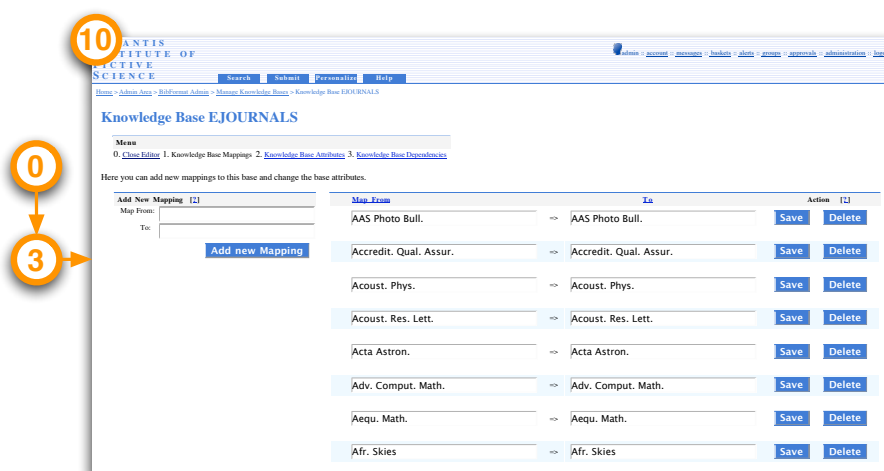


Figure 3.17: Knowledge base mappings

Knowledge Bases related interfaces A knowledge base defines mappings of values that need to be quickly and easily editable. Figure 3.18 shows the edition of a the mappings of knowledge base.

When a knowledge base is opened, the text insertion cursor is placed right into the *Map From* field such that users can directly start inserting value. The *tabulator* key let users enter the *To* value of the mapping, and *Enter* key adds the mapping to the list. Fields are erased and the insertion cursor is placed

back into the *Map From* field for a new addition. The mappings can be edited or deleted right from this page.

Format elements related interfaces Format elements are provided to users as the bricks to be used in their format templates, and are therefore not modifiable. Still users need to know what the elements are for. A dynamically generated reference documentation is not only available right from the template editor, but also from a dedicated web page that displays more information and offers more features than the short reference of the template editor. A sample notice of an element is shown in figure 3.18. Users can test a format element

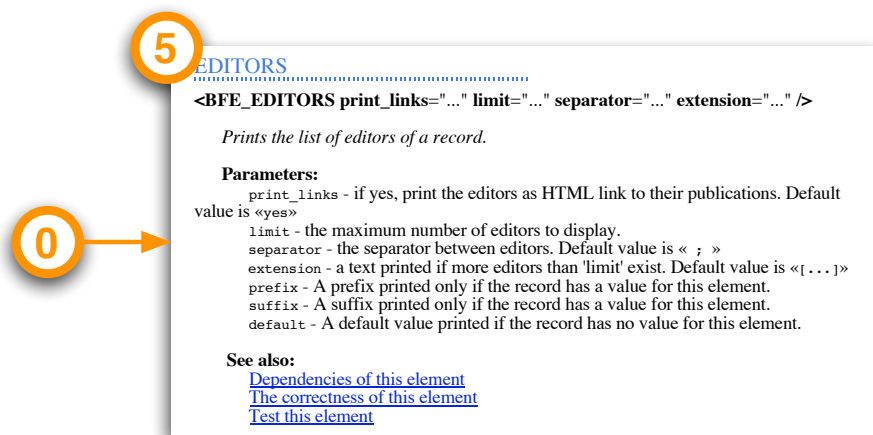


Figure 3.18: One format element notice of the format elements documentation

with custom parameters, see where the element is used (similar to other dependencies pages) and validate the code of the element (check that the format element has no error).

Migration related interface As stated in the specifications, the new BibFormat is not compatible with the configuration files of the old BibFormat. This means that all formats have to be rewritten. However a compatibility layer has been implemented such that during the transition to the new formats, customers can still use their old formats. To help them adopt the new system, a migration kit assistant has been made available (Figure 3.19). It can migrate the behaviors, knowledge bases and formats to the new system (Although it cannot translate formats into the new formatting language, it can help create the entries in the new BibFormat).

Steps (in suggested order)	Status
1. Migrate knowledge bases	Migrated
2. Migrate behaviours	Not Migrated
3. Migrate formats	Not Migrated

Figure 3.19: Migration kit steps.

3.4 Formatting Engine Design

Along with the user interface, a new formatting engine had to be designed. In order to support the different configuration levels (Output formats, format templates and format elements) suitable for our user population, a layered system has been architected (Figure 3.20).

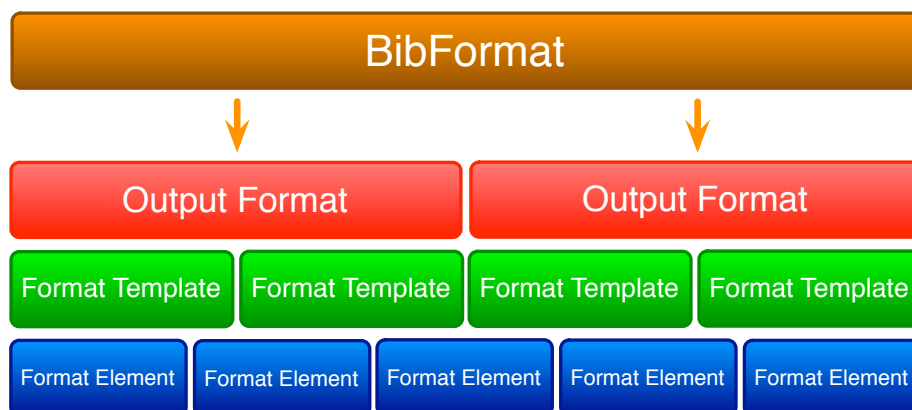


Figure 3.20: BibFormat layered architecture

Output formats can be considered as the entry point of the BibFormat workflow. Whenever a record has to be formatted, BibFormat is at least given a record ID (to fetch record meta-data from the database) and an output format short identifier code. Output formats simply define which format template has to be used for formatting the given record. Output formats are configured either from the user interface presented in section 3.3, or by editing directly the code of the output format. A sample output format code is shown below:

```

tag 980.a :
PICTURE --- Picture_HTML_detailed.bft
PERIODICAL --- Periodical_HTML_detailed.bft

default: Default_HTML_detailed.bft
  
```

In this example, if tag 980.a is equal to PICTURE (ignoring case), then the `Picture_HTML_Detailed` template will be used. Else if the same field is equal to PERIODICAL then `Periodical_HTML_Detailed` template will be applied. In all other cases, the default case will apply.

The syntax is the following one: for each field of the record on which we need to put a condition, write

```
tag marc_tag:↔
```

and make it followed by one or more lines with the value of the condition and the format template that must be applied if the condition is true, separated by three dashes.

```
value --- format_template_filename.bft↔
```

Note that the value can be expressed as a regular expression. In the same way as for the GUI administration, the conditions are evaluated from top to bottom, until a condition is found to be true for the current record or the default template is reached. Many conditions can be put on one field, and many fields can be declared.

Once a format template has been chosen, BibFormat has to process it. The format template contains the code that defines how the record will look like. It contains static and dynamic parts: static parts do not change according to the record that is being formatted and is outputted as such, while dynamic parts need to be interpreted by BibFormat to match the current record value. In most of the cases, static and dynamic elements of the code are HTML code, as the output need to be displayed in a browser. However static parts could be anything: the rule for static code is *What You Write Is What You Get*. Concerning dynamic code, it is always written with an HTML syntax. For example to output the title of the record, one needs to write `<BFE_TITLE />`. The text will be replaced with the title value of the record being formatted. Other such format elements are available. They can take special parameters as input: for example one could write `<BFE_TITLE prefix="Title: " />` in order to prefix the title with a label "Title: ". Different attributes can be configured depending on the format element.

Format templates contains other kinds of dynamic elements: the localized text. When a text need to be translated in various languages, a `<lang>` tag has to be used to enclose the different translations of the same text. Each translation is then enclosed with its language code tag: `<en>`, `<de>`, `<it>`, etc. For example, the localized version of "title" in English, German and Italian would be written:

```
<lang> <en>title</en> <de>Titel</de> <it>titolo</it> </lang>
```

The BibFormat engine will filter the localized version and only keep the language relevant to the formatting.

The code below shows a sample format template:

```
<h1><BFE_TITLE /></h1>
<p align="center">
  <BFE_AUTHORS/ limit="45" suffix="<br/>">
  <BFE_DATE suffix="<br/>" />
  <BFE_PUBLISHER suffix="<br/>" />
  <BFE_PLACE suffix="<br/>" />
  <lang><en>Abstract:</en>
    <fr>Résumé:</fr>
  </lang>
  <BFE_ABSTRACT />
</p>
<small>
  <BFE_FULLTEXT />
  <BFE_CITED_BY />
</small>
```

Format elements used in format templates are basically bindings to the database, with some post-processing capabilities. They are provided by programmers to

3.4. FORMATTING ENGINE DESIGN

users as black boxes that print values of a record. They cannot be modified through the user interface.

A format element is a small Python script. The script has to implement a function named `format`, that takes a mandatory parameter `bfo`. This parameter is an object that provides accessors to the context in which the formatting occurs, such as the record or the preferred language of the user. The function can also take additional parameters, which can be used to pass values to the format element from the format template. For example a format element which outputs the list of authors of a record can take the `limit` parameter to limit the number of authors printed by the element. One would write `<BFE_AUTHORS limit="5" />` in the format template to call this element with a limit of authors set to 5. The function must return a textual value, that will be printed in the format template where the format element is called.

A strong emphasis is put on the documentation of format elements, such the format elements layer of the BibFormat architecture is easily accessible to users of the format template layer. A *Javadoc* syntax in the *docstring* of the format function is used to define the description of the element, describe the parameters, and refers to other related format elements. This allows a documentation to be generated for the element, as shown in figure 3.18.

A sample format element is shown below:

```
def format(bfo, limit='10', separator=", "):
    """
    Prints the list of authors of the record.

    @param limit The maximum number of authors to print
    @param separator A separator between authors
    """
    authors = []
    authors_1 = bfo.fields('100--a ')
    authors_2 = bfo.fields('700--a ')

    authors.extend(authors_1)
    authors.extend(authors_2)

    nb_authors = len(authors)

    if limit.isdigit() and nb_authors > int(limit):
        return separator.join(authors[:int(limit)])
    else:
        return separator.join(authors)
```

The name of the Python script file used for the element corresponds to the name of the format element.

In summary the workflow of the BibFormat engine is shown in figure 3.21. One exception to this workflow is the case when BibFormat is given an unknown output format or format template: in that case, BibFormat will hand on the formatting to the old PHP formatting module. This allows customers of CDS Invenio to transition smoothly to the new BibFormat, as they can use all

of their old formats with the new module, and progressively replace them with the new formats.

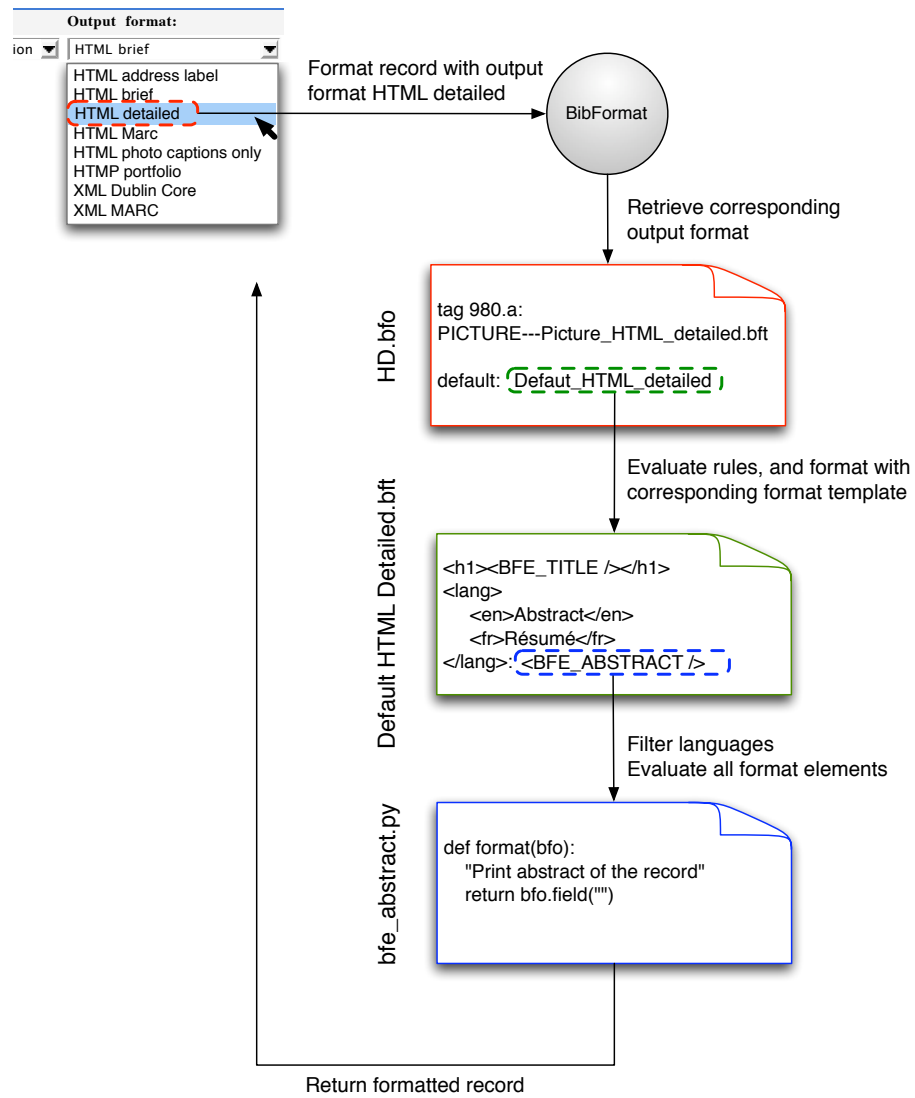


Figure 3.21: BibFormat workflow

The concept model of the BibFormat engine is shown in figure 3.22.

3.4. FORMATTING ENGINE DESIGN

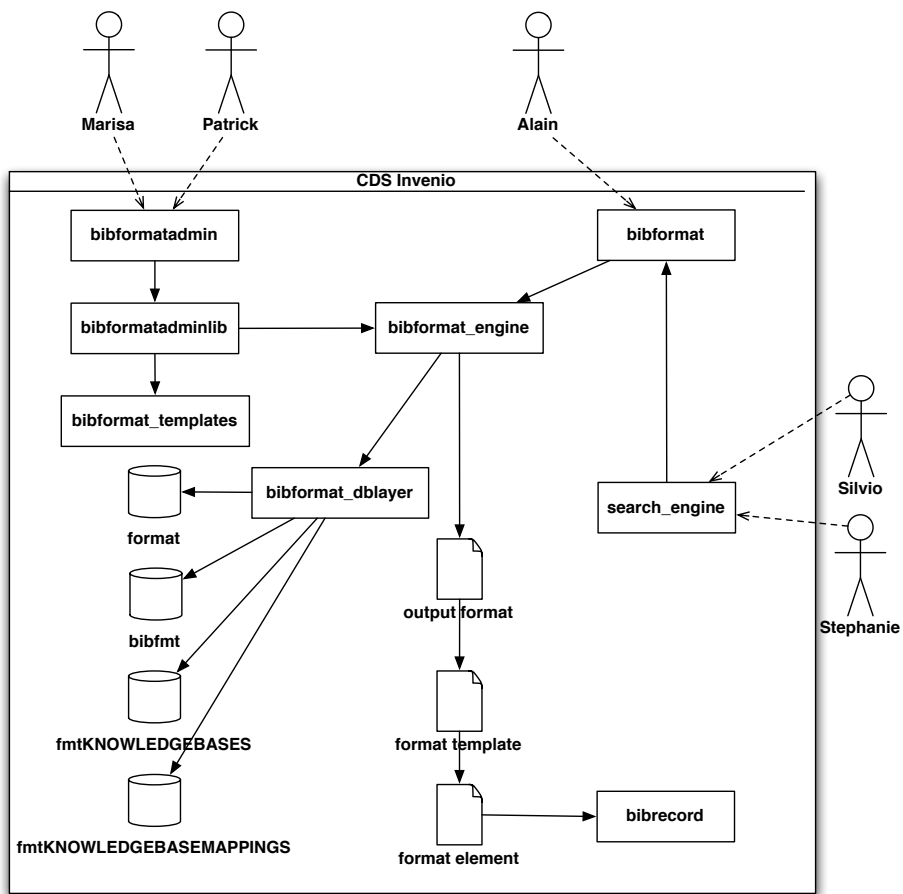


Figure 3.22: Concept model of BibFormat engine

Chapter 4

Evaluation

4.1 User Evaluation of the BibFormat Admin User Interface

The evaluation of the new BibFormat user interface was done informally with some of the target users. The person in charge of the formats at CERN and three members of the CERN library were given a short introduction to the new BibFormat interface. Although this was not very helpful at discovering issues in the interaction model, it helped to find out that:

- The notion of dependencies between the output format, format templates and format elements was not always clear to users. It was primarily the name that users suggested to revise.
- The “close editor” menu item in the format template editor was not clear for users: the editor does not open in a different window and it was not clear that it would make them go back to the list of formats when clicking on this option.
- The initial idea concerning the edition of templates with a custom HTML editor was that users would get access to the format templates through a locally mounted network volume. Librarians expressed the need to be able to upload or download format templates via the web interface. An upload button could be added in the format template editors, as well as download button in front of each format.
- Librarians were confused regarding the knowledge base usage. Although they already know the concept and purpose of knowledge bases, and understood how to edit them, they were wondering when they would be used in the formatting process. This interaction issue might come from the fact that knowledge bases are primarily used by format elements, but are shown in the interface as a separate entity. It would maybe judicious to print out a reference to the format elements using the knowledge bases right in the knowledge bases list.

Another evaluation of BibFormat was conducted more formally with someone totally unfamiliar with CDS Invenio. This person was only a little bit

familiar with HTML, and did not know MARC 21 at all. He was first shortly introduced to CDS Invenio and explained the goals of the evaluation. The participant then was given four tasks:

1. Modify how the formatting of search results for picture, such that picture is shown on the left instead of the right.
2. Create a totally new formatting for a new collection of article that has just been added, and display the title, abstract and authors (limited to 10 authors).
3. Assign the formatting created in last step to the new collection.
4. Normalize the abbreviated journal name *Phys Rev A* to *Phys. Rev. A*.

For the first task the participant first tried to modify the formatting using the output formats. Very quickly he found out that it was not the right place for editing the formatting, and by elimination chose the format templates options. It seems that the name was not adequate. Labels such as “Modify formatting” instead of “Format templates” and “Assign format to collection” would have better helped this user. Of course these labels were written just under the options, but the user did not read them. The task was then completed almost successfully. Only one problem occurred when the user saved his work: after clicking the end button, the participant expected to be brought back to the main menu, as he had finished his task.

The second task was more straightforward than the first one, as the participant had gotten familiar with the interface. The participant has gone through two minor problems. He tried to drag and drop a format element in the editor instead of clicking on it, but quickly figured out that clicking on the element was adding it to the code. He also did not immediately see that the Javascript toolbar could be expanded to offer more options for the insertion of HTML tags. This toolbar was important to him as he did not remember some HTML tags. Still the concept of adding elements and configuring their attributes was totally natural for the user.

During the third task, the only problem that occurred was that the user wanted to replace an existing assignment rule with the rule for the new collection, instead of creating a new rule. This is because the task was not clear to him. When told that he should not modify the existing rule, he immediately created a new rule for the new collection. This concept was clear to him excepted for the notion of MARC code.

The last task was completely successful. He did not asked any question for this task.

4.2 Usability Study of the Formats

A small online usability study of the default formatting of CDS Invenio was conducted at the end of the project, in order to reveal some usability issues with the current formats and give some ideas on their possible evolution. This study targets users different that the direct administrators users of BibFormat: it targets the before-mentioned readers users, who access CDS Invenio to find and read the documents.

4.2. USABILITY STUDY OF THE FORMATS

This study was not part of the initial project timeframe. Only a very short period of time was available to set this experiment up, such that it does not comply with a strict and scientifically accurate procedure. However it nicely complements this analysis of the formatting in CDS Invenio.

4.2.1 Goals of the Study

Users were asked to compare two kinds of formatting for the search results. The first one was the regular formatting of CDS Invenio (Figure 4.1(a)), modified to show two lines of the abstract instead of a single one. The second formatting (Figure 4.1(b)) was a modified version of the first one with the following hypothetical improvements:

- Sans-serif font: given as the CSS stylesheet of CDS Invenio does not force the type of the font and as most browsers are by default configured to use serif font, most users were seeing the results displayed in a serif font that could be difficult to read. The CSS stylesheet has been modified to use a sans-serif font.
- Clickable title: when I used CDS Invenio for the first time, I personally did not find out how to see the details of the record. It seems that persons who have been using popular web search engines are willing to click on the title to see the full record.
- Highlighted keywords: words used as search query are highlighted in the results.
- Contextual abstract: although the first lines of the abstract might be the most relevant lines to display for each record of the search results, an attempt has been done to extract a line that matches the most the keywords of the search query.


Physics at the front-end of a neutrino factory : a quantitative appraisal / [Mangano, M L et al](#) [CERN-TH-2001-131] [hep-ph/0105155]
We present a quantitative appraisal of the physics potential for neutrino experiments at the front-end of a muon storage ring. We estimate the foreseeable accuracy in the determination of several interesting observables, and explore the consequences of these measurements[...]
<http://documents.cern.ch/cgi-bin/setlink?base=preprint&categ=hep-ph&id=0105155>
[Detailed record](#) - [Similar records](#)

(a) Original search results

Highlighted search keywords Title links to detailed record

Physics at the front-end of a **neutrino** factory : a quantitative appraisal / [Mangano, M L et al](#) [CERN-TH-2001-131] [hep-ph/0105155]
We present a quantitative appraisal of the physics potential for **neutrino** experiments at the front-end of a muon storage ring. We estimate the foreseeable accuracy in the determination of several interesting observables, and explore the consequences of these measurements[...]
<http://documents.cern.ch/cgi-bin/setlink?base=preprint&categ=hep-ph&id=0105155>
[Detailed record](#) - [Similar records](#)

Sans-serif font Abstract contextual to search keywords



(b) New search results

Figure 4.1: Search results formatting

The contextual abstract (which is equal to non-contextual one in figure 4.1(b)) is implemented using a very basic technique: each sentence is given a weight

corresponding to the number of keywords it contains. Then each consecutive group of n lines (n corresponding to the number of lines to display for the abstract, 2 in our case) in the abstract is given the sum of the weights of the lines as weight. The group weighting the more is returned.

The goal of the study was to check that users preferred the second kind of formatting and that it allowed them to search more efficiently.

4.2.2 Experimental Conditions

The experiment took place on website only accessible from within CERN. Users connecting to the website were introduced to the experiment, asked some personal information and given the task to find

- a) either a document that matches their current research
- b) or a document that discusses the possibility of violation of traditional physics theory due to superconductors

For this first part they were given access to the regular CDS Invenio search engine, with search results formatted as in figure 4.1(a). Once they had found a matching document users were asked their feeling about the user interface. Users had then to perform a second task using a modified version of CDS Invenio that implemented the formatting shown in figure 4.1(b). They had to find

- a) either a document that matches the most the kind of research they were doing when they graduated
- b) or a course on history of particles

As for the first interface, they were asked their feeling about this second interface. Then they were asked to compare both interfaces.

None of the fields were mandatory and users could skip any part of the experiment with the “Skip” button located at the top of all pages.

The records set was made of about one thousand records imported from the CERN documents server, taken from all the most recent additions. This means that most of the documents were physics-related, although many IT-related documents were also available. Users were asked to find documents related to their research, but could also find one of the suggested documents (The documents were present in the documents set). Users could not just copy-paste the description of the documents to find them.

The time to complete each task was recorded, as well as the preferences of users, comments, their knowledge of search engines and the documents they have finally found. The experiment was also recording if users had tried to click on the title of search results in the first interface, even if it was not marked as a title.

Users were asked closed and open questions, with a majority of closed questions. They were also asked to grade some assertions, like their computer knowledge, using a 1 to 4 stars scale.

4.2. USABILITY STUDY OF THE FORMATS

The experiment was advertised during a presentation of BibFormat at a User and Document Services group meeting. Some flyers were also put in front of the CERN library. It was however not possible to send an email to all CERN users.

Given the time allocated for the implementation of the system and the low number of participants expected, the order of the task was the same for all users: they were always first shown the old formatting before the new formatting, and always had to find the same documents. This has prevented to check that the results were independent of these variables.

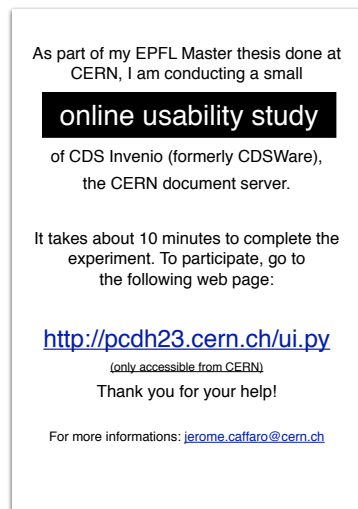


Figure 4.2: Flyers advertising the experiment

4.2.3 Results

As expected, due to the reasons mentioned above, very few people managed to participate to the study during the few days it was available. A dozen of persons have connected to the experiment, but only 8 of them have completely filled in the questionnaires. The results discussed in this section should therefore be taken with precaution, and not treated as scientifically accurate.

Users of the experiment were mostly IT workers and librarians. In average they estimated their computer knowledge as good (about 3 stars out of 4 for IT staff and 2 stars for others). They have all been using web search engines, and regularly use CDS Invenio.

During the first part of the experiment, users rated the relevance of the abstract with the first interface at 2.4 stars in average, and only 28% found that they took too much time to complete their task (5 minutes 28 seconds in average). There were no special comments, excepted one user who could not find a document related to her research, and who was not good enough in physics to search for the suggested document.

Only one user clicked on the (hidden) title of a records to display the details.

The relevance of the abstract with the second interface was rated 2.5 stars in average, and 25% estimated that it took too much time to find the document (3 minutes and 42 seconds). Comments suggested to decrease the importance given to the title (which is bold, highlighted and underlined in the new format) and to make the highlighting optional.

2/3 of users clicked on the title link to display the detailed view of records

When asked to choose their preferred interface, only 1 person chose the first one (certainly because of the highlighting which he found to be annoying). The others were in favor of the second kind of formatting. When asked about the font used, half of the users preferred the second one, while the others said they had no preference for one font over the other. Same results for the abstract, that only half of the user found to be more pertinent in the second case.

Although the numbers seem to show that users were more efficient at searching with the second kind of formatting that with the first one, it could be due to the fact that:

- the second document was easier to find than the first one
- users had become familiar with the set of document in the first interface, such that they could more easily find in the second one

Also when asked, participants did not feel that they were more efficient in the second interface.

The highlighting also seemed controversial. As suggested it could be turned off by default, and enabled by the user when needed. A more discreet style could also be studied, such as bolded text for keywords highlighting. This technique could not be applied to highlight the title, as title is already bolded in order to make it distinct from the authors list.

The linked titles seemed to be very much appreciated by users. The only problem is that it makes the title look much bigger, as it is now underlined and colored in blue. One solution would be to attach a style that would make it look like the first formatting. It could still be clickable by users, and even switch to the regular link appearance when the mouse would roll over the title.

It is difficult to state if the contextual abstract was more relevant than the first one: users graded the relevances almost as equal after each task, but when asked to directly compare the abstracts, users preferred the second one. This might come from a misunderstanding of the question which has made them evaluate the look of the abstract instead of the content of it.

4.3 Further Improvements of BibFormat

The small number of possible iterations for the analysis, design and implementation phases have let a lot of room for improving the formatting module.

4.3. FURTHER IMPROVEMENTS OF BIBFORMAT

Some refinements are desirable but not critical, and would mostly provide guidance to novice users. For example the studied WYSIWYG format template editor could be implemented to help beginners play with the edition of formats, but would not reach the expressiveness allowed with professional HTML editors such as Dreamweaver. Advanced users would also certainly prefer to edit the code directly.

Nevertheless a more task-based structure might still be considered. An implementation of the prototype shown in figure 3.4(b) (Section 3.2) for the allocation of formats to records would tend toward such a solution. While this could simply mean providing an alternative view to output formats, it could also lead to the abandon of output formats. A more complete analysis should be undertaken to see if this is a viable solution, especially regarding the loss of possibilities compared to the current implementation.

Of course results of the user evaluation should be taken into account to refine BibFormat. The most important tasks will be to:

- Add a way to download/upload formats through the web interface
- Rename some confusing labels
- Implement a resizable/customizable set of panels for the format template editor

The new possibilities offered by BibFormat should finally help make better formats, given that it is now possible to focus on the design instead of spending time on the implementation. This is where the most important improvements can be done, as formats have a direct impact on what end users will see. For example a new detailed format that lets users show or hide the list of authors has been implemented as a proof of concept of the modifications that could be done. This particular feature was extremely important, as CERN publications might be co-authored by more that 2000 researchers. The new BibFormat also features context-aware formatting: language, keywords of the search query, etc. can be taken into account to format records. Now that BibFormat is much faster than before, these advanced formatting techniques become possible and should be further investigated.

Chapter 5

Conclusions

In this report we have focused on the user-centered design of the new BibFormat. We have detailed the task analysis in section 2.1, and made a comparative analysis of related products in section 2.2.

The resulting specifications and corresponding prototypes have been described in chapter 3, along with a presentation of the implemented UI design and module architecture.

The results of the evaluation of the administration interface of the product have been given in section 4.1. The results of small usability of the formats have been discussed in section 4.2. Finally improvements to the new modules have been suggested in section 4.3.

This project resulted in a completely new implementation of the BibFormat module, which now ships as part of a new CDS Invenio release. The module has gone through a testing phase that has made it robust and fast. It can run along the previous version of BibFormat to let customers smoothly transition to the new module. All formats that were previously included by default in CDS Invenio have also been translated to the new version, such that most customers can easily adopt the new version.

Usability testing has shown that BibFormat is now much more accessible to novice users. The structure of the software has been designed to be simpler and to use less concepts. No documentation reading is needed to use the system, and users can play with it and try to make their own formats out of the box. Novice users can use any other HTML editor to edit format templates. Intermediate users can go further by editing the sources of format templates, and modify output formats. Advanced users can add new format elements, and modify all the configuration files with their preferred text editor.

Even though no reading is necessary to use BibFormat, an extensive documentation has been written. It includes a tutorial, a short explanation of how BibFormat works, and a manual that details every aspect of the software (which is also used for contextual help).

Although the new BibFormat features less concepts than the old one, it provides at least an equivalent expressiveness. The separation between the presentation layer (format template) and the business logic layer (format element)

has made the edition of the formatting both easier and more powerful, given that the layout is made using standard HTML, and that format elements can use the full power of Python and its libraries to format records. All concepts of the old BibFormat that have been dropped in the new release can be done using format elements.

This revision of BibFormat has also been an occasion to implement all features that have been requested by our customers for a long time. This includes the possibility to internationalize formats, and custom output content-type, such as Excel output.

A good trade-off between analysis, design and implementation has been found such that all objectives could be fulfilled. Moreover some guidelines have been given for future improvements that should be easy to bring, now that the formatting engine has been implemented and integrated into CDS Invenio. These improvements includes small adjustments to the user interface to satisfy users' needs, but also more significant changes, such as a revised way to assign format templates to records, in order to better support novice users.

Still the new BibFormat has been welcomed by concerned people and has generated a lot of interest, especially among librarians whose support is a requirement for a software like CDS Invenio.

Acknowledgements

I first would like to thank all the persons who made this Master project at CERN possible: Dr. Pearl PU FALTINGS for supervising my thesis, Jean-Yves LE MEUR, for proposing open projects to EPFL students, and Marisa MARCIANO WYNN, coordinator of the internship program, for her support during the application process.

I also would like to express my gratitude to all CDS members, who quickly integrated me in the team, and gave me technical support in various areas. I am especially indebted to Tibor ŠIMKO, who spent hours discussing the technical aspects of BibFormat, and more generally for helping me understanding the architecture of CDS Invenio. I furthermore would like to thank Belinda CHAN KWOK CHEONG for showing me how she was working with the PHP BibFormat, telling me her needs for the new BibFormat, and reviewing the prototypes and UI designs.

Finally I want to thank all the people — CDS developers, librarians, family and friends — who reviewed this document, tested BibFormat and gave any kind of support during this internship.

Appendix A

UML Use Cases

The scope of the following scenarios is the BibFormat administration system, at user goal-level.

Format templates related scenarios

Use Case: Create a new format template

Primary Actor: Patrick

Intention in Context: The user wants to create a new format template, either from scratch or copied from another one.

Main success scenario:

1. Patrick goes to BibFormat web interface, and opens the list of format templates
2. He adds a new format to the list, and enter the attributes of the format (name, description)
3. He adds static elements to the format (some labels, tables, etc.)
4. He dynamic elements (Title, authors, abstract, etc.) to the format from a list of elements.
5. He layouts the elements on the page (Align, reorder, etc.)
6. He modifies the styles of the elements (color, size, etc)
7. He previews his format with differents records
8. He makes additional modification to match his needs
9. He saves the modification made to the format once he is satisfied

Extensions:

- 2a. Patrick creates a duplicate of an existing format.

Here we assume that modification of layout and style of the elements are done through direct manipulation, as done it text editors. Marisa has a similar scenario for the creation of formats, but she uses a tool that she already knows. Alain also uses his own tools, but as he is an advanced user, he prefers to write directly the code of a format instead of using a WYSIWYG editor.

Use Case: Create a new format template

Primary Actor: Marisa

Intention in Context: The user wants to create a new format template, either from scratch or copied from another one.

Main success scenario:

1. Marisa opens Dreamweaver, her preferred HTML editor
2. She creates a new file and save it it the format templates directory of CDS Invenio
3. She adds dynamic elements (Title, authors, abstract, etc.) and static elements to the format
4. She layouts the elements on the page (Align, reorder, etc.)
5. She modifies the styles of the elements (color, size, etc)
6. She previews her format by going to a special URL in her internet browser
7. She makes additional modification to match his needs
8. She saves the modifications made to the format once she is satisfied

Extensions:

- 2a. Marisa creates a duplicate of an existing format.

Alain's scenarios

Use Case: Create a new format template

Primary Actor: Alain

Intention in Context: The user wants to create a new format template, either from scratch or copied from another one.

Main success scenario:

1. Alain opens emacs and creates a new file in his preferred text editor
2. He writes the code of the template in the created file
 - (a) He uses HTML to design his template
 - (b) He can refers to the documentation of dynamic elements to know how to add fields of the record such as title, abstract, etc.
3. He saves the file in the format template directory of the CDS Invenio installation

Extensions:

- 1a. Alternatively Alain duplicates the file of an existing format, and open it in a text editor.

The edition of format templates is similar to their creation for all users. We show here the scenario of Patrick. Other users just open the file of the template to modify in their preferred editor.

Use Case: Edit an existing format template

Primary Actor: Patrick

Intention in Context: The user wants to edit an existing format template.

Main success scenario:

1. Patrick Patrick goes to BibFormat web interface, and opens the list of format templates

-
2. He finds the template he wants to modify and click on the edit button
 3. Patrick can make the modifications he wants

Use Case: Preview a format

Primary Actor: Alain

Intention in Context: The user wants to preview the output produced by a format template with real records

Main success scenario:

1. Alain opens his web browser
2. He goes to a special URL in which the name of the format template to preview is specified
3. The web page shows the preview of the format template.

Use Case: Check the validity/correctness of formats:

Primary Actor: Alain

Intention in Context: The user wants to check that formats have no error or do not call undefined elements

Main success scenario:

1. Alain opens his terminal and type “`bibformat -validate`” to get the status regarding correctness of formats.
2. The terminal outputs “Ok” or a list of issues and the line numbers in the format files where the problems occur .

Extensions:

- 1a. Alternatively Alain goes to the list of format templates in BibFormat administration page and see if each of the formats marked as “ok”.

Use Case: Check the dependencies of a format

Primary Actor: Alain

Intention in Context: The user wants to check which MARC fields a format template uses, and in which output format it is being used.

Main success scenario:

1. Alain opens his terminal and type “`bibformat -dep format_name`” to get the dependencies of the format.
2. The terminal output a list of MARC field used by this format and in which case this format is used.

Extensions:

- 1a. Alternatively Alain goes to the list of format templates in BibFormat administration page, click on a format and choose “Check dependencies” option

The modification of the name and description of templates are straightforward scenarios not detailed here. The deletion of a template is as simple as clicking a delete button and confirm in the case of Patrick, and remove files from the format templates directory in the case of other users.

Output format related scenarios Marisa should not have to modify output formats. Nevertheless the scenarios of Patrick should also apply to Marisa.

Use Case: Create a new output format

Primary Actor: Patrick

Intention in Context: The user wants to create a new output format, and set the rules that will define which template is used when this output format is called for a record

Main success scenario:

1. Patrick goes to BibFormat web interface, and opens the list of output formats
2. He adds a new output format to the list, and enter the attributes of the format (name, description, content-type, short code identifier) and confirms
3. He can set the default template that will be called as last option
4. He can then add rules and edit them as he wants
5. He saves the modification made to the output format once he is satisfied

Use Case: Create a new output format

Primary Actor: Alain

Intention in Context: The user wants to create a new output format, and set the rules that will define which template is used when this output format is called for a record. He uses his own editor.

Main success scenario:

1. Alain opens his preferred text editor and creates a new file
2. He saves the file in the output format directory of CDS Invenio
3. He write the code of the output format
4. He saves the modification made to the output format once he is satisfied

Use Case: Edit an output format

Primary Actor: Patrick

Intention in Context: The user wants to edit an existing output format, and sets the rules that will define in which condition a format template is used for that output format

Main success scenario:

1. Patrick goes to BibFormat web interface, and opens the list of output formats
2. He opens the output format he wants to modify
3. He can add rules to the set of rules, and reorder them
4. For each rule he defines which template is used, and which condition on which MARC field of the record must be valid to use the template
5. He can set the default template when no rule applies for a record
6. He saves the modification made to the output format once he is satisfied

Alain can also open an output format in his preferred editor and modify it. Other scenarios not detailed here allow to check the validity of an output format, check its dependencies (which templates it uses), delete it or modify its attributes.

Appendix B

BibFormat Developers APIs

The APIs of `bibformat.py` consists in these functions:

```
def format_record(recID, of, ln=cdslang, verbose=0,
                 search_pattern=None, xml_record=None, uid=None,
                 on_the_fly=False):
```

```
    """
```

Formats a record given its ID (or its XML representation) and an output format.

Returns a formatted version of the record in the specified language, with pattern context, and specified output format. The function will define by itself which format template must be applied.

Parameters that allow contextual formatting (like 'search_pattern' and 'uid') are useful only when doing on-the-fly formatting, or when caching with care (e.g. caching all formatted versions of a record for each possible 'ln').

The arguments are as follows:

recID – the ID of the record to format. If ID does not exist the function returns empty string or an error string, depending on level of verbosity. If 'xml_record' parameter is specified, 'recID' is ignored

of – an output format code. If 'of' does not exist as code in output format, the function returns empty string or an error string, depending on level of verbosity. ;of' is case insensitive.

ln – the language to use to format the record. If 'ln' is an unknown language, or translation does not exist, default cdslang language will be applied whenever possible. Allows contextual formatting.

verbose – the level of verbosity in case of errors/warnings

0 – Silent mode
 5 – Prints only errors
 9 – Prints errors and warnings

search_pattern – the pattern used as search query when asked to format this record (User request in web interface). Allows contextual formatting.

xml_record – an XML string representation of the record to format. If it is specified, *recID* parameter is ignored. The XML must be pasable by *BibRecord*.

uid – User ID of the user who will view the formatted record. Useful to grant access to special functions on a page depending on user's privilege. Allows contextual formatting. Typically 'uid' is retrieved with *webuser.getUserId(req)*.

on_the_fly – if False, try to return an already preformatted version of the record in the database.

"""

Example:

```
>> from invenio.bibformat import format_record
>> format_record(5, "hb", "fr")
```

```
def format_records(recIDs, of, ln=cdslang, verbose=0, search_pattern=None,
                  xml_records=None, uid=None, record_prefix=None,
                  record_separator=None, record_suffix=None,
                  prologue="", epilogue="", req=None, on_the_fly=False):
```

"""

Returns a list of formatted records given by a list of record IDs or a list of records as xml.

Adds a prefix before each record, a suffix after each record, plus a separator between records.

Also add optional prologue and epilogue to the complete formatted list.

You can either specify a list of record IDs to format, or a list of xml records, but not both (if both are specified *recIDs* is ignored).

'*record_separator*' is a function that returns a string as separator between records. The function must take an integer as unique parameter, which is the index in *recIDs* (or *xml_records*) of the record that has just been formatted. For example *separator(i)* must return the separator between *recID[i]* and *recID[i+1]*. Alternatively *separator* can be a single string, which will be used to separate all formatted records. The same applies to '*record_prefix*' and '*record_suffix*'.

'*req*' is an optional parameter on which the result of the function are printed lively (prints records after records) if it is given. Note that you should set '*req*' *content-type* by yourself, and send http header before calling this function as it will not do it.

This function takes the same parameters as 'format_record' except for:

- recIDs* – a list of record IDs to format
- xml_records* – a list of xml string representations of the records to format. If this list is specified, 'recIDs' is ignored.
- record_prefix* – a string or a function the takes the index of the record in 'recIDs' or 'xml_records' for which the function must return a string.
Printed before each formatted record.
- record_separator* – either a string or a function that returns string to separate formatted records. The function takes the index of the record in 'recIDs' or 'xml_records' that is being formatted.
- record_prefix* – a string or a function the takes the index of the record in 'recIDs' or 'xml_records' for which the function must return a string.
Printed after each formatted record
- req* – an optional request object on which formatted records can be printed (for "live" output)
- prologue* – a string printed before all formatted records string
- epilogue* – a string printed after all formatted records string
- on_the_fly* – if False, try to return an already preformatted version of the records in the database

```
"""
def get_output_format_content_type(of):
    """
    Returns the content type (eg. 'text/html' or 'application/ms-excel') \
    of the given output format.

    The function takes this mandatory parameter:

    of – the code of output format for which we want to get the content type
    """

def record_get_xml(recID, format='xm', decompress=zlib.decompress):
    """
    Returns an XML string of the record given by recID.

    The function builds the XML directly from the database,
    without using the standard formatting process.

    'format' allows to define the flavour of XML:

```

- 'xm' for standard XML
- 'marcxml' for MARCXML
- 'oai_dc' for OAI Dublin Core
- 'xd' for XML Dublin Core

If record does not exist, returns empty string.

The function takes the following parameters:

recID – the id of the record to retrieve

format – the XML flavor in which we want to get the record

decompress – a function used to decompress the record from the database
"""

The API of the BibFormat Object ('bfo'), given as a parameter to the `format` function of format elements, consists in the following functions. This API is to be used only inside format elements.

def control_field(self, tag):

"""

Returns the value of control field given by tag in record.

If the value does not exist, returns empty string

The returned value is always a string.

The argument is:

tag – the marc code of a field

"""

def field(self, tag):

"""

Returns the value of the field corresponding to tag in the current record.

If the value does not exist, returns empty string

The returned value is always a string.

The argument is:

tag – the marc code of a field

"""

def fields(self, tag):

"""

Returns the list of values corresponding to "tag".

If tag has an undefined subcode (such as 999C5), the function returns a list of dictionaries, whose keys are the subcodes and the values are the values of tag.subcode.

*If the tag has a subcode, simply returns list of values corresponding to tag.
The returned value is always a list.*

The argument is:

tag – the marc code of a field

"""

```
def kb(self, kb, string, default=""):
```

"""

Returns the value of the "string" in the knowledge base "kb".

*If kb does not exist or string does not exist in kb,
returns 'default' string or empty string if not specified*

The arguments are as follows:

*kb – the knowledge base name in which we want to find the mapping.
If it does not exist the function returns the original
'string' parameter value. The name is case insensitive (Uses
the SQL 'LIKE' syntax to retrieve value).*

*string – the value for which we want to find a translation–
If it does not exist the function returns 'default' string.
The string is case insensitive (Uses the SQL 'LIKE' syntax
to retrieve value).*

default – a default value returned if 'string' not found in 'kb'.

"""

```
def get_record(self):
```

"""

Returns the record encapsulated in bfo as a BibRecord structure.

You can get full access to the record through bibrecord.py functions.

"""

Example (from inside BibFormat element):

```
>> bfo.field("520.a")
>> 'We present a quantitative appraisal of the physics potential
    for neutrino experiments.'
>>
>> bfo.control_field("001")
>> '12'
>>
>> bfo.fields("700.a")
>> ['Alekhin, S I', 'Anselmino, M', 'Ball, R D', 'Boglione, M']
>>
>> bfo.kb("DBCOLLID2COLL", "ARTICLE")
>> 'Published Article'
>>
>> bfo.kb("DBCOLLID2COLL", "not in kb", "My Value")
```

```
>> 'My Value '
```

Moreover you can have access to the language requested for the formatting, the search pattern used by the user in the web interface and the userID by directly getting the attribute from 'bfo':

```
bfo.ln  
"""
```

```
Returns the language that was asked to be used for the  
formatting. Always returns a string.  
"""
```

```
bfo.search_pattern  
"""
```

```
Returns the search pattern specified by the user when  
the record had to be formatted. Always returns a string.  
"""
```

```
bfo.uid  
"""
```

```
Returns the user ID of the user who shall view the formatted  
record.  
"""
```

Example (from inside BibFormat element):

```
>> bfo.ln  
>> 'en '  
>>  
>> bfo.search_pattern  
>> 'mangano and neutrino and factory '
```

Bibliography

- [1] CERN. *The world's largest physics laboratory*.
<http://www.cern.ch>, September 2006.
- [2] Tim Berners-Lee. *Information Management: A Proposal*.
<http://www.w3.org/History/1989/proposal.html>.
- [3] Alberto Pepe, Thomas Baron, Maja Gracco, Jean Yves Le Meur, Nicholas Robinson, Tibor Simko, and Martin Vesely. *CERN Document Server Software: the integrated digital library*.
<http://doc.cern.ch/archive/electronic/cern/preprints/open/open-2005-018.pdf>, Apr 2005.
- [4] *CDSweb, the CDSInvenio installation at CERN*.
<http://cdsweb.cern.ch/>.
- [5] *Open Archives Initiatives*.
<http://www.openarchives.org/>.
- [6] Library of Congress. *Marc 21 Concise Format for Bibliographic Data*.
<http://www.loc.gov/marc/bibliographic/>.
- [7] Gregory Favre. *Extending CDSware with social tools*.
<http://documents.cern.ch/archive/electronic/cern/others/itnote/it-note-2005-002.pdf>, 2005.
- [8] EPFL. *Infoscience*.
<http://infoscience.epfl.ch/>.
- [9] *Ex Libris Group*.
<http://www.exlibrisgroup.com/>.
- [10] CERN. *HR Statistics Home Page* [access restricted to cern].
<http://humanresources.web.cern.ch/humanresources/internal/general/HN-statistics/default.asp>.
- [11] EPFL. *Service académique, Statistiques* [french only].
<http://sac.epfl.ch/page9623.html>.
- [12] RERO. *Information Générales* [french only].
<http://www.rero.ch/page.php?section=infos&pageid=rero.info>.
- [13] MIT Libraries and Hewlett-Packard Company. *DSpace Federation*.
<http://www.dspace.org/>.

- [14] *EPrints*.
<http://www.eprints.org/>.
- [15] Cornell University Information Science and University of Virginia Library.
Fedora Project.
<http://www.fedora.info/>.
- [16] Frédéric Gobry. *Pybliographer*.
<http://pybliographer.org/>.
- [17] Sonny Software. *Bookends, reference and bibliography software*.
<http://www.sonnysoftware.com/>.
- [18] Alan Cooper and Robert M. Reimann. *About Face 2.0: The Essentials of Interaction Design*. Wiley, March 2003.
- [19] Kathy Baxter Catherine Courage. *Understanding Your Users: A Practical Guide to User Requirements Methods, Tools, and Techniques*. Morgan Kaufmann Publishers, November 2004.