

# **Pybliographer Development Guide**

Frédéric Gobry and Peter Schulte-Stracke

July 26, 2003

---

# Contents

<b>Preface</b>	<b>7</b>
0.1. Where to Find More Information . . . . .	7
0.1.1. Internet Sources . . . . .	7
 <b>I Overview</b>	 <b>9</b>
<b>1. Introduction</b>	<b>10</b>
1.1. The Evolution of Bibliographical Software . . . . .	10
1.1.1. Reference Managers . . . . .	11
1.1.2. Library Catalogues . . . . .	11
1.1.3. Records Management . . . . .	12
1.1.4. Internet Database & Document Access . . . . .	13
1.1.5. Integration with Document Preparation . . . . .	13
1.1.6. Summary of current trends . . . . .	14
1.2. Reference Documents . . . . .	14
 <b>2. Requirements Overview</b>	 <b>16</b>
2.1. Goals and Purposes . . . . .	16
2.2. Program Audience and Context . . . . .	17
2.2.1. Scenario: Write a Paper . . . . .	18
2.2.2. Scenario: Maintain a Bibliography . . . . .	18
2.3. Interfaces . . . . .	20
2.3.1. Actors and Activities . . . . .	20
2.3.2. External Interfaces . . . . .	20
2.3.3. Internal Interfaces . . . . .	20
2.4. Program Features . . . . .	20
2.5. Usecases . . . . .	21
2.6. Requirements Directory . . . . .	24
2.7. Objects and Classes – the problem domain . . . . .	25
2.7.1. Interfaces . . . . .	25
2.8. Other Requirements . . . . .	25
2.9. Constraints . . . . .	25
 <b>3. Architecture Overview</b>	 <b>26</b>
3.1. Application Core . . . . .	26
3.2. Architecure Baseline . . . . .	26
3.3. Implementation Plan . . . . .	26

<b>II</b>	<b>Design Considerations</b>	<b>27</b>
<b>4.</b>	<b>The Graphical User Interface</b>	<b>28</b>
4.1.	Menus . . . . .	28
4.1.1.	The Menu Bar . . . . .	28
4.1.2.	The Application Menu . . . . .	29
4.1.3.	The Database Menu . . . . .	29
4.1.4.	The Folder Menu . . . . .	29
4.1.5.	The Record Menu . . . . .	29
4.2.	Displaying and Using Indices . . . . .	29
4.2.1.	Marking items . . . . .	30
4.2.2.	Accelerator Key Assignments . . . . .	30
4.3.	Appendix: Accessibility Guidelines . . . . .	30
<b>5.</b>	<b>Organising and Manipulating References</b>	<b>32</b>
5.1.	Folders and Lists . . . . .	32
5.1.1.	Lists . . . . .	33
5.2.	Keywords . . . . .	33
5.3.	Subject Headings . . . . .	35
5.3.1.	Syntactic Structure . . . . .	36
5.3.2.	Thesaurus Structure . . . . .	36
5.4.	Classifications . . . . .	36
5.5.	Indexes . . . . .	37
<b>6.</b>	<b>Searching, Selecting, and Formatting Data</b>	<b>38</b>
6.1.	A generic query service . . . . .	38
6.2.	Integration with editors, wordprocessors, and browsers . . . . .	39
6.2.1.	Emacs . . . . .	40
6.2.2.	Other editors . . . . .	40
6.2.3.	Mozilla . . . . .	40
6.2.4.	Other browsers . . . . .	40
6.2.5.	Kword . . . . .	40
6.2.6.	Abiword . . . . .	40
6.2.7.	Open Office . . . . .	40
6.2.8.	Other word processors . . . . .	40
6.2.9.	Other applications . . . . .	40
6.3.	Formatting references and bibliographies . . . . .	40
6.4.	Other applications . . . . .	40
<b>7.</b>	<b>Interfacing to External Ressources</b>	<b>41</b>
7.1.	Connecting to external databases . . . . .	41
7.2.	Writing Import Filters . . . . .	41
7.2.1.	Framework Services . . . . .	42

7.2.2. Class parameters . . . . .	44
7.3. Writing Export Filters . . . . .	45
7.4. Pybliographer as Web Server . . . . .	45
7.5. Requesting remote operations . . . . .	45
7.6. Requesting documents . . . . .	45
7.7. Bibliographic Data: types, formats, schemata . . . . .	46
7.7.1. Bibliographical Objects . . . . .	46
7.7.2. Persons . . . . .	46
7.7.3. Works . . . . .	46
7.7.4. Subjects . . . . .	46
7.7.5. Database Organisation . . . . .	46
7.8. Scripting and Configuration . . . . .	47
7.9. Technical Questions . . . . .	48
7.9.1. Architectural issues . . . . .	48
7.9.2. Compatability issues . . . . .	48
7.9.3. Charset and markup issues . . . . .	48
<b>III Component Design</b>	<b>49</b>
<b>8. The Application Core</b>	<b>50</b>
<b>9. Common Control</b>	<b>51</b>
9.1. Description . . . . .	51
<b>10. The Storage Component</b>	<b>52</b>
10.1. Description . . . . .	52
10.2. Module Storage Overview . . . . .	52
<b>11. The Bibliographic Component</b>	<b>54</b>
11.1. Description . . . . .	54
<b>IV Appendices</b>	<b>55</b>
<b>A. Feature lists</b>	<b>56</b>
A.1. User Requirements . . . . .	56
A.2. From Software Reviews . . . . .	57
A.3. Biblioscape . . . . .	73

---

## List of Tables

2.1. Actors . . . . .	20
2.2. Mandatory Features . . . . .	21
A.1. User Requirements (Chemnitz) . . . . .	56
A.2. Pybliographer Deficiencies . . . . .	57
A.3. Database Structure Features . . . . .	57
A.4. Database building capabilities . . . . .	58
A.5. Database search and retrieval capabilities . . . . .	58
A.6. Database applications . . . . .	58
A.7. Dell' Orso: Generalities . . . . .	59
A.8. Dell' Orso: Technology Substrate . . . . .	60
A.9. Dell' Orso: Database and record structure . . . . .	61
A.10.Dell' Orso: Input/Edit . . . . .	62
A.11.Dell' Orso: Import . . . . .	63
A.12.Dell' Orso: Search . . . . .	64
A.13.Dell' Orso: Search <i>continued</i> . . . . .	65
A.14.Dell' Orso: Output and Print . . . . .	66
A.15.Dell' Orso: Formatting language to define output styles . . . . .	67
A.16.Dell' Orso: Sort . . . . .	68
A.17.Dell' Orso: Sort <i>continued</i> . . . . .	69
A.18.Dell' Orso: Export . . . . .	69
A.19.Dell' Orso: Manuscript formatting . . . . .	70
A.20.Dell' Orso: Manuscript formatting . . . . .	71
A.21.Dell' Orso: Term/Entry list, authority file . . . . .	72

---

## List of Figures

2.1. Use case diagram (overview) . . . . .	21
--	----

---

## Preface

This document serves as the base for development of the Pybliographer application. This document is intended to give the Pybliographer developers both an introduction to the application area and an reference view of the application as vision, guidance, and motivation. Accordingly, it is divided into three parts:

**Part 1, Overview** introduces the subject area, and lists major sources of reference information. Then it treats design goals, requested and desired features and functions; it gives an overview of the use cases that are to be considered and an overview of the architecture, and the further development.

**Part 2, Design Considerations** discusses major areas and problems of Pybliographer's design in order to give the development a better foundation.

**Part 3, Component Specifications** contains the specifications for all subsystems and packages.

---

### 0.1 Where to Find More Information

Further information can be found in:

- *Pybliographer User's Guide*, which contains explanations of the user interactions, data elements, and the application programming interface, and
- *Pybliographer Reference Manual*, which contains all diagrammatic and textual information grouped according to type.

---

#### Comments:

- Comments and corrections are always welcome. Please address your message to: `pybliographer-general@lists.sourceforge.net`.
- 

#### 0.1.1 Internet Sources

The following resources are available through the Internet:

- **Pybliographer home page**

You can visit the Pybliographer home page on the World Wide Web using this address: `http://pybliographer.sf.net/`.

- **Pybliographer discussion list**

You can also participate in the discussion on the Pybliographer general mailing list operated through Source Forge. See the following information page:  
<http://lists.sourceforge.net/mailman/listinfo/pybliographer-general>

To subscribe send a message to:

`pybliographer-general-request@lists.sourceforge.net` with a subject line set to Subject: `subscribe`.

To post a question or response on the Pybliographer general mailing list, send it to:

`pybliographer-general@lists.sourceforge.net`

Include an appropriate Subject: line.

See the list's archive at the following URL:

<http://sourceforge.net/mailarchive/forum.php?forum=pybliographer-general>.

- **Download area**

You can get source code and additional information through **anonymous ftp** from various places. Please refer to the information here: <http://canvas.gnome.org:65348/pybliographer/download.html>.

**Note:** Pybliographer is included in all major Linux distributions. In most cases, you will prefer to use the specially prepared package that your distribution offers. See *Pybliographer User's Guide* 3.1 on information how to obtain and install Pybliographer using the facilities of your distribution.

- **CVS access**

You can access the current stable and unstable code on **CVS** at `cvs.gnome.org/cvs/gnome/pybliographer`.

For information on joining Pybliographer development address yourself to the mailing list.



---

## Part I. Overview

---

# Chapter I. Introduction

Pybliographer provides Reference manager services to create, load, modify, list, read, transport, and copy descriptions of bibliographic resources like books, articles, and electronic documents, including derivative and private information. In addition it helps in accessing external databases and document repositories.

Today, scholars and writers need efficient, consistent, and less complex ways to work with bibliographic resources and to maintain their existing inventory of references. The trend in bibliography is to integrate and share data, and to develop applications that transcend traditional boundaries.

In the past, reference managers had only limited ability to adapt to the plenitude of working environments and to interact with other applications. This has often constrained those with special needs to work with inadequate programs, and burdened them with work-arounds and makeshift arrangements. Bibliographic software has on the whole tended to be idiosyncratic and limited in its outlook, or to be expensive, complicated and nevertheless constrained to a small subset of the requirements, as it is, e.g., the case with **MARC**-based library software (from the point of an individual writer/scholar, at least).

Pybliographer establishes a common base for many tasks. It combines essential services, such as database access, formatting and searching records, with a common vocabulary and structure. In making use of established practice and standards it avoids many of the problems that other programs have when it comes to extending and inter-operating. All of these services are available through a set of interfaces that are easy to adapt and a fertile foundation for extensions.

Many restrictions inherent in the BibTeX database format used by earlier versions of this program (and many competing programs) are removed or relaxed in this version.

This version continues to support BibTeX databases to assist you in converting to the new database structure. Subsequent releases, however, might not support these features, so we strongly recommend conversion to the exclusive use of the new database structure.

Support for using BibTeX as an import and export format, however, is planned for the foreseeable future.

---

## I.1 The Evolution of Bibliographical Software

As a bibliographic database manager, *Pybliographer* places itself at the intersection of various expectations, traditions, and requirements, each of which developed originally independent, and often in ignorance of the other, and which are still shaping the field according to their own peculiarities, although they are getting into closer contact recently. For orientation, it may be useful to give a short view of some of these ancestral

developments. They embody still important principles, and may serve as benchmarks for our endeavour.

### 1.1.1 Reference Managers

Very early the scholarly and technical document preparation has been supported with programs to augment document sources with correctly formatted references, of which *BibTeX* may be regarded as prime example. Those early applications are slanted towards the production of reference lists, almost at the exclusion of other uses, their storage formats are implicitly defined and easy to extend, and the building and maintaining of its data base is left to other means.

In spite of (or perhaps because of) these deficiencies these have been very successful programs. Their restrictions, that follow a well proved tradition after all, were intended and adequate at the time of their creation. They have been addressed by the next class of applications which developed, characteristically, in the highly competitive marketplace of the *personal computer*.

With the advent of the Personal Computer a new class of users came to the fore. While the majority of the scientific/technical writers (who have little choice, anyway) stayed with their *TeX* applications, the new users were mostly attracted to the perceived simplicity of the word processors, and eschewed the command line and simple editors of Unix. Almost by necessity, the new applications stressed those aspects of the job that the older tradition neglected. Word processor users didn't care for the typographical refinements of *TeX*, nor the frugality and intellectual self restriction of the command line. They wanted simplicity and ease of use, ... and over the time, they succeeded. The graphical user interface comes no longer as an afterthought, it is at the centre of the development effort (often to the detriment of the application proper).

The features of *Biblioscape*, a commercial product are given in appendix A.3. For us it is important to stress the *database* aspects and pay attention to the *organising* of the references, by providing suitable instruments, e.g., the *virtual folders* of *Biblioscape*.

### 1.1.2 Library Catalogues

In the beginning library catalogues were nothing but simple *inventories* and they and their counterparts in archives serve this purpose to this day. After they became accessible to the public, though, a number of problems arose that are still important, as are some of the solutions. Required by the inter-library loan services and other fields of cooperation, *cataloguing rules* were developed, culminating in the ISBD, which set a widely accepted standard. The actual rules have always aimed at a compromise between precision and cost.

## 1. Introduction

Over time, the descriptions tended to grow in size. There are several reasons for this: (i) the more items exist, the more difficult it is to distinguish between them, the more details are thus needed, (ii) if one tries to spread the cost over more parties, one has to accept information that individually one might not find worthwhile, and finally, (iii) the need for better pre-selection entails even more information, as, e.g., abstracts, to be given.<sup>1</sup>

It should be noted, that traditionally only books have been catalogued, together with other physical objects, that might land in a library (**MARC** being quite comprehensive in its scope of materials allowed). That reflects the point of view of the librarian, who is the custodian of these said objects, buys, lends, ranges them. The contents are not his business. In that his view differs from the usual patron's view, as exemplified by the BibTEX and reference manager software (see 1.1.1). This orientation is slowly changing under the influence of Internet resources and integration with patron's software (section 1.1.4).<sup>2</sup>

Perhaps the most important heritage from this development is the idea of *entry points*: try to establish for everything that you catalogue well-defined and precise enough attributes and make the database search-able for them.

For a card catalogue this implies defining not only under which *form* an author is filed, but in addition under which author a book is filed (if there is a choice). The first class of normalisations is done via *authority files* the importance of which, under the conditions of on-line catalogues, has only increased, because it is more difficult to find entries which differ slightly (4) while the so called question of the main entry has no longer the importance it used to have.

### 1.1.3 Records Management

Of course, the management of current administrative records (records and document management applications) is out of the scope of bibliographical software, mostly. But to a degree, it would be beneficial to be able to handle one's own documents with more ease and consistency by adding a records management feature to the reference manager.

If, as it is the case in particular in the United States, archival resources are integrated into library systems, the idea is not far to use the same software and the same set

---

<sup>1</sup>In the classical catalogues one could afford to be concise, as one could always go and fetch the copy for actual inspection.

<sup>2</sup>**MARC** intended, indeed, to support so called *analytical* cataloguing as well, if only as a secondary task. Even the description of archival resources is not, or so one thought, out of its scope. This reflects the facts that American libraries hold archival collections to a far greater degree than European ones, and that these are far simpler than the classical European archival fonds; but remarkably this proved to be a failure – a separate standard (EAD) evolved, this time based on XML. Perhaps the reason was the absence of any form of hierarchy in **MARC** databases.

of rules for both kinds of material. The use of **MARC** for archival resources failed, however, not because of any impossibility, but because of the cultural inability of using the in this case inevitable multi-level description technique. In Germany, archivists use database systems for data entry, in America, word processors – neither choice is fully adequate.

Extending the database schema to enable the description of *administrative records*, *personal papers*, and *manuscripts*, and collections thereof, should be taken into consideration.

### 1.1.4 Internet Database & Document Access

Although the Internet precedes the personal computer by some years, the impact of the Internet was but slowly felt. We can at this time distinguish the following issues:

**Database access** in particular with the intent of adding the results at least partially to one's database.

**Document access** in particular where the document's URL forms part of the bibliographical description.

**Electronic documents** continue to challenge the librarian's profession. Ephemeral, they were often felt not to warrant the expense of traditional cataloguing (out of which considerations the *Dublin Core* standard evolved), hierarchical, they stretched the card-bound structure of **MARC** to the limits, ever changing, they challenged the concepts of bibliographical unit, of work, and edition (see chapter ??).

It is still not clear how great the impact of electronic documents and collections will be. Recent developments include the Library of Congress' **MODS** projects, following the lead of the *Dublin Core* so-called *Metadata* standard. The idea is often to replace traditional cataloguing with less expensive alternatives, in the first place by placing information in the documents (in a deplorable concession to linguistic vanity) called *meta data*, from where it can then be *harvested* automatically. Unfortunately this is not easily accomplished and perhaps even paradoxical.

### 1.1.5 Integration with Document Preparation

**Note:**

- Usability issues and the changed workflow

### 1.1.6 Summary of current trends

**Note:**

- Where are the current projects heading?
- What is felt to be the most pressing development today?

## 1.2 Reference Documents

The following gives a list of articles and other reference sources that are helpful in understanding the issues.

**Note:**

- A more comprehensive list could perhaps be drawn up, in another place.

**Introduction** Elaine Svenonius 2000 gives a broad overview of the problems of bibliographic description, starting from objectives and principles and aiming at giving the field a sound foundation. Klaus Haller 1998, in a more limited way, does the same for a German professional public. These authors are only concerned with library catalogues, with Svenonius being more profound. In addition, she has a lot of references.

**Requirements** Dagmar Knorr examines in her thesis 1998 practical processes of academic writing with a particular regard towards the use of bibliographical information and reference manager tools. While she finds a very inhomogeneous landscape, she is correct in my view, in stressing the importance of adequate tools, and of activities antecedent to and supportive of actual writing, such as note taking, including task related notes, and tracing, e.g., of quoted literature. She documents an earlier period (early 1990s) where often very basic BibTeX features were not available, so this would be *relatively* even more important today. Another study of actual user needs has been done for the *Open Office* project Job-Sluder u. a. (9).

**Reviews** Of the many overviews of existing reference manager systems, the following have been particular helpful: (author?) (ors:bfs02)

**Cataloguing rules** All newer cataloguing rules are influenced by the work of the *International Federation of Library Associations*, which has at its website [www.ifla.org](http://www.ifla.org) a number of *International Standards for Bibliographical Description* ISBD available. In addition there can be found the important report on *Functional Requirements for Bibliographical Description* FRBR 1998. The Anglo-American rules, AACR, are explained in Maxwell's Handbook 1997. See also the *Cataloguer's bookshelf* web site and the also the following for MARC: a good official site is [www.loc.gov/](http://www.loc.gov/) . . . including a lot of discussion papers; very good material also at the OCLC site, in particular . . . RAK The German rules are explained at

**Applications** *Allegro* is a small library system developed at the Technische Universität in Braunschweig by Bernhard Eversberg, its documentation comprises also an outstanding comparison and discussion of library data formats.

**Records Management and Archives** Management of living (current) administrative records has evolved, of course, independent of bibliographic considerations. For a recent standard see (1). The German practice before the advent of Electronic documents is detailed in Hoffmann (6)

---

## Chapter 2. Requirements Overview

This chapter gives a complete overview specification of the requirements for Pybliographer.

For a general view of *existing* bibliographical and related applications see *supra* 1.1. The existing interfaces are detailed below at 2.3 insofar as they remain relevant. General usage considerations can be found also in Part II. – A summary statement of purpose is given at the beginning of chapter 1.

General information can also be found in the *User's Guide*.

---

### 2.1 Goals and Purposes

Many people, and academics in particular collect a set of references or citations. Historically, this collection may have been contained on a set of index cards. However, for some time now, programs have been used to help with the tedious aspects of the work, in particular if the number of references has grown into hundred or thousands.

Bibliographic database managers or *reference managers* are a specialized type of a database manager designed for the handling of bibliographic references. They are also known as personal information systems, bibliographic reference managers, or personal bibliographic software. These systems help with essential research or publishing tasks:<sup>1</sup>

- Building a database of references to journal articles, books, and other research publications, using both manual and electronic input methods
- Searching the created database by author, subject, journal name, and other criteria
- Using the database to link references in word processed documents.
- Generate the bibliography in the correct style for publication.
- Output to separate file for interchange or printing.

Such a database, once established, might be put to many uses, some examples (with modifications taken from (sat:hsu01)) follow:

- Create course reserve lists and reading lists for students
- Maintain faculty publication lists
- Catalog special collections
- Keep track of reprint collections
- Maintain bibliographies of references in research areas of personal interest
- Prepare instantaneous formatted in-text citations and bibliographies during manuscript preparation

---

<sup>1</sup>this discussion is taken from (sat:hsu01) and (16)



## 2. Requirements Overview

- Create and maintain a reference database shared among a group of resources across a network
- Publish a web-based bibliography

Reference manager software has been around for a number of years, but lately has assumed a larger role, and faces new challenges owing to several recent advances, as follows:

- Tighter integration with word processors. – These features enable you to insert references from your bibliographic database while you type.
- Conversely, the flow of information from the Editor/Word processor comes into focus, with ideas such as imbedding full bibliographic records into documents and automatic tracking of citations.
- Z39.50 Search and Retrieval Protocol – Many library catalogues are accessible over the internet via this protocol. This means that the user can search the catalogues of any accessible library, and download references directly into his personal bibliographic database.
- This exposes the user to a mixture of descriptions that are often more detailed and use more sophisticated methods of description than what he is used to.
- The Web – PBM software can link directly from a URL in a citation to the supporting site and also open a locally stored document.
- In addition the web browser has become the universal virtual terminal, allowing access from every place in the world. Therefore many wish to publish their bibliographies on the web, and perhaps even more.
- This development has led to a re-appraisal of many received ideas and rules, leading to a more fully understanding of the issues
- The gap that, e.g., has separated library and archival theory and practice, is slowly narrowing. Thus it becomes conceivable to intergrate more types of resources, i.e., the unique resources the office and archive deals with, or the products of project work, that hitherto received only haphazard attention.

---

### 2.2 Program Audience and Context

The intended main user of this program is the academic scholar/writer, who needs a tool to maintain his growing set of references and to access his own and other databases and document repositories. The *workflow 2* captures the essence of these transactions with the system, as the purpose is usually to author own documents.

In work group settings the inquiry-only type of operation (*workflow 1*) will supposedly become more important; consider students using a *bibliography server*. – Note that, in any case, searching and browsing are the fundamental interactions with the system and have the greatest usability impact.

## 2. Requirements Overview

The degree of administrative/programming and cataloguing/organising work, as detailed in *workflow 3* will vary greatly. Major factors are:

1. Individual vs. collaborative use.
2. The degree of detail needed, e.g., maintaining a database for long term use requires a greater level of detail than short time, project type work.
3. Standard bibliographical description vs. special needs, such as recording of archival sources, which may involve unique data acquisition requirements, such as complex case histories.
4. Standard reference lists *à la* BibTeX vs. complicated output processing, e.g., a web based classified, annotated bibliography.

---

**Note:**

- The above points should be explained in User's Guide chapters 1 and 3.
- 

The context is further defined by the interfaces to:

1. Document repositories, local and remote.
2. Other databases and sources of information – mainly Query and Import interface.
3. Document preparation tools – to be defined, except for BibTeX compatibility.
4. Output processors – XSLT should unify this.

Typical workflows are explained below. In the last column, references are given to the *Use cases* in section 2.5. In general the workflows can be subdivided into the following stages and tasks:

### 2.2.1 Scenario: Write a Paper

Within the following scenario, you will find yourself searching and browsing (as above in ??) many times; now these *Use Cases* are imbedded in more complex processes. So, e.g., stages 6 and 7 below consist not only of searching and browsing activities, but integrate them with the writing process: an item is not only found, but its citation is included in the document in preparation, it is placed on or taken from a work list, a reference is linked back to the cited item etc.

### 2.2.2 Scenario: Maintain a Bibliography

The linking process that lies at the core of Pybliographer's support of the writing process is applied in the next, most demanding situation to documents of various kinds that are *given* and *augmented* for better analysis and access.

## 2. Requirements Overview

Stage	Description	Related
1	Choose a database, <i>if necessary</i>	011
2	Do a <i>Known-Item-Search</i> , e.g., with an author/title pair, or even	001
3	Formulate an <i>expert query</i> , this requires some acquaintance with the data base, or	003
4	<i>Browse an Index</i> , if you are looking for things you don't know beforehand (or might have forgotten), e.g., in a Journals Index, or	010
5	Follow a cross-reference, e.g., by looking at the works of an author, or papers which cite another.	
6	<i>Look over the results</i> . Examine individual records.	
7	Based on the results, variate or make more precise the search question.	002
8	Mark items to set them aside for further examination or processing.	007, 008
9	Add a note to an item.	005, 008
10	Output all or any of the items, in various formats, to another file, to the clipboard for inclusion in another document (using <i>Drag-and-drop</i> ), to the printer.	006

Stage	Description
	<b>Collect Information</b>
1	Maintain a reading list.
2	Acquire information from databases.
3	Acquire information from other sources.
4	Add notes and quotes.
	<b>Write</b>
5	Maintain a work plan.
6	Find items, notes, quotes, etc.
7	Insert and mark-up references and quotes.
8	Audit work done. (Check cited works, etc.)
9	Process references for output formatting.

Stage	Description
1	Organise work.
2	Add and import records.
3	Check, edit and merge records.
4	Add documents.
5	Describe (mostly) archival items and collections.
6	Maintain authority data.
7	Add and edit access points.
8	Add annotations, other texts.
9	Process output.
10	Export data for backup, integration.

---

## 2.3 Interfaces

### 2.3.1 Actors and Activities

Actors	Description
User	assumes different roles: <ul style="list-style-type: none"><li>• <i>searches</i> internal and external databases</li><li>• <i>adds</i> references and notes</li><li>• <i>formats</i> reference lists and bibliographies</li><li>• <i>accesses</i> external documents</li><li>• <i>administrates</i> Pybliographer itself.</li></ul>

Table 2.1.: Actors

### 2.3.2 External Interfaces

### 2.3.3 Internal Interfaces

---

## 2.4 Program Features

The program offers the standard features of similar programs, as shown in Appendix A.

### Optional functions

- Connection with word processors and Emacsen.
- Web interface
- Work group support
- Archival support
- Spell checking and other input enhancements

## 2. Requirements Overview

Ref.	Description	Related	See also
R010	Entering, merging records	9004	
R020	Searching records	9001	
R030	External search, importing data	9002	
R040	Formatting bibliographies and citations.	9005	
R050	Efficient database		
R060	Adequate, extensible database schema	9006	
R070	Adequate classification		
R080	Annotations of various types		
R090	Multiple languages and scripts		

Table 2.2.: Mandatory Features

### Excluded functions

- Library loan and acquisition functions.
- Repository and collection management.
- Sophisticated output – only BibTeX level

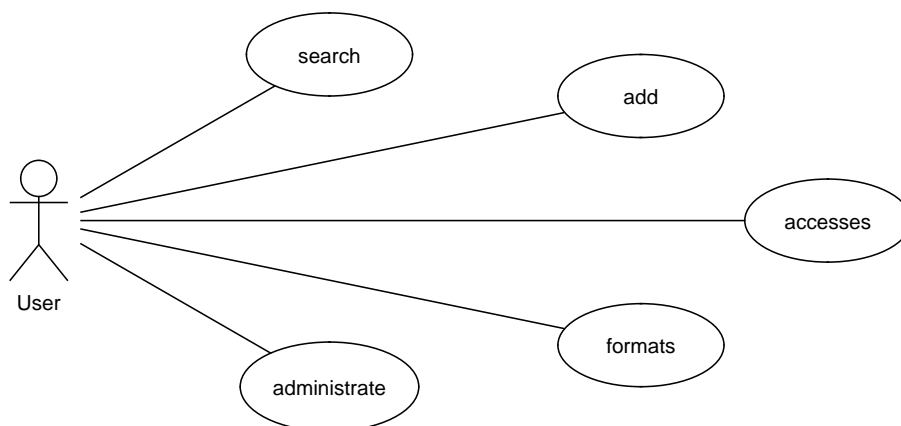


Figure 2.1.: Use case diagram (overview)

*Note:* Perhaps it is better to use the describe – organise – refer structure from ?? supra?

## 2.5 Usecases

001	<b>Search Database</b> A simple search (known item search) is performed. The result set is displayed.	
-----	---	--

## 2. Requirements Overview

002	<b>Modified Search</b> A search that was not fully satisfying is done again with slight variations or on the result set to further reduce it.	
003	<b>Expert Search</b> A so called expert search is prepared and executed (perhaps it is stored or reused).	
005	<b>Add an note to an item</b> A note is added to an item.	
003	<b>Output Item</b> An item is selected from a result set, formatted (according to preset choices) and presented via drag and drop or other means to the user for inclusion in other documents etc.	
004	<b>Mark and list Items</b> Make a list of items for further processing (e.g., mark the items and retrieve them together)	
008	<b>Add an action to an item</b> An item (or a set of them) is selected for further processing, an indicative action is added to them.	
009	<b>Navigation in result sets</b> Most activities produce result sets. It is important to be able to look back at them, combine them, etc. (This is a difficult job – I know of no good example to follow!)	
007	<b>Browse an index</b> An index, or equally valid, any result set, is browsed.	
006	<b>Select a database</b> All processing is against a database (?). A choice is given, even if a default database is defined.	
012	<b>Cite a reference</b> Place a marker in a document to indicate a cited reference. This function needs the collaboration of the document processor/editor.	
013	<b>Manage references</b> The references that form part of a certain reference list (i.e. that are cited by a certain document) are collected. Sometimes database services already produce such lists, there are also projects to extract these data from documents (OpCit).	

## 2. Requirements Overview

014	<b>Select display and output formats.</b> For various display purposes as well as for output to diverse targets the user can select and modify the applicable formatting.	
011	<b>Make a bibliography (reference list)</b> This is the equivalent of running bibtex.	
016	<b>Operate on result sets</b> Result sets are a fundamental UI element, in addition one can use them for set algebra.	
017	<b>Organise folders and Databases</b> The Pybliographer database is logically organised as a folder hierarchy; external databases are listed separately. Those lists and the associated entries may need user attention.	
020	<b>Create a Record</b> A record is created. This can be a Topic record or a Bibliographical record.	
021	<b>Modify or delete a record</b>	
023	<b>Preset input</b> This usecase groups various ways to simplify data entry.	
024	<b>Search and replace records</b> Perform a (regexp) search-and-replace action on a result set. It is important that the replacing can be suitably limited.	
025	<b>Import data</b> Obtain and store data from external sources.	
015	<b>User administration</b> Modify preferences and possibly access rights on a per user basis. [pro memoria: must not assume that there will be only one user]	
018	<b>Control external connections</b> Control parameters for cooperation with word processors or the like.	

---

## 2.6 Requirements Directory

The following table directs you to the detailed lists of functional requirements that are distributed over this document.

1. Generalities
  - a) Packageing and Installation
  - b) Documentation
  - c) Authorisation and Protection
2. Bibliographical description
  - a) Generalities
3. Subject description and access
  - a) Generalities
  - b) Persons
  - c) Subject Headings
  - d) Classifications
  - e) Coded lists and Classifiers
  - f) Keywords, etc.
  - g) Full text search
4. User Interface, Input, and Import
  - a) GUI generalities
  - b) Manual Data Entry
  - c) Editing and Checking
  - d) Import facilities
  - e) Spellchecking and Other Assitive Techniques
  - f) Duplication Checking
  - g) Accessibility and misc.
5. Searching, Formatting, Display, and Output
  - a) Generalities
6. Annotations, Actions, Extensions
  - a) Generalities
7. API, Interfaces, Services
  - a) Generalities
8. Miscellaneous, dubious, ad interim
  - a) Duplication checking
  - b) Character sets
  - c) Indexing
  - d) Query service



---

**2.7      Objects and Classes – the problem domain**

**2.7.1    Interfaces**

---

**2.8      Other Requirements**

---

**2.9      Constraints**

---

## Chapter 3. Architecture Overview

This chapter describes the architecture and components.

---

### 3.1 Application Core

Pybliographer provides access to **Databases** of bibliographical informations and **Documents (Resources)**. It allows **Users** to store and retrieve records in an **internal Database**, including the ability to import the results of **Queries** against external databases.

It organises the stored records in **Lists** and **Folders**, maintains **Indexes** on various attributes, as well as **Authority files**, such as for **Persons**, **Subject headings**, **Classifications**, and other **Descriptors**.

Items in the database can be equipped with **Annotations**; these include **Actions** that are requested (and assorted status), **Abstracts** or **Notes** reflecting the contents, and also **Quotations** taken from the source document, structured **Case annotations** for further processing, and also **Proxy references** to external documents which might serve the same purposes.

**Holdings** are recorded, which include references to electronic copies as well as shelf numbers or other location information, both at the user's site or any other institution, anywhere.

The bibliographical **Description** allows for **Extensions** to deal with e.g., special materials. For multi-level description a **Hierarchy** of items is created.

Items may be assigned a **Security Label**, to restrict their dissemination.

Items are formatted according to **Formats** that are user selectable and -definable.

---

### 3.2 Architecure Baseline

---

### 3.3 Implementation Plan

---

## **Part II. Design Considerations**

---

## Chapter 4. The Graphical User Interface

In this chapter we discuss the user interface, in our case based on the **Gtk2.0** toolkit. First we give an overview of the user interface elements, then we look at the requirements of individual widgets.

**See also:** For the integration of Pybliographer into Word Processors and Browsers, see section 6.2; for using Pybliographer from the commandline or scripts, see chapter 7.8.

---

### 4.1 Menus

The menus are the main way to access Pybliographer's functions, and must accordingly be laid out with care. All functions should be accessible through the menu interface.

Usually the main menu is laid out as follows:

File • Edit • View • ... Help

This is an organisation that has been criticised often enough, as being appropriate only for one special case of program, viz. an editor like application. It is, however, well established.

Nevertheless, there is broad support for an adapted menu structure, if the application doesn't fit the mold of an editor.

#### 4.1.1 The Menu Bar

So I propose the following:

**Pybliographer** The fulcrum of the application. Contains the **Preferences...** button, and then buttons for **Database**, **Folder**, ..., each of which immediately switches to the respective main view. In addition, exit function.

**VarObject** Where Varobject is Database, ... depending upon the preceding choice.

**Edit** Context dependent standard editing,

**View** Select view options, among them for window layout, e.g., toolbat toggle, and for record display. *Perhaps:* elementary formatting options for entries

**Go** Necessary to navigate the result sets

**Search** To select the search dialog, to select a query for editing or execution, and to select an index for browsing.

### **Format**

**Scripts** to run external scripts (perhaps even to write them?) *presuming that the user uses this way of customisation at all*

**Window** *if there should be multiple windows*

**Help** Standard help menu

### **4.1.2 The Application Menu**

**Note:**

- Compare also the Apple Aqua Guidelines.

### **4.1.3 The Database Menu**

### **4.1.4 The Folder Menu**

### **4.1.5 The Record Menu**

---

## **4.2 Displaying and Using Indices**

The usual way of perusing a bibliography or a catalogue is by means of a listing, either according to the author's name, or according to the title, a keyword, or any other suitable piece of information.

With online catalogues, one usually has access to the underlying (technical) *indices*, if only to compensate for the limitations of usual search facilities and the missing overview possible with today's display devices.

It is best to have multiple indices (Indexviews), so as to make best use of the structure of the data (e.g., the *Person or Author Index* would usually show the works of the author upon selection (as a tree view) not the authority record of the author, a title index might in a similar way allow subordinate entries for editions or translations, etc.

Sometimes special indices are required. A manuscript collection might want to enter and find listed the first few words of a manuscript or letter, a specialist might need the names of translators, or scribes, or printers. So there must be a degree of configurability in addition to that needed to cater for the differing tastes and traditions.

### 4.2.1 Marking items

A mark is available in every list view to the left of an entry, it is used as the indicator for one special folder “Marked”. By toggling the mark, one can easily change the exten

### 4.2.2 Accelerator Key Assignments

Accelerator keys allow easy selection of functions, in particular easy switching of views, which is in particular desirable when doing data entry and editing. These are activities which repeatedly and predictably need a great number of screens to proceed, switching them with the mouse is particularly unpleasant. Consistent assignment of *shortcuts* is needed.

---

## 4.3 Appendix: Accessibility Guidelines

From [www.pro.gov.uk/recordsmanagement/2002referencefinal.pdf](http://www.pro.gov.uk/recordsmanagement/2002referencefinal.pdf)

These samples guidelines are indicative only, and are drawn from information available at [www.state.me.us/CIO/accessibility/software\\_policy.html](http://www.state.me.us/CIO/accessibility/software_policy.html)

- A program must provide keyboard access to all functions of the application. All actions required or available by the program must be available with keystrokes, i.e., keyboard equivalents for all mouse actions including but not limited to, buttons, scroll windows, text entry fields and pop-up menus.
- A program must have a keyboard control sequence among all program controls and focal points. (e.g. using the tab key to navigate among edit fields, text boxes, buttons, and all other controls).
- The focus must follow the keystroke, that is, using the arrow keys to navigate through a list followed by pressing the ENTER key or spacebar to select the desired item.
- The software shall not interfere with existing accessibility features built into the operating system, such as Sticky keys, Slow Keys and Repeat Keys.
- Timed responses are not to be used unless the timing parameter can be adjusted by an individual user.

#### *4. The Graphical User Interface*

- There shall be selectable visual and auditory indication of key status for all toggle keys. (i.e. visual and auditory status indicators for keys such as the Number Lock, Shift/Caps Lock, and Scroll Lock keys.
- All icons shall have clear precise text labels included on the focus or provide a user-selected option of text-only buttons.
- The use of icons shall be consistent throughout the application. Pull-down menu equivalents must be provided for Icon functions (menu, tool and format bar).
- There must be keyboard access to all pull-down menus.
- For graphic text, system text drawing tools or other industry standard methods must be used so that screen reader software can interpret the image.
- A visual cue for all audio alerts must be provided. The Sounds feature must be supported where built into the operating system. The user must be allowed to disable or adjust sound volume.
- Colour-coding is not to be used as the only means of conveying information or indicating an action. An alternative or parallel method that can be used by individuals who do not possess the ability to identify colours must always be provided.
- The application must support user defined color settings system wide. Highlighting should also be Viewable with inverted colors.
- No patterned backgrounds behind text or important graphics are to be used.
- User adjustment of, or user disabling of flashing, rotating or moving displays must be permitted to the extent that it does not interfere with the purpose of the application. Consistently position the descriptions or labels for data fields immediately next to the field.
- All reports and program output must be available in a format that is accessible by screen readers and other access systems.

---

## Chapter 5. Organising and Manipulating References

One aspect of Pybliographer is its use to *organise references* – that lays the accent onto the variety of uses that these references could be put to, and the concomitant variety of possible organisations we have to cope with. This de-emphasises the bibliographical description, as a rule, but stresses our ability to annotate and link.

As a rule, our data is kept in *one* database, organised into *folders* (compare Biblioscape A.3). A variety of *Indices* are kept to allow fast access.

---

### Note:

- Most entities used in this connection are fully blown DB objects, that can hold metadata, annotations, etc. Possible exceptions are *keywords*,
  - Another question not yet discussed is *indexing* – but there are strong connections.
  - Checking against a simple index could be used alongside spelling correction during input, that is what many programs do.
- 

---

### 5.1 Folders and Lists

We often have to do with the need to structure the search space, to group entries, and generally to organise the data in various ways.

One way to do so, which is perhaps particularly useful and of near universal application, is to use folders. These are modelled after the Biblioscape feature of this name, cf. also Fowler's *Portfolio pattern*. Lists are in this connection simply stand-alone (unrooted) folders that may contain any number of items but do not allow sub-folders.

With each folder are associated various attributes and data structures besides the items that it contains (or the subfolders, resp.).

- Display format: in particular for special purpose lists.
- Sorting order: same considerations. These could be combined, but are of unlike type.
- Filter condition.
- Base set.

Major operations:

- Add (move, copy) an item into one list or folder.  
Supposedly more often items are marked into a list, or subsumed into a folder.
- Display a folder (means display the union of all subfolders if not a leaf folder).
- Open a folder (means display if a leaf folder).
- Close a folder.



## 5. Organising and Manipulating References

In Biblioscape there is a so-called *dynamic folder* capability, meaning a folder the extend of which is determined by executing a SQL [like] query against the data base. I.e., a standard folder is defined by an arbitrary assignment, while a dynamic folder is defined by the value of one or more attributes. It is evident that this is an interesting variant.

To a degree, the assignment to a folder could imply the assignment of an attribute, but it is not clear if this direction there is much to be gained.

I would argue against tying the folders to any extrinsic purpose or organisation, but allow them to be freely used by the user. This does not exclude (so it seems) using *existing* attributes for classification, while acknowledging their limitations in most cases.

### 5.1.1 Lists

Any result set is a *List*, and vice versa. So lists are a very prominent feature in Pybliographer – by using them we allow intermediate results to be saved, as task related information to be easily kept.

List cannot be nested, to distinguish them from folders and to relieve the application from the problem of inferring an hierarchy, but they could be turned into folders, if the need arises.

---

## 5.2 Keywords

It is easy to add a *keyword* to an entry and easy to search for it. So this is an favourite feature. But this is also a technique that has its limitations; let us consider some situations to find out more:

We may simply think of coarsely classifying an item, putting it onto one heap or another, where it is understood that it can be very diverse schemes involved, i.e., the one want to classify according to his work plan, the other according to tasks involved, a third assigns a priority. This is, however, a task that is much more better accomplished with folders, indeed, this is their foremost purpose.

Another use is to indicate the contents, as a sort of diminutive *subject headings* (see *infra* section 5.3). Compared to these they have the following deficits:

- There is not *thesaurus structure*
- There is no support for multi-lingual application
- In consequence, little stability can be expected in its application

Therefore, it is of little interest in that respect.

## *5. Organising and Manipulating References*

However, another usage that in fact might not come easily into one's mind is valuable: adding search terms to compensate for variations in spelling, terminology or for omissions in titling that would otherwise require more elaborate mechanisms.

## 5.3 Subject Headings

*Subject headings* are, by contrast, keywords with special enhancements which make them more useful for the purpose of indicating the content of an item.

Subject headings are like keywords given to an item in order to indicate its content, but they have a *syntactic structure* that sets them off from keywords, and they are taken from a *thesaurus*, a controlled vocabulary with special features for improved usability, both at the time the item is described and at the time the search is formulated.

**Note:** Some interesting contributions are in the little volume (14).

An item is given one or more subject headings (up to six in Germany), each of which is a chain of one or more (up to six in Germany) subject elements (my translation).

### 3.3 Subject Headings

- 3.3.01** It shall be possible to add Subject Headings to Entries. *Minimally for all manifestations, but also valuable for e.g., quotes and notes*
- 3.3.02 ?** It shall be possible to distinguish primary and secondary references *In the sense of giving a weight.*
- 3.3.03** It shall be possible to limit the number of subject heading per entry *Some rules require this – including zero depending upon type, e.g.*
- 3.3.04** It shall be possible to derive subject headings during 9271 import
- 3.3.05** The assignment of subject headings (i.e. manual entry) shall be assisted *by look up, browsing, completion, copying etc.* 9276, 9280
- 3.3.10** Chained subject headings shall be used *Syntactic structure v. infra*
- 3.3.11** Types of heading elements shall be distinguished *persons, geographical names, historical periods, forms*
- 3.3.12 ?** It shall be possible to limit permutations of the chain *(This is not without problems, though.)*
- 3.3.13** Predefined chains shall be available *v. infra*
- 3.3.14** Subject headings shall be entities, not strings *i.e. shared not copied – generally so* 9273, 9275
- 3.3.15** Subject headings shall be editable, im-/export- and annotated 9272, 9277, 9278, 9279
- 3.3.20** Syntactical searching must be used *v. infra*
- 3.3.21** PF references must be followed automatically
- 3.3.22** A folding display must be used
- 3.3.30** Multiple schemas must be supported *To allow mapping during or after import, e.g.*
- 3.3.31** A Thesaurus structure must be used 9113 9114 9128
- 3.3.32** Multiple languages must be supported
- 3.3.33** Import mappings must be possible

- 3.3.34** Adding a classification to SHs must be possible (*This is done in Zürich, e.g.*)
- 3.3.35** It should be possible to produce (hierarchically) sorted list, e.g., for subject bibliographies 9208
- 

### 5.3.1 Syntactic Structure

Why do subject headings need a syntactic structure? Isn't this only good for card catalogues? But consider the following.

There are indeed two kinds of syntactical structure present. First we compose subject headings as chains of subject elements in order to make more precise the intended meaning (subordinating index). The improvement is, of course, more easily felt with greater databases. But even for smaller databases the following considerations are important:

If some subject elements occur very frequently, in variable combinations, a worthwhile gain in precision results compared to mere juxtaposition. This is for example the case for papers situated on the border between two disciplines, or on the horizon of one's interests, where the choice of headings becomes less specific.

It is also easier to produce and to peruse both, printed and displayed lists and registers alike, if an additional structure is available.

And, whenever we import data we might well get data structured in this way, why should we destroy information needlessly?

A second kind of syntactic structure that must be taken into account, is given within each element: the qualifying information that is used to distinguish between homonymes. It is particular important to avoid the qualifiers when searching, as e.g. in group <mathematics>, lest a search for *mathematics* should produce all entries indexed with the former.

### 5.3.2 Thesaurus Structure

---

## 5.4 Classifications

Although a *thesaurus* and a classification share a common storage structure, they stem from different principles, are build and used in quite different ways, and as a result complement each other more than that they would compete.

## 5.5 Indexes

Indexes are, of course, a rather general concept; it is treated here, and not in the database chapter, because they are essentially in their unrefined way, used in many programs to emulate what Pybliographer accomplishes with more sophistication.

Thus we can often find the following picture (see, e.g., the evaluations by F. Dell'Orso (ors:bfs02))

- a field is indexed, sometimes every field gets its own index, perhaps separating authors from editors,
- we deal usually with strings, thus there is no option of annotating, or of silently following references
- sometimes we have copy semantics,<sup>1</sup>
- the index is often used as data entry helper, checking for typos, or offering completion

### 3.0 General considerations – Indexes

<b>3.0.01</b>	The creation of indexes shall be freely possible	9270
<b>3.0.02</b>	The creation of Indexes shall be usually possible with little or no programming, by subclassing or cloning	
<b>3.0.03</b>	Indexes shall be maintained <i>real-time</i> with option of background (re-)building	9271
<b>3.0.04</b>	Multiple fields (data elements) shall be able to share one index	
<b>3.0.05</b>	All indexes shall be searchable	9281
<b>3.0.06</b>	All indexes should be usable during input for validation or completion	9280
<b>3.0.07</b>	Application should provide sufficient utilities for printing etc.	9278
<b>3.0.08 ?</b>	Indexes should be shareable	9283
<b>3.0.09</b>	The number of matching records should be displayed in appropriate circumstances	9282

<sup>1</sup>Which is, b.t.w., the rule in American library applications, such as OCLC.

---

## Chapter 6. Searching, Selecting, and Formatting Data

A *search* is an operation that is performed against one or more *databases* – usually locally represented by Pybliographer and remotely by database servers known as *connections* – and that results uniformly in a *result set*.

As a rule, such a result set is if necessary *imported* as such into the Pybliographer database, and sooner or later *selected* from.

A *selection* is any subset of the database, usually the result of manually inspecting (*scanning*) some result set, that is presented to a processing option: either to be set apart for further processing at a later time, or to be processed by the output routines in order to be presented or exported. [XXX Are there other uses? ]

---

### Note:

- Results sets can vary considerably in size. Efficient means of handling them are important.
  - Various situations are to be considered and equally supported
    - One looks up price/holdings/availability information for an already known title. That should result in the new information being added to the existing entry (and perhaps the latter being a bit improved). Possible actions include posting a note to pick up the book at the next visit to the library, or to make a copy from a journal, or to order it from the bookseller or the ILL service.
    - One browses several data bases and cumulates the results. Later one reduces the set and adds it to the data base.
    - One adds, perhaps on a regular schedule, records from an external source. Evidently this situation calls for other mechanisms and tools than the previous one, in spite of its superficial similarity.
    - One builds a bibliography or similar documentation database – involves most of the above situations and adds requirements for stringent administrative control.
- 

---

### 6.1 A generic query service

The exact possibilities that a database provides for searching are as manifold as are the data. Paradoxically this makes it more easy to provide one integrated query service – any attempt to equip every database and every service with its own query interface cannot but falter in view of the enormous programming investment that would be needed.

It's not clear, perhaps, how the individual services can communicate their capabilities upwards towards the query service, but it should be evident that *some* communication would be helpful – even if one fully acknowledges that the assessment of a database service must usually include information that is not available to the program. (Example: cataloguing rules – RAK, AACR, . . .)

---

## 6.2 Integration with editors, wordprocessors, and browsers

Integration with editors like Emacs, word processors like Abiword, and browsers like Mozilla enhances ease of use, broadens its field of application and reduces user's keystrokes and thus the opportunity for errors on his side.

We distinguish the following use cases:

**Adding a citation** When writing, the user wants to add a reference to certain document, of which he might remember no more than some words from the title, or the author, or even less, but as a rule, not the exact citation key that BibT<sub>E</sub>X e.g., would use to identify it.

The user should refer to the document by selecting from a result set or shortlist.

**Inserting citations** Quite often the case arises that a document must be cited in, say, an email without wanting the whole (BibT<sub>E</sub>X) machinery to be put in action.

A simply formatted citation could be offered via *Drag and Drop*.

**Annotating** When reading, the user wants to add a note.

He is helped with creating and maintaining annotations to documents (registering them with Pybliographer); annotations can be large and are not suitable for storage *in* the bibliographical record, but a link could. On the other hand it is conceivable to point back from the annotation to the document, so it could stand on its own.

**Viewing documents** An electronic document could be requested.

Define appropriate viewers and requestors (mainly for eprints – the rest is more simple).

**Extracting metadata** An electronic document is perused and the wish is that it be catalogued.

A limited bibliographical description can be build from the metadata that is stored and communicated with the document. *Importantly*: If downloaded, the local file address should be stored as well.

## *6. Searching, Selecting, and Formatting Data*

### **6.2.1 Emacsen**

### **6.2.2 Other editors**

### **6.2.3 Mozilla**

### **6.2.4 Other browsers**

### **6.2.5 Kword**

### **6.2.6 Abiword**

### **6.2.7 Open Office**

### **6.2.8 Other word processors**

### **6.2.9 Other applications**

---

## **6.3 Formatting references and bibliographies**

---

## **6.4 Other applications**

[Connecting to OPACS, etc. ]



---

## Chapter 7. Interfacing to External Ressources

---

### 7.1 Connecting to external databases

Pybliographer uses *connections* to external ressources for the following purposes:

- To look up a catalogue (for immediate consumption, so to say), i.e., as a client program.
- To obtain, as a result of a query, additional data for import.
- To obtain additional documents that are referred to from the data base, i.e., via an *URL* or an *Eprints* link.
- To perform operations on the remote system, i.e., to request books or copies by means of an *OPAC*.
- ∅ To cooperate with other Pybliographers, sharing and updating data.

---

### 7.2 Writing Import Filters

Import filters are used not only when a *file* is imported, in a narrow sense, but also when opening a file, and, perhaps more importantly, when processing the results of a *query*.

An *Import Module* provides the needed support. Typically, there is one such module for every major data format, say MARC or BibTEX.

A major part of such a module is a Reader class, which subclasses (is derived from, and specialises) the class `Importer` in the module `Pyblib.Import`.

In fact, most Reader modules do not derive directly from `Importer`, but from an intermediary, as `TaggedReader` for data structured as a sequence of tagged fields, like in the MARC format, `TextReader` for data that comes as ‘unstructured’ text, like taken from the references of an article, `XmlReader` for XML formatted text.

In this way, we have a *framework* for import modules, one that makes writing or modifying an import reader much more simple. This is important, because requirements and interface continue to evolve.

The format specific modules are not the end of the story, yet. For one, a format like MARC is in fact so vast that it will probably never be completely implemented and used, neither by a library, nor by Pybliographer. It is, however, to be expected that one time or the other, someone would be better off if he could make use of that exotic feature, and would be happy to add support. In addition, one may encounter a situation in which the standard processing is inadequate and better be modified. Thus

it is useful to be able to have an easy way of adapting to one's very special needs, be it in terms of adding or of overriding standard behaviour, i.e., methods.

There is yet another point to be made. A framework is also a way of sharing the implementation effort. That means that adding code to the common base is more attractive and this in turn makes the use of Pybliographer more attractive. So everyone profits.

### 7.2.1 **Framework Services**

The base and intermediate classes, the framework, provide the following services and features:

**Input** The source data is accepted by various means: files, iterators, in-core objects, and made available in a standardised way to the processing.

**User interfacing** The user interface, always requiring a big programming effort, is structured by the base classes (together, of course, with the Gui Component), thus sharing a lot of work.

**Data handling** In the past, the handling of the input data was almost completely left to the format specific code, resulting in an enormous duplication of code and often poor exploitation of the source data. With the new design, most of the semantic data handling will be done by functions in specialised modules, sharply distinguishing between the parsing (syntactic) and database aspects of the task.

**Duplication checking** Like the user interface, the duplication check is an example of a programming task, that is not easily undertaken within the confines of one particular application, but only if it can be done wholesale.

**On- and offline correction** It is highly desirable to be able to correct imported data in a systematic way, the framework will provide for that, both during initial processing, dubbed on-line, as well as at a later time.

**Customisation** There are ways to adapt the import processing which can be done easily in a generic way. It is, e.g., simple to exclude all data from certain categories (tags) from further processing and therefore TaggedReader provides a parameter to specify such tags – it can even be done interactively from an options dialogue. No programming needed.

**Data management aspects** The data imported will be kept logically identified as long as it is desired, without the need to keep separate files for it. That remedies a major problem with the work organisation in the past and it is necessary in any way for a data base oriented Pybliographer.

### 7.2.1.1 Class hierarchy

The fundamental class is `ImporterReader` in Module `Import.py`. It implements an Iterator interface, viz. `first()` and `next()` methods.

```
class ImportReader (Iterator.Iterator):

    """Base class for all import reader classes.

    Support for input methods, encodings, database and GUI connections,
    """

    ### The following need no change as a rule:
    def first (self):
        return self.next()

    def next (self, entry=None, data=None):
        """Process the next entry from the input source.
        entry and data argument are provided for easy testing."""

        e = entry or self.next_entry()
        x = data or self.read_next()
        if self.options.has_key('preserve_input'):
            e.lines = x
        self.entry = e
        return self.parse(x)
```

Subclasses will need to provide, among others, implementations of `read_next` and `parse`.

A typical implementation of `parse` from `TaggedReader` follows:

```
def parse (x):

    self.begin_record(x)

    ## it may be convenient, if the read_next routine already assembles
    ## continuation lines. Let's assume this

    for i in x:
        tag = i[0:self.tagcol]
        data = i[tagcol:]

        if discardrx and discardrx.match(tag):
```

## 7. Interfacing to External Ressources

```
        continue

    if _cache.has_key(tag):
        _cache[tag] (self, tag, data)
    else:
        methname = 'do_'+str(tag)

        if hasattr(self, methname):
            method = getattr(self, methname)
        else :
            method = self.do_tag

        method(self, tag, data)
        _cache[tag] = method

    self.end_record(x)
    return self.entry
```

So in this case, for a tag 'XYZ', a subclass implemented method 'do\_XYZ (self, tag, data)' would be called, if it exists. If not, 'do\_tag' would be called and as a rule, distinguish according to the tag, as directed by parameters of the class, whether to ignore, or to store (in a generic way) the data.

### 7.2.2 Class parameters

#### Control objects

**control=None** specifies the associated control object.

#### Input

Three possibilities are provided for. The data parameter is particularly useful for testing. *One and only one must be selected from the following*

**file=filename** if input is from a file,

**data=data object** if input is from an incore object,

**iter=iterator** if an iterator is specified.

---

### **7.3 Writing Export Filters**

---

### **7.4 Pybliographer as Web Server**

---

### **7.5 Requesting remote operations**

---

### **7.6 Requesting documents**

Given an entry that refers an electronic document, it is an easy idea to consider requesting it by mouse-click, so to say.

## **7.7 Bibliographic Data: types, formats, schemata**

### **7.7.1 Bibliographical Objects**

See the IFLA report.

### **7.7.2 Persons**

### **7.7.3 Works**

### **7.7.4 Subjects**

### **7.7.5 Database Organisation**

## **7.8 Scripting and Configuration**

## **7.9 Technical Questions**

### **7.9.1 Architectural issues**

### **7.9.2 Compatability issues**

**0016** Remain compatible with BibTeX

### **7.9.3 Charset and markup issues**

**0001** Establish Unicode internally and for storage



---

## **Part III. Component Design**

---

## Chapter 8. The Application Core

---

## Chapter 9. Common Control

**Purpose** Separates the utility functions that pertain to many classes in the application with respect to: (i) UI functionality (ii) preferences setup and storage, and (iii) persistence

**Modules** Coco

**External Dependencies** XML for reading/writing minimum configuration data – GUI toolkit for GUI functions

**Internal Dependencies** Package Storage provides persistence – used by whole application

**Initialisation** Upon start-up load minimum data, uses primary database for the rest.

---

### 9.1 Description

---

## Chapter 10. The Storage Component

**Purpose** Persistence support for the rest of Pybliographer. This package forms part of the application core. It is always required.

**Modules** Storage ...

**External Dependencies** Database server or package (BsdDB3, Postgresql, Mysql, SQLite). – XSLT formatter. – Internet/Z39.50 access.

**Internal Dependencies** Common Control for preferences, UI connection, – Bibliographic and Topic for loading/restoring objects and extracting indexing information. – Writer for output.

**Initialisation** Primary database needs special consideration as it contains most configuration data (i.e., need for a bootstrap file).

**Conventions** Prefix: DB\_

---

### 10.1 Description

To the rest of the application, the package Storage provides

- the ability to store items (records) under a numeric ID,
- the ability to maintain (so called secondary) indexes,
- an object cache,
- access by ID, or via an index,
- iteration over subsets of items,
- connection to external databases,
- execution of queries locally and externally,
- import of items from the results of queries or files,
- storage for configuraion data.

---

### 10.2 Module Storage Overview

This module contains the interface to the database implementation and the common interface classes DB\_Set, DB\_Iter...

## 10. The Storage Component

Class	Major Attributes	Description
Recordset Database		any set of records (a list in the standard case) a recordset that immediately connects to a database
DB_Iter	<i>first()</i> , <i>next()</i>	« <i>interface</i> » Abstract class that

---

## Chapter 11. The Bibliographic Component

**Purpose** Description of resources; with accent upon the classical bibliographical data (types). – Contains the core of the application domain objects.

**Modules** Biblio...

**External Dependencies** Depends on GUI for display interaction

**Internal Dependencies** Depends on Storage for persistence, on Common Control for a little configuration and interaction support, on Writer for formatting.

**Initialisation** Primary DB provides needed configuration and control data. Some bootstrapping and fallback mechanism desirable.

---

### 11.1 Description

**Note:**

- It is to be determined, how much of the application domain is to be catered for by this package-component.
- Demarkation against Storage and Writer seems clear to me,
- Candidates for related packages are IMHO Topics, Annotations, Holdings

The bibliographical description lies at the heart of Pybliographer, together with the features for annotation and organising of references.

This package provides:

- description up to recent ISBD developments, including multi-level description and material specific description
- extensible efficient indexing
- customisable, assisted data entry (spellchecking)
- extensible structured description (case analysis)
- annotation, integration of external documents w.r.t. searching
- task-oriented (writing) support
- maintenance of associations between bibliographical objects
- maintenance of classifications, etc.; efficient use thereof
- (some of the above should be expanded upon, in particular the following question should be answered: what is the specific *bibliographical* content, and what is just generic? Example: versions of a document, translations, reviews: how are the related?)
- extensions for collections, archival resources

---

## **Part IV. Appendices**

---

## Chapter A. Feature lists

---

### A.1 User Requirements

The following is a list of user requirements as given in a talk in the technical university of Chemnitz<sup>1</sup>

Id	Description	Related	See also
9001	Flexible access via various categories, viz. author, title, publication year, journal, and combinations thereof.		
9002	Import and export in various formats, viz. BibTeX, Medline, Refer, RFC1807, [MARC, MAB, ...]		
9003	Supports working groups, allows combinations of databases.		
9004	Allows to merge data from various sources, and to normalise it (e.g., with respect to differing ways to abbreviate a journal title ...)		
9005	Adapts to the various formatting requirements of different journals, easy development of new styles.		
9006	Extensible by local fields, to hold notes, order information, annotations, excerpts, summaries, etc.		
9007	Simple use during document preparation: allows to search the references from the editor/word processor (by author, title, e.g.), thus eschews the use of labels – the latter are to be used and provided only internally. → 6.2		
9008	Allow confidential data in shared databases (e.g., annotations of theses).		

Table A.1.: User Requirements (Chemnitz)

Other points that were mentioned on the mailing lists:

---

<sup>1</sup><http://archiv.tu-chemnitz.de/pub/2001/0013/data/anforderungen.htm> Note given there: These requirements transcend BibTeX's abilities; thus it should be used only as an export format.



### *A. Feature lists*

Id	Description	Related	See also
9010	Pybliographer starts too slow; with larger databases one has no indication of progress, and must wait for a long time.		
9011	Pybliographer holds the whole database in memory, and uses reportedly 70MB for a medium sized file.		
9012	RIS and INSPEC import formats are missing		

Table A.2.: Pybliographer Deficiencies

---

## **A.2 From Software Reviews**

A short table is found in the website “Bibliographic Database Managers”, prepared by Robert Sathrum, Natural Resources Librarian of HSU Library(sat:hsu01):

Id	Description	Related	See also
9100	Accommodate unlimited number of records		
9101	Create more than one database. Records in one database can be copied or moved to other databases.		
9102	Numerous input forms for different citation formats, e.g., journals, theses		
9103	Customizable field displays		
9104	Sort records by different fields, e.g., date, author, journal title		
9105	Sort at more than one level, e.g., first by author, then by journal, then by date		

Table A.3.: Database Structure Features

### A. Feature lists

Id	Description	Related	See also
9106	Import records downloaded from external electronic indexes and catalogs		
9107	Search Z39.50 compliant databases and automatically import records		
9108	Import records from word processor files		
9109	Import records manually		
9110	Detection of duplicate records		
9111	Spell check records as they are input		
9112	Edit records individually or globally		
9113	Create authority lists for selected fields, e.g., authors, journals, subjects		
9114	Define synonyms (cross-references) for related keywords		

Table A.4.: Database building capabilities

Id	Description	Related	See also
9115	Define specific fields to search		
9116	Search using keywords in authority lists, e.g., author, subject		
9117	Use of Boolean operators and word truncation		
9118	Search results retrieved as a separate file		
9119	Ability to mark individual records		

Table A.5.: Database search and retrieval capabilities

Id	Description	Related	See also
9120	Format references in multiple bibliographic styles, e.g., MLA, APA, CBE, to meet the requirements of scholarly publications.		
9121	Create independent bibliographies organized by subject		
9122	Create a manuscript and associated bibliography in one unified operation using a word processor		
9123	Export references in different file formats for use in other programs		
9124	Export bibliographies to the web		

Table A.6.: Database applications

### A. Feature lists

Francesco Dell’Orso (ors:bfs02) provides a long evaluation scheme from which we take the following 3.3 *Summary of Available Functions*:

1. Search
2. Remote search (Z39.50)
3. Print
4. Export
5. Sort
6. Input/Cataloguing
7. Global corrections
8. Import
9. Reformatting during import
10. Input via catching www pages
11. Manuscript formatting (also from within the wp)
12. Managing term lists
13. Thesaurus
14. Duplicates detection
15. Circulation (Loans)
16. Can activate external files and applications (OLE or OS’ Shell)
17. Compute
18. Graphic files management

Id	Description	Related	See also
9125	2 Installation and start: 1.Guided installation 2. Uninstall		
9126	2.1 Password required for 1. access 2. specific tasks 3. files 4. fields		
9127	15 Documentation 1. Reference manual 2. Tutorial 3. Context-sensitive Help screens 4. Error messages		
9128	8 Thesaurus		
9129	3.9 Macro		
9130	3.10 Mouse 1. right click 2. drag-and-drop		

Table A.7.: Dell’ Orso: Generalities

## A. Feature lists

Id	Description	Related	See also
9131	3.1 Network version 1. Multi-user network version 2. All functions available 3. Some functions can be protected (e.g. search, input, output available to many, but not management: rebuild, import ...) 4. One write access vs. many read accesses (read = search, print, manuscript formatting) 5. Many write accesses and many read 6. Read only version		
9132	3.2 Internet 1. can use different browsers 2. can automatically import/capture of WWW page 3. launch URL from within a record 4. can print HTML output 5. client Z39.50 6. can post database to the WWW that can be dynamically searched		
9134	3.4 Internal database management 1. Create 2. Rename 3. Copy 4. Delete 5. Undelete 6. Rebuild 7. Save 8. Compress (release space) 9. Record undelete 10. Revert global corrections 11. Statistics		
9135	3.5 Functions across different databases 1. search 2. print 3. global corrections 4. duplicate detection 5. manuscript formatting		
9136	3.6 Database subsets 1. retrieved records 2. highlighted records 3. marked records 4. virtual sets 5. imported set 6. duplicate set		
9137	3.7 Other library functions (e.g. acquisitions, periodicals control, circulation, statistics) 1. built-in 2. can integrate		
9138	3.8 Data type 1. textual (also ANSI 128-255) 2. numeric 3. graphic 4. sound 5. bar codes		

Table A.8.: Dell' Orso: Technology Substrate

## *A. Feature lists*

Id	Description	Related	See also
9140	4.1 Files [used, number etc.]		
9141	4.2 Internal database and record structure		
9142	4.3 Horizontal links: between databases, records, list entries		
9143	4.4 Hierarchical links (es. thesaurus, mother/sons records, text/notecards)		
9144	4.5 Ready, predefined record structure		
9145	4.6 Input worksheets 1. features 2. can be modified 3. can create others		
9146	4.7 Fields attributes can be changed; can be applied to other fields		
9147	4.8 Multiples (multi-value) fields 1. present/absent 2. specific input rules		
9148	4.9 Indexed fields for searching		
9149	4.10 Record number 1. system assigned 2. reserved, cannot be altered 3. user assigned 4. allows duplicate numbers 5. renumbering 6. is a sortable field 7. is a searchable field 8. can be displayed in printed output 9. reuse deleted numbers 10. can be alphanumeric		

Table A.9.: Dell' Orso: Database and record structure

## *A. Feature lists*

Id	Description	Related	See also
9150	5.1 Cataloguing Reference Standard		
9151	5.2 How to recall records for editing 1. list browsing 2. query search 3. while editing another record		
9152	5.3 Input must be preceeded by searching		
9153	5.4 Compulsory input within certain fields		
9154	5.5 Particular edit features 1. copy fields and/or records 2. symbols table 3. term lists 4. date stamping 5. undo 6. default values (session and/or permanent) 7. validation 8. automatic assignment of values 9. configure edit window 10. other		
9155	5.6 Spellchecker		
9156	5.7 Direct print of a record while editing it		
9157	5.8 Duplicate record 1. drag-and-drop 2. copy command 3. clipboard (formatted) 4. export/print on disk (also formatted)		
9158	5.9 Duplicates detection 1. fixed criterium 2. criterium can be defined		
9159	5.10 Editor 1. upper/lowercase conversion 2. cut/paste text 3. b/e record 4. b/e field 5. delete to End-Of-Field		
9160	5.11 Font editing: font, size, bold, italics, underline, super/subscript, small caps		
9161	5.12 Global corrections 1. add string (b/e) 2. locate/replace (words, case) 3. delete entire field content 4. move field content 5. change record type 6. can use wildcard		

Table A.10.: Dell' Orso: Input/Edit

### *A. Feature lists*

Id	Description	Related	See also
9162	6.1 Different ways. 1. direct copy (write to database) 2. batch import a) with reformatting filters a1) ready-made a2) un/modifiable a3) user can define more 3. can capture WWW pages 4. can download data ready formatted in its own proprietary format from specific data sources		
9163	6.2 Delimited/Tabbed structured input ASCII text file 1. fixed/variable number of fields 2. fixed/variable fields position 3. RT can be changed 4. multiple value fields allowed 5. fields also on different lines 6. can define field separator 7. can replace delimiter if embedded in field 8. can define end of record		
9164	6.3 Can import alpha/numeric data from a spreadsheet		
9165	6.4 Can import proprietary format files		
9166	6.5 Can import ISO 2709 format.		
9166	6.6 Can import MARC format file.		
9167	6.7 Tagged structured input ASCII text file File. Reformatting: 1. condition check 2. change RT 3. merge fields 4. delete/discard fields 5. field content parsing 6. add field content, strings 7. tolerate fields in variable position 8. upper/lower case conversion 9. replace text		
9168	6.8 Operability 1. read range of records rather than all 2. read one record at the time, confirm y/n 3. handle duplicates 4. preview 5. log file		

Table A.11.: Dell' Orso: Import

### A. Feature lists

Id	Description	Related	See also
9170	7.1 Different levels and approaches: 1. easy / expert 2. menu/command driven 3. browsing term lists / indexes 4. query expressions 5. browsings record list		
9171	7.2 Browsing the Search Index 1. entries show number of related docs 2. relationships (e.g. x-refs) between entries are displayed 3. direct selection of index terms and display of related documents		
9172	7.3 Query expressions		
9173	7.4 Natural language queries		
9174	7.5 Search strategy 1. can save and recall search expressions 2. can recall previous queries within the same session 3. can combine previous search steps		
9175	7.6 Can save and recall search result		
9176	7.7 Shows hits of each search expression component		
9177	7.8 Can print directly one or more records while in search mode		
9178	7.9 Refine		
9179	7.10 Advanced search features 1. best match, weighted terms, ranking 2. fuzzy, sounds like 3. hypertext-like		
9180	7.11 Response time		
9181	7.12 Instant display of retrieved records 1. short record list 2. one record at the time		
9182	7.13 Highlighting search terms in result ( + jump to next occurrence of term)		
9183	7.14 Indexing operation 1. automatic, real time 2. batch		
9184	7.15 Indexing techniques 1. any character string 2. word by word 3. phrase (adjacent words) 4. marked portions of fields		

Table A.12.: Dell' Orso: Search



*A. Feature lists*

Id	Description	Related	See also
9185	7.16 Scope of searching 1. one or more distinct fields 2. cluster of fields 3. full text = any field 4. same occurrence		
9186	7.17 Case sensitiveness		
9187	7.18 Diacritics		
9188	7.19 Can use and nest parenthesis		
9189	7.20 Priority within search operators and queries		
9190	7.21 Boolean operators. 1. AND 2. OR 3. NOT (unary) 4. AND NOT (binary) 5. XOR		
9191	7.22 Relational operators contains, <>, <, <=, >, >=, range, equal		
9192	7.23 Can combine boolean and relational operators		
9193	7.24 Truncated search 1. explicit or implicit 2. right 3. left 4. r/l		
9194	7.25 Can search by position: b/e field and/or occurrence		
9195	7.26 Search for not/empty fields		
9196	7.27 Internal wildcards (e.g. *?)		
9197	7.28 Can combine boolean, relational, parenthesis, truncation etc.		
9198	7.29 Adjacency and proximity operator		
9199	7.30 Search only within the same occurrence of a repeatable field		
9200	7.31 Search only within the same paragraph		
9201	7.32 Stopwords		
9202	7.33 Input allowed while indexing		
9203	7.34 Z39.50 Searching		

Table A.13.: Dell' Orso: Search *continued*

Id	Description	Related	See also
9294	9.1 Send output to printer, file, video		
9295	9.2 Real bibliography formatting software: offers large quantity of – ready and modifiable – styles, can create new		
9296	9.3 Structure of system display 1. short record list 2. one formatted record at the time 3. more formatted records		
9207	9.4 Report generator		
9208	9.5 "Subject list", i.e. list with sorted headings from record content 1. one level 2. more levels 3. headings only		
9209	9.6 Output file format 1. RTF 2. Word 3. WP 4. TXT 5. HTML 6. other.		

Table A.14.: Dell' Orso: Output and Print

Id	Description	Related	See also
9210	10.1 FL Selection 1. fields 2. subfields.		
9211	10.2 FL Can add text 1. in front of/after		
	2. regardless of field presence 3. depend-		
	ing on field presence 4. depending on field		
	content		
9212	10.3 FL Can distinguish among occur-		
	rences of a repeatable field: 1. by punc-		
	tuation -separators 2. because of position /		
	sequence number 3. can count them		
9213	10.4 FL Can produce tagged format out-		
	put (e.g. to export)		
9214	10.5 FL can display RT		
9215	10.6 FL can produce permuted indexes		
	(words in-out-and context)		
9216	10.7 FL offers conditional commands (IF		
	... THEN...)		
9217	10.8 Upper/lowercase conversion		
9218	10.9 Look-up tables to expand acronyms,		
	abbreviations, replace text		
9219	10.10 Contextual Record Preview		
9220	10.11 Text added in styles can be language		
	dependent for each record 1. text lists can		
	be modified 2. new lists can be added (new		
	language) 3. text can be present in various		
	fields		
9221	10.12 Check format syntax		
9222	10.13 FL Level of difficulty		

Table A.15.: Dell’ Orso: Formatting language to define output styles

Id	Description	Related	See also
9223	11.1 Scope 1. sorting records within the database 2. sorting records in output		
9224	11.2 Basic sort criterium for characters: 1. MS-Windows tables 2. program specific settings and/or tables 3. user defined table (es. ?= ae)		
9225	11.3 Sort keys on different levels		
9226	11.4 How to sort the database		
9227	11.5 How to sort for printing		
9228	11.6 Database sort is kept over sessions		
9229	11.7 Sort key length can be defined		
9230	11.8 Sort for printing belongs to output styles		
9231	11.9 Sort records in output using: 1. words 2. single occurrence of a multiple field 3. marked strings 4. portions of field and subfield 5. whole field		
9232	11.10 Edit can alter the sort value of a string		
9233	11.11 Conditional commands available for sorting		
9234	11.12 Sort keys derived from different fields		
9235	11.14 Sorting speed		
9236	11.13 Ignore initial articles and punctuation marks		
9237	11.15 Other		

Table A.16.: Dell' Orso: Sort

### A. Feature lists

Id	Description	Related	See also
9238	11.16 Sort can produce headings above sorted records ("subject bibliography")		
9239	11.16 Sort can produce headings above sorted records ("subject bibliography")		
9240	11.16.1 Sort keys are another item from headings, thus can match or be different		
9241	11.16.2 Headings might not be displayed within records		
9242	11.16.3 More than one level of sort key as headings		
9243	11.16.4 Sort of records under the same sort key		
9244	11.16.5 Sort occurrences of a repeatable field as headings 1. altogether 2. all separated 3. just one		
9245	11.16.6 Records referenced more than once by different sort headings		
9246	11.16.7 Sort headings can be formatted		
9247	11.16.8 Can produce indexes referencing records in the database by a short element (e.g. RN)		

Table A.17.: Dell' Orso: Sort *continued*

Id	Description	Related	See also
9448	12.1 Export formats 1. delimited: comma, tab, <CR>, other 2. tagged 3. ISO 2709 4. MARC 5. proprietary format of other database		
9249	12.2 Fields that can be exported 1. all 2. some 3. RT 4. RN		

Table A.18.: Dell' Orso: Export

### A. Feature lists

Id	Description	Related	See also
9250	13.1 Compatible wordprocessors		
9251	13.2 On-line contextual help		
9252	13.3 Can format more than on document at the time		
9253	13.4 Can generate bibliography from more than one database at the time		
9254	13.5 Entering placeholders within text 1 from within wp text (ad hoc tool bar and/or pull down integrated menu): 1a manually writing; 1b automatic insert 2 from within db: 2a ad hoc command (operational in-text placeholder); 2b via clipboard (in-text format has to comply with placeholder format)		
9255	13.6 Location for placeholders 1. main text 2. end/footnote 3. hidden text		
9256	13.7 Content and structure of placeholders 1. author name (1a any order) 2. title 3. keywords 4. date 5. RN 6. one or more, also truncated, string from any field 7. delimiters/markers can be changed 8. within the same style as many as RT		
9257	13.8 Different references same author same year 1. must differentiate within db (1990b) 2. program can distinguish them 3. must differentiate within text		
9258	13.9 Multiple citation (same or different authors) 1. can sort 2. can join: 2-5 3. can suppress repeated names 4. can insert text		
9259	13.10 Final in-text citation format is different from placeholder		
9260	13.11 Final in-text citation format is ruled by ad hoc style within db made up of: 1. citation number 2. author-date 3. other 4. shortened version of complete bibliographic reference 5. two styles: in-text vs. note 6. nothing		

Table A.19.: Dell' Orso: Manuscript formatting

### *A. Feature lists*

Id	Description	Related	See also
9261	13.12 Changes to in-text citation standard format 1. add text (in front of / after); 2. hide portion; 3. hide completely; 4. put only in the list, exclude in-text; 5. first different from its repetitions		
9262	13.13 Bibliographic reference list format ruled by db style different for each RT 1. generated within wp text 2. generated in a copy of wp text 3. settings remain for next session 4. can automatically exclude citations placed in notes 5. list can include references not shown as in-text citations		
9263	13.14 Word processor or db control reference list format 1. Paper size, margins, headers footers; 2. Heading; 3. Font, size; 4. Indent, line spacing; 5. Citation numbering, pre/suffix, justification		
9264	13.15 Duplicates are automatically removed from the list		
9265	13.16 Reference list sort order 1. citation order; 2. sort order defined within db A/D; 3. sort order defined within wp by the program		
9266	13.17 In-text citations and list references can be changed without changing wp text		
9267	13.18 Errors 1. during inserting and formatting; 2. user can act on the spot; 3. log file		
9268	13.19 Manuscript formatting takes place: 1. within wp text (1a same document 1b other document) 2. within db 3. in one step 4. in more than one step		
9269	13.20 More citations of the same reference can bear same number		

Table A.20.: Dell' Orso: Manuscript formatting

## A. Feature lists

Id	Description	Related	See also
9270	14.1 Fixed number	3.0.01	
9271	14.2 Lists' content is automatically derived	3.0.03	
	from db data (or can contain external data)		
9272	14.3 Lists are physically separated from		
	database		
9273	14.4 List reflects records content in real	3.0.03	
	time		
9274	14.5 List can be directly edited		
9275	14.6 When list entries are edited, records		
	change		
9276	14.7 New entries are validated (go list:	3.0.06	
	new, old, probably a duplicate)		
9277	14.8 List entry can contain its own sup-		
	plementary data: note, abbreviation, date,		
	compiler, x-refs		
9278	14.9 List can be printed		
9279	14.10 Import external data into the list		
9280	14.11 Lists are useful for input	3.0.06	
9281	14.12 Lists are useful for searching	3.0.05	
9282	14.13 List entries show total number of re-	3.0.09	
	lated documents		
9283	14.14 Lists can be shared among different		
	db		
9284	14.15 Where and how they are used 1.		
	browsing, display related records 2. search		
	expressions, pick up terms from one or an-		
	other 3. input, pick up terms from one or		
	another 4. output		
9285	14.16 How lists are created and updated		
	1. input: new entries automatically update		
	the list 2. ad hoc command to edit lists out		
	of records 3. import 4. as external text file		
9286	14.17 How lists are printed 1. from the		
	outside as text file 2. from the inside by ad		
	hoc printing function		

Table A.21.: Dell' Orso: Term/Entry list, authority file



---

## A.3 Biblioscape

*Organize references with folders, dynamic folders, etc.*

**Folder** Add references into folders. One reference can be put into multiple folders without creating a duplicate. Organize references into folders with drag and drop.

**Dynamic folder** Organize and save queries into a tree structure. All references meeting the search criteria will be listed under a dynamic folder.

**Indexed search** Return search results in a couple of seconds no matter how big the database is. One line search works the same way as most Internet search engines. Supports logical searches, fuzzy searches, etc.

**Import filter** Bibliographic data from any data sources can be imported with a proper import filter. User can create new or edit existing import filters.

**Output style** References can be displayed any any style like MLA, APA, etc. A large number of styles are provided for different journals. Users can also create new ones.

**Cross linking** Link a reference to other references in the same database. You can define a relationship for the links, like "Supportive", "Contradict". You can also add comments for each link.

**Navigation view** A reference can be displayed in an organizational chart where each node can lead to related records.

**Formatted preview** Display a reference as formatted text according to the active output style.

**Live preview** Display data fields of the selected reference in a grid without opening it. Changes made to the data will be saved to database as you move to another record.

**Graphics and OLE** If a reference has associated graphics and OLE objects, add them to the Document field. The document field can be used to store the full text of a reference.

**Field lookup** List all unique values along with the number of occurrences of a data field. All data fields with possible repeated values can be shown in lookup view. These include Author, Keyword, Publisher, Language, Country, Subject, etc.

**Recycle bin** All deleted references are put into the Recycle bin. You can recover them from the Recycle bin or remove them permanently from the Recycle bin.

**Advanced search** Query any data field with a visual query builder.

**Find and Replace** Search for a word or phrase and limit the search to a data field or all data. The same is true for the Replace operation.

### *A. Feature lists*

- Sorting Sort a column by clicking on the header. Click again and the column will sort in reverse order. User can also define multi-level sorting.
- Filtering Define a filtering criteria with a visual filter builder and apply the filter to any dataset.
- Term list Users can keep frequently used phrases in a term list. Terms can be organized in the list by category.
- Move field The content of a data field can be moved from one field to another.
- Global edit The content of a data field can be changed at once for all selected references.
- Eliminate duplicates Duplicate records can be found and removed. Fuzzy search is supported for finding duplicates.
- Analyze references Data fields in the reference table can be analyzed for data distribution.
- SQL commands Users who are familiar with SQL can query the database directly with SQL commands.
- Report A built-in database report writer will a print data report, including a subject bibliography grouped by keyword, author, year, subject, etc.
- Format papers to generate citations and bibliography*
- Format a paper Convert the temporary citations of a document into formatted citations and a bibliography.
- Unformat a paper Convert a Biblioscape formatted paper back to unformatted form (with temporary citations) so citations can be added or deleted before the final formatting.
- Word support Full integration with Microsoft Word, Biblioscape menus and toolbar can be added to the Word menu and toolbar system.
- WordPerfect support Full integration with Corel WordPerfect, Biblioscape menus and toolbar can be added to the WordPerfect menu and toolbar system.
- Other word processors Biblioscape methods for word processors integration are published and open to all word processors that support DDE.
- HTML support Biblioscape can generate formatted papers in HTML format. A hyperlink can be created automatically between an in-text citation and its reference in a bibliography.
- Natural citation Use words or phrases to uniquely identify a reference in a temporary citation instead of using a Reference ID. If references are moved into another database, temporary citations don't need to be changed.

## *A. Feature lists*

**Cite while you write** Use BiblioSidekick to display references in a small, always on top windows. While in a word processor like Word or WordPerfect, just drag and drop the selected reference in the place where you want to cite it.

**BiblioWord** A full featured word processor inside Biblioscape. Just drag selected references from a panel on the right when you want to cite. BiblioWord supports live spelling check, thesaurus, tables, graphics, OLE, multi-level undo, etc.

### *Access the Internet to capture bibliographic data, Web pages*

**Remote databases** Access thousands of remote bibliographic databases on the Web with an integrated Web browser. These sites include university sites, commercial databases, and government sites. Most of them are free.

**Capture references** Search web based bibliographic databases from inside Biblioscape, click a button to capture search results into a Biblioscape database with the right import filter. New import filters can be created by users.

**Capture Web pages** Research on the Web with the Biblioscape integrated browser, capture a web page into a Biblioscape References table or Notes table. All words in the Web page will be indexed for future search. Graphics and links are captured along with the page.

**Resources** A directory of bibliographic resources on the Web. Each entry listed has an associated import filter. The local Resources list can be expanded and edited by the user.

**Web directory** Biblioscape Web site lists a collection of sites valuable to researchers. Web sites are organized by subject. Bibliographic databases are the main part of the listing. Although other types of Web resources are also listed.

**Z39.50** Most Z39.50 enabled bibliographic databases also have a Web interface, Biblioscape's integrated Web browser can be used to search such sites and capture search results directly into a database.

**Link to a note** Easily create a link between a note and a Web site.

### *Take notes and link them to references, tasks, web sites, etc.*

**Tree structure** Organize notes in a tree structure. Note's position in the tree can be rearranged by drag and drop.

**Indexed search** Find your note fast with indexed search. Each word in your Notes database is indexed for super fast search. The search words are colored in red on the hit page. Indexed search supports logical operators, wildcards, fuzzy search, etc.

### *A. Feature lists*

**Advanced search** Limit your search to a data field like Date Created, Keywords, etc. Build complex searches with a visual query builder.

**Format text** The text in your note can be formatted with all the standard options, including fonts, color, background color, superscript, subscript, paragraph alignment, bullet list, number list, etc.

**Link** Each note can be linked to other notes, references, tasks, catalog items, Web URLs, local files, etc. Double clicking on a link will take you to the linked item.

**Web capture** Notes can be used to organize captured Web pages. All the graphics and hyperlinks of captured web pages can be properly displayed.

**Table support** You can insert tables in your notes. Additional rows can be added and deleted.

**Find and Replace** Standard Find and Replace tools for finding and replacing text in your notes.

**Graphics and OLE** Graphics can be added to your notes. OLE is also supported. Therefore, you can add chemical structure drawings, spreadsheets, CAD drawings, etc. in your notes.

**Table view** The notes can also be displayed in a table besides the default tree view. Notes can be sorted and grouped in a table.

**Keyword lookup** Each note can have associated keywords. These keywords can be displayed in a lookup list along with its number of occurrences. Double clicking on a keyword will retrieve all related notes.

**Spelling and thesaurus** A powerful spelling checker is included. Additional dictionaries can be downloaded for all major European languages. A thesaurus is also included to help the user to find the right words during writing.

**Icons** Each note can be assigned a different icon to distinguish it from other notes.

**Export** Each note can be exported to a file in RTF or HTML format.

#### *Manage tasks and organize your research ToDo list*

**Sort tasks:** Click on the column header to sort tasks, click again to sort in reverse order.

**Group tasks** Group tasks by Priority, Status, Date Created, etc. ]

**Task progress** Track the progress of a task by marking its percentage completed.

**Task creation** Create tasks inside References module, and add selected references into the Description field of the new task.

**Link to a note** Create a link between a selected task and a note.

## *A. Feature lists*

Advanced search Search tasks with a visual query builder.

### *Draw a chart to present your ideas*

Flow chart Draw a flow chart with an easy to use chart editor.

Knowledge map Draw a chart and link a chart object to other modules. For example, double clicking on a chart object will open a group of references, tasks, notes, etc. A SQL query can be associated with each chart object. A knowledge map can be built with such associated queries.

Tree structure Organize your charts in a tree structure. The position of each chart in the tree can be rearranged by drag and drop.

Link to a note Create a link between a chart and a note.

Zoom Display options like zoom in and zoom out, actual size, and fit to screen are supported.

Icon Each chart can have an icon associated and displayed.

Shape and color The shape and color of each chart object can be customized. The label text can be displayed in different fonts and colors.

Connectors Chart objects can be connected with a flexible connector which can be curved. A connector can have its own label, font, color, size, different sources and destination arrows, and link points.

### *Manage a library without a steep learning curve*

Catalog Manage library collection data into 56 data fields, organized into several groups including Bibliographic, Holding, Request, Order, Serial, and General.

Serials Manage serials and related activities including tracking, routing, etc.

Circulation history Search, sort, and group circulation data. Display circulation activities by borrower, status, subject, etc.

Check Out Check out books for library patrons, add notes, easily change due dates.

Check In Check in books returned by borrowers. Automatically reminds librarian about Hold status.

Renew Renew books for borrowers, add a note. Find renewed items by ID or title.

Hold Put a hold on a checked out book. Show a reminder when that book is returned.

Interlibrary Loan Manage interlibrary loan requests, track loan status, log shippings, etc.

Borrowers Manage borrower's information (address, phone, fax, email, etc.).

### *A. Feature lists*

- Lenders Manage lender's information (contact's name, phone, fax, email, notes, etc.)
- Suppliers Manage supplier's information (address, phone, fax, email, notes, etc.)
- Sort Click on any column header to sort then click again to sort in reverse order.
- Group Group data by drag and drop. Data can be grouped at multi-levels by any data field.
- Field chooser Choose which data fields to include in the data grid by drag and drop.
- Report and print Build or customize data reports with a powerful report builder. Users can create new reports with a wizard. New reports can be easily added to the menu system. Reports can be previewed, printed, or saved as a files.
- Web enable your bibliographic database with one click*
- Web publishing Publish databases on the Web with BiblioWeb server. No other web server required. Runs on any Windows 95, 98, Me, NT4, 2000 machine.
- Indexed search Search references with a powerful search engine. Enter search commands like you do with a Web search engine. Supports search keywords AND, NOT, OR, LIKE, NEAR, Wildcards, etc.
- Advanced search Limit searches to certain fields. Build complex queries with up to 3 conditions.
- Add references Users with a Write account can add new references to the database using a web browser.
- Edit and delete Users can edit or delete their own references over the web.
- Import Import references over the Web with the right import filter, so you don't need to enter references one by one.
- Hyperlinks Search results are displayed with hyperlinks. Clicking on the hyperlink will trigger a new search for related items.
- Style Marked references can be displayed in any of the output styles that exist in Biblioscape.
- Export Marked references can be exported in several formats to be easily imported into other programs.
- Format papers Users can even format a paper over the Internet. Temporary citations in a document will be converted to formatted citations and bibliographies.
- User forum Includes a user forum application, so you can host a web based forum without extra cost.

---

## Bibliography

[ors:bfs02]

[sat:hsu01]

[KK:SWD90] In: *Die Schlagwortnormdatei, Entwicklungsstand und Nutzungsmöglichkeiten*

[1] DEPARTMENT OF DEFENSE CHIEF INFORMATION OFFICER: Design Criteria Standards for Electronic Records Management Software Applications / U.S. Department of Defense. URL <http://www.dtic.mil/whs/directives/p50152s2.pdf>, 2002. – Forschungsbericht

[2] FATTAHI, Rahmatollah ; PARIROKH, Mehri: *Restructuring the Bibliographic Record for Better Organization and Representation of Knowledge in the Global Online Environment*. 2002. – URL <http://www.um.ac.ir/~fattahi/ISK0/abstract1.htm>. – Paper presented at the 7th ISKO International Conference, in Granada, Spain (10-13 July 2002) on Challenges in Knowledge Representation and Organization for the 21th Century: Integration of Knowledge across Boundaries,

[3] GÄTJENS-REUTER, Margit: *Ablage : d. Organisation d. Information ; [Philosophie, Kunst u. Know-how d. tägl. Ablegens u. Wiederfindens von Papier, Filmen u. elektron. gespeicherten Daten]*. 1988. – ISBN 3-409-19106-2

[4] HALLER, Klaus: *Katalogkunde : Eine Einführung in die Formal- und Sacherschließung*. 3., erw. Aufl. München : Saur, 1998. – 91 avf 1332(3)+1

[5] HENSEN, Steven L.: *Archives, personal papers, and manuscripts : a cataloging manual for archival repositories, historical societies, and manuscript libraries*. 2nd ed. Soc. of American Archivists, 1989. – ISBN 0-931828-73-2

[6] HOFFMANN, Heinz: *Schriften des Bundesarchivs*. Bd. 43: *Behördliche Schriftgutverwaltung : ein Handbuch für das Ordnen, Registrieren, Aussondern und Archivieren von Akten der Behörden*. Boppard am Rhein : Boldt, 1993. – ISBN 3-7646-1924-4

[7] HYLTON, Jeremy A.: *Identifying and Merging Related Bibliographical Records*. Cambridge, Mass., MIT, Diplomarbeit, Juni 1996. – URL <http://www.python.org/~jeremy/pubs/thesis/index.html>. – MIT-LCS-TR-678

## *Bibliography*

- [8] IFLA STUDY GROUP ON THE FUNCTIONAL REQUIREMENTS FOR BIBLIOGRAPHIC RECORDS: Functional Requirements for Bibliographical Records : Final Report / International Federation of Library Associations and Institutions. URL <file:frbr.pdf>, 1998 (19). – UBCIM Publications New Series. – ISBN 3-598-11382-X
- [9] JOB-SLUDER, Kirk ; HANEK, Greg ; ZOANG, Hong: User needs for bibliographic software / OpenOffice.org. URL <http://php.indiana.edu/~csluder/OpenSource/>, 2003. – Forschungsbericht
- [10] KNORR, Dagmar: *Informationsmanagement für wissenschaftliche Textproduktionen*. 1998. – ISBN 3-8233-5351-9
- [11] MAXWELL, Robert L. ; MAXWELL, Margaret F.: *Maxwell's Handbook for AACR2 : Explaining and Illustrating the Anglo-American Cataloging Rules and the 1993 Amendments*. rev. Edition of Handbook for AACR2 / by Margaret Maxwell, 1989. Chicago : American Library Association, 1997. – ISBN 0-8389-0704-0
- [12] MCHARITY: *The opportunity for a flexible bibliographic format*. 18 Dezember 1996. – URL <http://www.vendian.org/mncharity/dir3/flexible\protect\T1\textunderscorebib\protect\T1\textunderscoreformat>
- [13] SCHNEIDER, Wolfram: *Ein verteiltes Bibliotheks-Informationssystem auf Basis des Z39.50 Protokolls*. Berlin, Technische Universität Berlin / Konrad-Zuse-Zentrum für Informationstechnik, Diplomarbeit, Juli 1999. – URL <http://www.de.freebsd.org/~wosch/lv/diplom/>
- [14] STEPHAN, Werner (Hrsg.) ; Deutsches Bibliotheksinstitut (Veranst.): *Die Schlagwortnormdatei, Entwicklungsstand und Nutzungsmöglichkeiten : Vorträge eines Kolloquiums zur Schlagwortnormdatei (SWD) in Frankfurt a.M. am 5. und 6. Oktober 1990*. Bd. 90. Bundesallee 184/185, Berlin, 1990. (DBI-Materialien)
- [15] SVENONIUS, Elaine: *The Intellectual Foundation of Information Organization*. Cambridge, Mass. : MIT Press, 2000 (Digital Libraries and Electronic Publishing). – ISBN 0-262-19433-3
- [16] VOGT, Carol: *About Personal Bibliographic Management Software*. Web site. may 2000. – URL <http://ist.uwaterloo.ca/ew/biblio/>. – Accessed: Tue, 11 Feb 2003 21:30:57 +0100