Jul 19th, 12:00 AM

# 13 Years of allegro: questions, demands, users

Bernhard Eversberg
*Braunschweig Technical University*

## 13 Years of *allegro* : questions, demands, users

Eversberg, Bernhard.
Braunschweig Technical University Library,
Pockelsstr. 13,
D-38106 Braunschweig, Germany

### Preliminary remarks

This paper is supposed to present a broad survey of what *allegro* is and what it can be used for. I try to do so by answering 5 basic questions (Part A) and by indicating how *allegro* responds to 7 fundamental demands of today's library computing (Part B). Part C then lists ten "library communities" within which *allegro* has become a factor.

*allegro* is a database system designed for variable data, developed since 1980 at Braunschweig Technical University Library. It is specifically adaptable to library needs and can accommodate formats like MARC, including authority control. Modules for circulation and acquisitions are fully integrated. High degrees of efficiency and adaptability combine with low demands on systems resources.

Library computing has a history of conflicting demands, the conflicts being perhaps more acute than in many other areas of computer use.
The 70s were dominated by a conflict between speed and functionality: a program could be either fast or convincingly up to its task but not both. In any case, computing was unaffordably expensive for most libraries, so the conflict bothered only a few. Despite steadily improving price-performance ratios, most of the 80s then saw a conflict between performance and cost: systems were either good but costly, or cheap but poorly performing.
Recent years have seen an ongoing explosion of demands along with hardware prices plunging ever lower and raw systems performance skyrocketing. Computer systems of adequate power are now within the reach of many more institutions than ever before. Today, we have a conflict between ease of use and flexibility as well as - not just in library computing - a persisting and very general conflict between expectations and reality. A multi-billion dollar industry (that did not even exist a mere 10 years ago) must convince ever more customers that they need ever more powerful machines because those that are adequate for most jobs are no longer profitable. The industry must persuade people to take a

sledgehammer to crack nuts. That way, too many, not truly computer-literate, customers cannot realistically assess their needs and project their goals but get seduced into thinking that computing is child's play, if only they buy the right system. At least for library work this is still just not true and one may wonder how soon it will become so.

## A. The five basic questions

### 1. Philosophy, Paradigm

*"What are the concepts and ideas that characterize the system as a whole?"*

For *allegro*, there are separate answers for the aspects of database use and database management:

**Use**
- no command language; direct action by as few command keys as possible
- browsing (index and short display are the most important tools)
- the most frequently needed functions must be the fastest.

**Management**
- all power to the user in terms of adaptability
- high availability of data (for example, no downtime during updates)
- fast update, reorganization, and recovery.

### 2. Systems requirements

*"What hardware and systems software does it take to run allegro?"*

Nothing more is needed than what everybody has or can afford: industry standard PCs and any DOS from 3.3 upwards. Version 12.2 still operates on 286 machines with 640K, the OPAC program even on XTs. Any LAN can be used for multi-user operation, even the low-cost peer-to-peer networks have proved to be sufficiently reliable. Disk storage requirements are around 600 bytes per record for catalog data (index included, ca. 20 keys per record, with up to 72 bytes per key). In other words: 10 MB of disk space can accommodate well over 15.000 records.

## 3. Software principles

*"What are the foundations of the software?"*

The top concern in development has been to have complete control over all parts of the software: there is no dependence on any third party, and all code is in C and documented in English. In particular, the software is not based on any commercial database product.

The concept is not a relational model, therefore there is also no SQL interface. However: an API (application programmers' interface), empowers C programmers to add functionality to their databases. Proficiency in C is, however, a rarity in library circles.

For some time now, there has been much talk about the client-server model of data processing. This concept requires a database server, either integrated into the file server or standing beside it. On self-contained LANs, this doesn't make too much sense but adds to the cost. *allegro* works more efficiently without it. It is in open networks where it makes sense, but there must be a well-defined interface, such as the proposed SR or Z39.50. For the index-related functions of *allegro*, a database server module indeed exists. An SR module capable of accessing *allegro* databases is therefore realistic, but it hasn't been programmed yet.

## 4. Data structures

*"Are internal data structures accessible for programmers?"*

Primary data structures are relatively simple and fully documented. The standard formats (MARC, MAB, Unimarc, Pica) can be accommodated. Index files are fairly complex for reasons of efficiency but need not be dealt with directly because their structure is parametrized, and the parameter files are open.

## 5. Functions

*"What functions does the software support, what modules does it contain?"*

There are six modules related to particular areas of library work, and five programs to support database management (the methods of operation indicated in parentheses):

5.1 Library related functions

1. Cataloging (function keys, commands)
2. Subject Cataloging (menu, function keys, commands)
3. OPAC (menu, function keys)
4. Full text search and export (batch procedure)
5. Monographic acquisitions (menu, function keys)
6. Circulation (menu, function keys)

5.2 Database management functions

| | | |
|---|---|---|
| 1. | Conversion | programmable reformatting of non-*allegro* data |
| 2. | Sorting | alphanumeric sort of large files of variable records |
| 3. | Updating | merge new data into a database, optionally replacing records |
| 4. | Exporting | programmable reformatting of *allegro* data into other structures |
| 5. | Indexing | build a database (mainly the index) from basic files |

All of these services can be controlled from the CockPit, a kind of data and program manager.

**B. The 7 fundamental issues**

In development work, the problem is that the various issues cannot be dealt with one after the other. One must keep all, or most of them, in mind all the time. The developer might use this list to aid his memory:

**Openness**
- Networks
- Standards
- Integration

**Flexibility**
- Local maintenance
- Adaptability
- Import/Export

**Security**
- Data protection
- Privacy
- Resilience/Recovery

**User interface**
- Simplicity
- Intuitiveness
- Adaptability

**Availability**
- Ease of access
- Rights and royalties
- Continuity
- Documentation
- Know-how distribution

**Resources**
- Cost <-> Performance
- Platforms
- Range of applicability

**Performance**
- Real-time update
- Rapid access
- Productivity

We do not try to group the seven according to priority (it's just not possible) but alphabetically:

**Availability**

When a software is made available on any larger scale, a well-defined version must exist at any time: it must be trouble-free for the customer to obtain and install it. Braunschweig Technical University Library sees this as its responsibility. In 1990, the Ministry for Research and Culture in Hannover officially assigned the *allegro* development work to the library as a special task with importance to the state of Lower Saxony. This means it is, and

will most probably remain, a non-commercial, not-for-profit product. The basic package currently sells for as little as DM 350 in Germany and DM 550 abroad.

Development work cannot be done in isolation, it needs frequent feedback from the field. Therefore, development and distribution have worked hand in hand from the beginning.

The current *allegro* version is 12.2, to be replaced by 13 by the end of 1993. The Systems Manual is a fairly complete documentation, supplemented by the quarterly *allegro news*.

Braunschweig Technical University Library is the sole supplier. No third parties have any rights or claims on any parts of the distributed software, so no royalties have to be observed and no other software needs to be acquired in order to use the system.

There is now an additional distribution channel beside the mail: the FTP server. It can be dialled from every point in the Internet, and registered customers can download the most current releases of all parts of the system as well as "beta versions" of new programs.

The development group make every effort to keep new versions compatible in the sense that existing databases and parametrization need not be reorganized in any way.

Since working creatively with *allegro* is anything but child's play, there has to be assistance. Newcomers can now work their way into the system with the help of the new textbook and training diskette. From 1990 to 1992, before the textbook was published, several hundred librarians attended training courses in Braunschweig. The material produced for these courses was integrated into the textbook, part of which is available in English.

A growing number of *allegrologists* are in touch with the development department through Internet mail. There are infrequent, informal, and often unofficial meetings in Braunschweig and elsewhere, and the official annual Experts' Convention in Braunschweig is attended by about 60 active users (as many as can be accommodated). An *allegro* database of all known users exists and is continuously updated. This is consulted every day to refer callers to somebody in their neighbourhood who is likely to be helpful. There is a number of circles in special branches of librarianship, for example church libraries, public libraries, sinologists, media centers, archives, and others (see Part C) who have established themselves and act as self-help groups, and sometimes exchange their databases along with their experience. Each of these groups has at least a few *allegrologists* who are happy to share their know-how and experience.

**Flexibility**

German libraries (or librarians?) are exceptionally individualistic. Foreigners frequently find it hard to believe that there is no universally accepted standard format. Some advocate the Deutsche Bibliothek's MAB as a standard, but it is not nearly as rich as US-MARC or Unimarc. No wonder there exist lots of dialects and specialized versions, but also totally different formats, with Pica as the most prominent example. In this chaos, the logical approach was to create a data definition language to configure *allegro* for any format and a data manipulation language for conversion between formats. In fact, the latter consists of two different languages: the import language to convert any other format (*allegro* or not) into the one of choice, and the export language to produce output in the widest possible variety. The export language covers not just print and download production but screen display and even index generation.

The data definition language is being refined now for version 13. With this, *allegro* will be able to accommodate any MARC format as its internal format, as well as MAB or Pica. "Field descriptors" will specify all aspects and properties of every field that is part of the format. A US-MARC configuration has been produced already for the test phase and has been made available to "beta testers".

The export language is the most difficult part of the system. The new textbook covers it in two large chapters, and the greater part of the training material consists of example parameter files, demonstrating all commands and features of the export language.

The chapters 10 and 11 of the Systems Manual describe all details of the export and import languages. These two chapters are the first which have been completely translated into English and are now under critical review by a competent user in Oxford.

**Openness**

The abilities to convert foreign formats and to produce them by export are important aspects of openness. At Braunschweig Technical University Library, being a member of Pica Lower Saxony, we had to make it our concern to convert Pica data into *allegro* data. We can now download result sets and single records from the Pica database and, right after the session, have the downloads converted and merged into the *allegro* catalog. The same is feasible, and is being done, in other networks. But "openness" means a lot more. (For multilingual

operation see "User Interface".)

The import and export languages, as was said in the previous section, are difficult but completely documented thus making *allegro* an open system for configuration and maintenance in the field and by the expert user.

For the more ambitious user, who also happens to be a C programmer, there is the API (application programmers' interface) which makes it possible to add functionality to the database core. This is also the approach by which the circulation and acquisition modules have been constructed. Being extensions to the database core, these programs enjoy the full searching capabilities of the cataloging module and are fully integrated with the catalog software. Also, the circulation and acquisition functions benefit from the functionality of the export language to make them flexible and adaptable.

If the future is going to be an OSI world (which remains to be seen), some further programming tasks will have to be faced. An interface with some SQL functionality in the sense of the Z39.50 standard (the Search & Retrieve protocol) could be constructed as a special extension on the API level (see A.3). Requirements on the index can be met by the existing export language potential.

The present, being a TCP/IP world, calls for accessibility by the TELNET protocol. This is reality now: the Braunschweig OPAC is open to the Internet world under 134.169.20.3, representing a SUN workstation. The *allegro-X* version is, however, not yet fully compatible to the MS-DOS version. Both are supposed to converge with version 13 (see also under "Resources").

**Performance**

Speed and storage economy have been high on the priority list from the start. Braunschweig Technical University Library is looking back on almost 250 years of thrift and even long periods on the brink of poverty. For software development, this unbroken tradition implied there was to be no waste in processor power or disk space.

The preliminary remarks mentioned a conflict between speed and functionality. In many cases this has been "resolved" either by buying a more powerful machine or by sacrificing some (often rather much) disk space. Neither of these approaches appeared acceptable, the solution therefore had to be different. The solution that was found is at the core of *allegro*'s success; it means that we can now host relatively large databases on relatively small machines, with relatively good performance. And it means that, on the more resourceful

machines now becoming affordable, we can achieve much more. Or rather, we are not forced to scramble for the latest technology all the time just to keep up.

Besides the main database program PRESTO, combining index browsing, data entry and editing, there are the programs INDEX (to build or rebuild indexes) and UPDATE (to merge new data into an existing database, optionally replacing records that match a primary key). These also need to be efficient. Of course, the larger a database, the longer it will take to index it, but the time needed rises linearly, not exponentially. Theoretically this is the best possible behavior for this kind of task. As an example, with the Braunschweig OPAC of 230,000 records it takes about 5 hours to produce the index of 4.2 million entries: this was recorded on a 486/66MHz workstation and a 386/33MHz server under Novell 3.1.

UPDATE can be run while the database is in full use. In fact, we often run it in peak hours just to see if we get into performance trouble. So far, there has been none. "Peak hours" means there are 50 to 65 active connections on the server, about 5 to 10 of which are using the database being updated.

As a result, we have high availability of the databases. In fact, during the past 2½ years, we have had no downtime during opening hours, except one from a power failure. If an index needs to be rebuilt, a rather rare event, it can always be done overnight, up to a size of about 400.000 records. The worst case would be the destruction of the index of a very large database: then, the index can be rebuilt in stages, the most important keys in the first night, and the size could then be well over 1 million records. This means that databases of that size (really big by PC LAN standards) can still be made available overnight. This worst case will very rarely be encountered, for there is also a restore facility that can repair minor damage.

## Resources

As was already stated, *allegro* can still run single-user applications on low-end machines. The software itself still needs no more than 2 MB of hard disk. The number of libraries who run databases on LANs has increased sharply over the last 12 months, many of them operating on one of the cheap peer-to-peer networks. Several universities now have campus-wide Ethernet access to a server running under Netware 3.11. (Diskless workstations can be used in such environments.) In Braunschweig, there are between 2 and 7 connections from outside the library at any one time.

This range of platforms on which *allegro* can be implemented is now being extended by

*allegro-X*, the UNIX version. Since all program code is in C, most of it could be ported without trouble. Trouble arises almost exclusively with screen and keyboard routines. Here, UNIX is a totally different world. The version that is now running on a SUN is a temporary solution, but version 13 should be available on SUN by September 1994. Versions for other UNIX systems will then still require some additional effort, and it will clearly be a task for UNIX and C specialists.

The Braunschweig OPAC on SUN is open to virtually all networked terminals and PCs anywhere in the University or on the Internet. (Everybody can "telnet" to it at 134.169.20.2, the userid is OPAC, the password is also OPAC.) Given the flexibility of the software and its low cost, not just OPACs but many kinds of special databases could then be made available to large, and/or widely-dispersed, clienteles. One early example is the German database of the EROMM project (European Register of Microform Masters): it is hosted on a computer at the Göttingen State and University Library and can be reached by any modem connection. This is currently using an Omniware gateway wired into a Netware system in Göttingen which can serve only two connections at a time. A UNIX implementation could serve many more.

**Security**

This topic is way down in the alphabetic sequence, but high in importance.

The software itself very rarely crashes, and when it does, in most cases no harm is done to the data. We even had server blackouts due to power surges, but almost always Netware got itself up again automatically and there was no loss. The point is that a crash will have no effect at any time except in the middle of a write operation. This will most probably disable the index, but not the other files. The "LOG file" then comes to the rescue. One will certainly (?) have a not too old backup copy of the database. This must be retrieved first, then the UPDATE program will be able to merge the current LOG file into it. This is called a "playback" operation because it replays all transactions that have occurred between the last backup and the crash.

To encourage users to do backups regularly, the CockPit supports these functions very conveniently. It also reduces maintenance jobs, like compaction (elimination of unused space) and index recreation, to the press of a button.

**User Interface**

Talking about software in general, the user interface is the most complex issue because at the "man-machine interface" artificial intelligence meets natural intelligence, and these two are not all that compatible. No wonder, then, that contemporary graphical user interfaces (GUIs) tend to consume more memory and processor power than the application itself and to exceed all other parts in complexity.

It is necessary to distinguish between two kinds of "users": library patrons and librarians. The librarian's *allegro* interface as it stands now will get disastrously poor ratings from every Windows user, for example. As stated above, priorities lay with performance and memory economy, and this conflicts squarely with ambitious user interface design.

We made some effort, however, to create a user-friendly OPAC: about 40 libraries have already abandoned cards in favor of it. The OPAC has menus. The more experienced users can, however, avoid their use and activate all functions by single function keys or combinations.

A user interface is often looked upon as a shell that seals a kernel from the outside world. The kernel is the really important thing, however. Design of the interface should follow, not precede, the design of the kernel: the opposite tends to be the rule, particularly with commercial software. A software firm must try to get as much business as they can handle, and potential customers can best be convinced by bright interfaces. The qualities of the kernel can never be judged from outward appearances and certainly not from short demonstrations. Software evaluation has become a more difficult job than ever because software today is immensely more complex than it used to be only 10 years ago. Interfaces can obscure rather than illuminate the true qualities of a software.

All this is not said as a convenient excuse for *allegro*'s lack of a good user interface but as a general caveat for everybody who is about to make software decisions.

The CockPit is only half a solution, or less. It helps a lot but it is not too intuitive and it is only a file and program manager. It doesn't help with data input and editing, and much less with the really difficult tasks of configuration and parametrization. For the latter, a menu-driven interface would have to be an exceptionally large program. It is more likely that a kind of compiler with debugging functions will be developed to help with the construction of parameter files. As yet, there is no such software under development, but demand for this is certainly growing.

The circulation and acquisitions modules do already have a lot of menu functionality, as does the subject analysis program and, to some extent, the OPAC. All this is, however, not the grand solution.

The language of operation is German or English. Since all menu texts and messages are in text files, the software can be regarded as multilingual.

For a long time now, there has been concern about how to keep MS-DOS and UNIX versions from drifting apart. This is the main reason why the Microsoft and Borland tools for object oriented interface development have not been employed. They confine themselves to MS-Windows: what we need is a cross-platform development toolbox. We are about to pick one out of a small number of such devices that are available now, one that allows platform-independent programming, in the sense that one and the same source code can be compiled at least for DOS, Windows, and UNIX character mode. OS/2 and Motif would be welcome, too, but the first three are the most important for the near future. It remains to be seen, however, what the price will be in terms of overhead: the 286 will surely be the first victim.

## C. *allegro* Communities

There are some 10 communities of *allegro* users. Some are just groups of libraries with similar profiles but little or no cooperation whilst others have developed quite some "community life" (meetings, sharing of know-how and data), either nationwide or within smaller areas.

### 1. State libraries

All German libraries with either "state" in their names or with state library functions are using *allegro* for some purpose or other, mostly for special bibliographies and projects. The alphabetical list is as follows: Berlin, Bremen, Frankfurt, Göttingen, Hamburg, Leipzig, Munich, Wolfenbüttel. The largest database to date is being maintained in Hamburg: all 1.4 million titles of the North German Network (Norddeutscher Verbund) have been implemented on a Novell server accessible on the University of Hamburg LAN. The British Library also have *allegro*, but their project belongs to group 9.

## 2. University libraries

The majority of German university libraries (and others in higher education) are using *allegro* for different tasks and on different levels. Some have OPACs, some use the subject authority database of the Deutsche Bibliothek, some have begun to use the acquisitions and/or circulation modules. Many of them have campus licences and coordinate the cataloging activities of university institutes to collect their data in a university union catalog. Most library schools in Germany have included some exposure to *allegro* in their curricula. They find it difficult, though, to allocate enough time to make students really familiar with it.

## 3. University institute libraries

These are often under the supervision of the university's main library (see 2.). Their cataloging often comprises periodical articles as well as books. Many have OPACs, but very few are about to use circulation. Some universities have strict regulations that require the institutes to use no other library software and to conform to rules set by the university library.

## 4. Public libraries

Sizable communities exist in Berlin and in Lower Saxony. The latter are being serviced by staff at an office in Lüneburg who have made it their business to set the standards and work out the parametrization. The state's Ministry for Science and Culture has allocated funds for this specific purpose. One aim is to have access to and be compatible with the Lower Saxony Pica network based in Göttingen. Many public libraries import CD-ROM and diskette data from the Deutsche Bibliothek. Most want to use the circulation module, some have begun to test it.

A very special subgroup is that of the Goethe Institute libraries, serving the German cultural institutions abroad. Their headquarters in Munich supplies them with data on diskettes.

**5. School libraries**

Across the country, there are about 20 secondary schools using *allegro* for cataloging and OPAC: in all these cases, it was a teacher who found an interest in the matter and carried it through. In Lower Saxony again, a small group of teachers, supported by their ministry, are trying to work out a parametrization which may then be used more or less as a turnkey solution in other schools.

**6. Parliamentary, Court, and Archives libraries**

The main interest in this group is cataloging and documentation. Some have OPACs, some use a subject authority database. A few archive libraries have begun to produce databases and indexes of archival materials.

**7. Special libraries**

In research institutions and government branches, quite a few libraries have already built sizable catalog databases and OPACs as well as special documentation files. Some make their data available on the LAN of their institution, some give *allegro* to their researchers who use it for personal databases, which they also enrich with imported data from downloads. Media centers, music libraries, and manuscript departments are notable subgroups.

**8. Church libraries**

A cross-denominational group of church libraries are very active *allegro* users, exchanging lots of data and parameters. They are in episcopal offices, monasteries, archives, and educational units of the churches. The monasteries even run a union catalog of old books.

**9. Sinological libraries**

This is a European group (EASL = European Association of Sinological Librarians): at the forefront are Oxford (Bodleian), Berlin (State Library), and Heidelberg (Sinological Institute of the University). Several others are following their lead, most notably the British Library. PC front end software for Chinese characters is being used, and it has been found that *allegro* has no trouble working with these codes, even producing usable indexes for Chinese words although these do not have an alphabetical arrangement.

A similar front end was produced very recently for Japanese which is already being used in Oxford. They can download data from NACSIS, a Tokyo-based institution, whose database contains the de facto Japanese national bibliography. It seems very likely that libraries in Britain with collections of Japanese literature will all use this source, and that some will use *allegro* for their local databases.

**10. Projects and private use**

In many kinds of institutions, there are projects being run by individuals who discovered *allegro* as a database tool for their specific purposes. In universities, even students have begun to use *allegro* for private databases on their PCs.

**Usage outside Germany**

Most foreign users are, understandably, in Austria and Switzerland. The Goethe Institute libraries, scattered around the globe, are still German libraries. Group 9, as mentioned, has members in a few more countries. For another example, some Belgian music libraries have embarked on an *allegro* project. Recently, the Ukrainian Academy of Sciences in Kiev has been actively supported to get started with *allegro*, which has involved the implementation of Cyrillic.

After 13 years of development work, *allegro* now covers the whole range from very small single-user databases all the way up to state libraries and large multi-user applications.
The UNIX version, *allegro-X*, is being used at present in just four places. Its potential for Internet presentation of databases will most probably make it highly attractive as soon as version 13 becomes available.