NMRShiftDB—Constructing a Free Chemical Information System with Open-Source Components

Christoph Steinbeck,* Stefan Krause, and Stefan Kuhn

Max-Planck-Institute of Chemical Ecology, Jena, Germany

Received July 3, 2003

The process of designing and implementing NMRShiftDB, an open-source, open-content database for chemical structures and their NMR data based solely on free software is described. NMRShiftDB is available to the community on http://www.nmrshiftdb.org. It allows for open submission and retrieval of data sets by its user community. The software and the content itself is freely distributable, allowing for the establishment of a highly available mirror system of databases in collaborating laboratories.

1. INTRODUCTION

The open-source movement has shown that a software's quality can be improved by collaboratively developing it in an open manner.[1] Designing and maintaining the content of complex scientific database shares many characteristics with software development. Allowing for as many motivated individuals as possible to concurrently contribute to the database will certainly speed up data accumulation, providing open access to the data will create a sense of ownership among users and will create the desire to contribute to bug fixing and quality checking. It thus seems a reasonable idea to both exploit the open source principles for writing the database software and to assemble the content by motivating the user community to contribute to a database. Clearly, this will only succeed if the availability of the contributed data can be secured at any time, i.e., if the content is available freely under an open content license. Even further, the database service itself should be easily replicatable, which is ensured by both the open source character[2] of the underlying software as well as by the open content license[3] under which the content is maintained. While this has become common place in the young area of bioinformatics, many chemical databases are still proprietary and so NMRShiftDB is a novelty in the relatively mature world of chemical information systems.

An open content policy has further advantage for the academic scientist: When we decided to publish our own research in the area of computer-assisted structure elucidation[4] as free software, we also made an implicit decision on the type of data we could use for our research. Collaborating with commercial database providers in order to distribute, for example, neural networks for shift prediction based on their content with our software clearly was no longer an option, putting us in the need to search or create a free collection of NMR data.

The open source ideas inspired us to create NMRShiftDB, an open source, open content database for chemical structures and their NMR data, as an alternative for the chemical community. This article will explain our project from a technical point of view and assess the benefits of the open content system, of NMRShiftDB itself being an open-source project and of using open-source software as the basis for a chemical information system. Aspects of the database's functionality may be subject of another publication. NMRShiftDB lives on http://www.nmrshiftdb.org, with the full source code of the system being available on http://www.sourceforge.net/projects/nmrshiftdb.

2. SCOPE

NMRShiftDB is a research database for the end-user. This includes organic chemists trying to characterize their reaction products or byproducts by NMR and natural product chemists who want to quickly identify a freshly isolated compound, before reaching for more sophisticated structure elucidation methods. To perform this task, also known as dereplication, a scientist would record an NMR spectrum of the unknown, which then serves as a kind of fingerprint in a database search. Besides proton NMR spectroscopy, carbon-13 NMR spectroscopy especially has proven useful for this task.[5,6] The NMR spectrum is converted into a list of signal positions, which is entered into a text field on NMRShiftDB. Performing either an identity or a subspectrum search, the scientist would eventually be presented with a list of spectra being either as similar as possible to the query spectrum or containing the query spectrum as a subspectrum. As a structure-property database, NMRShiftDB's data sets are compound-centric, with at least one spectrum assigned to each structure. Thus, the spectral hits found in the above query will lead the scientist to chemical structures identical with or similar to the unkown. In addition to this dereplication task, the user can also perform the inverse search—entering a chemical structure and performing a (sub-) structure search—which will lead him to identical or superstructures and their assigned spectra. Searching for a number of other data set properties, like measurement conditions, chemical names, CAS number, etc., is of course possible. Based on the data contained in the database, NMRShiftDB is capable of providing shift predictions based on the HOSE code method.[7]

* Corresponding author phone: +49 (0)221 470-7426; fax: +49 (0)-221 470-7786; e-mail: c.steinbeck@uni-koeln.de. Present address: Cologne University Bioinformatics Center (CUBIC), Zülpicher Str. 47, D-50674 Köln, Germany.

10.1021/ci0341363 CCC: $25.00 © 2003 American Chemical Society
Published on Web 10/23/2003

A noteworthy feature is the open but peer reviewed submission to NMRShiftDB, enabling the database to grow from submissions by its users, as has been and is common place in bioinformatics or crystallography. Besides the free availability and its ability to grow by contributions from the community, NMRShiftDB shares its functionality with a number of older, nonfree, commercial systems.[8−10]

3. TECHNICAL REALIZATION

To make access to NMRShiftDB easy, we decided to offer a Web interface as the primary access method. Throughout the following discussion, the reader may keep in mind that the requirement for any of the employed components to be open source or at least freely distributable was an indispensable boundary condition.

The chemoinformatics part of the database was based on the recently published Java library for structural chemo- and bioinformatics, the Chemistry Development Kit (CDK).[11] With Java also being the language of choice for efficient and structured Web programming, the decision was made to fully base the NMRShiftDB information system on this programming language. Apart from advantages such as its coherent architecture, Java seems especially suitable for Web applications, with many good application servers available for running Java application.

Web applications share a lot of common characteristics. They typically need user authentication and administration, some sort of layout engine, and of course efficient database access. These common requirements have seeded the creation of a considerable number of toolkits, so-called Web application frameworks, providing the user with prewritten modules that support the above needs.

Our decision was made for two Web application modules written by the Apache Software Foundation, well-known from their open source Web server, which is now the defacto standard for this type of application.

Our first choice, Jetspeed,[12] is a framework for portals. Typically, portal applications share a common layout, with various parts of the display area being dedicated for displaying particular types of content. In Jetspeed, these areas are handled by Java classes called Portlets. Jetspeed handles the task of arranging these portlets, and the developer can concentrate on just writing the specialized application logic.[13] In a real portal these would contain items such as news, access to legacy applications, a content management system, etc. In our case we misused Jetspeed as an application framework and realized tasks such as search, data input, data display, or review as distinct portlets. Jetspeed offers a highly flexible user interface, where the user can decide for himself which portlet to view and how to arrange them, making customized portals possible. Although these facilities are not vital for our application they still can be used. A user, for example, can put search portlets she/he frequently uses on top of the page instead on bottom.

Jetspeed is based on another Apache project, Turbine,[14] serving the lower level parts of a Web application framework, enabling the application developer to hook his own presentation and business logic elements into an existing framework. Turbine has one entrance servlet handling all requests. The developer adds so-called action classes which handle user input and adds presentation logic elements either as Java

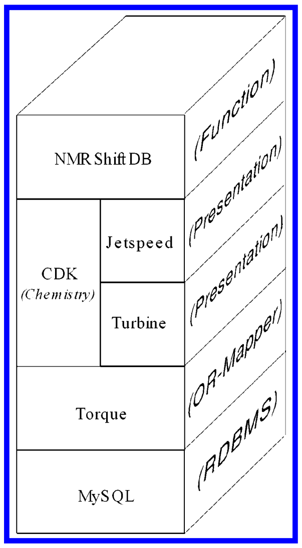

Figure 1. NMRShiftDB's component architecture.

Server Pages, a Sun technology, or as Velocity templates, an Apache technology, both integrating HTML and Java elements. In the HTML code the developer specifies which action to call if the user submits a form or clicks a link. The action class in turn contains the information which presentation element is to be used for answering the request. Turbine further features a user administration and an object-relational mapper called Torque, which frees the programmer from writing most of the tedious mapping logic projecting the properties of OO classes to RDBMS tables.

Altogether, the use of a Web application framework has been a very good choice. Certainly, building a Web application with Web frameworks will soon be as common as using a database management system for data storage.

NMRShiftDB's content is stored in a standard relational database. Our choice was for MySQL, a well-known database system, which has proven its speed and reliability in many heavy-load applications.[15] The mapping between the data stored in relational tables and the Java objects used in our application logic is performed by Torque.

For the chemoinformatics part, we choose The Chemistry Development Kit (CDK),[11] offering representations of basic chemical concepts such as atoms, bonds, molecules, 2D and 3D structure handling, file input/output for a wide range of formats such as MDL, SMILES, PDB, etc., and a lot of utility functions such as fingerprinting, ring perception, or aromaticity detection.

4. DATABASE DESIGN

The underlying database is the heart of the NMRShiftDB system. The entity-relationship diagram in Figure 2 shows molecule and spectrum as the central elements. In principle, the database is compound-centric, but no molecule may exist without at least one spectrum. There can be multiple spectra for one compound, differing either in their spectrum type or, for example, in their measurement condition. While spectra are decomposed into signals, molecules split up into atoms and bond. Spectra and structures are linked by an assignment of each signal in a spectrum to one or more atoms in the molecule. This design enables us to code assignments of molecular structures to even higher dimensional spectra by assigning groups of atoms or bonds to a particular signal

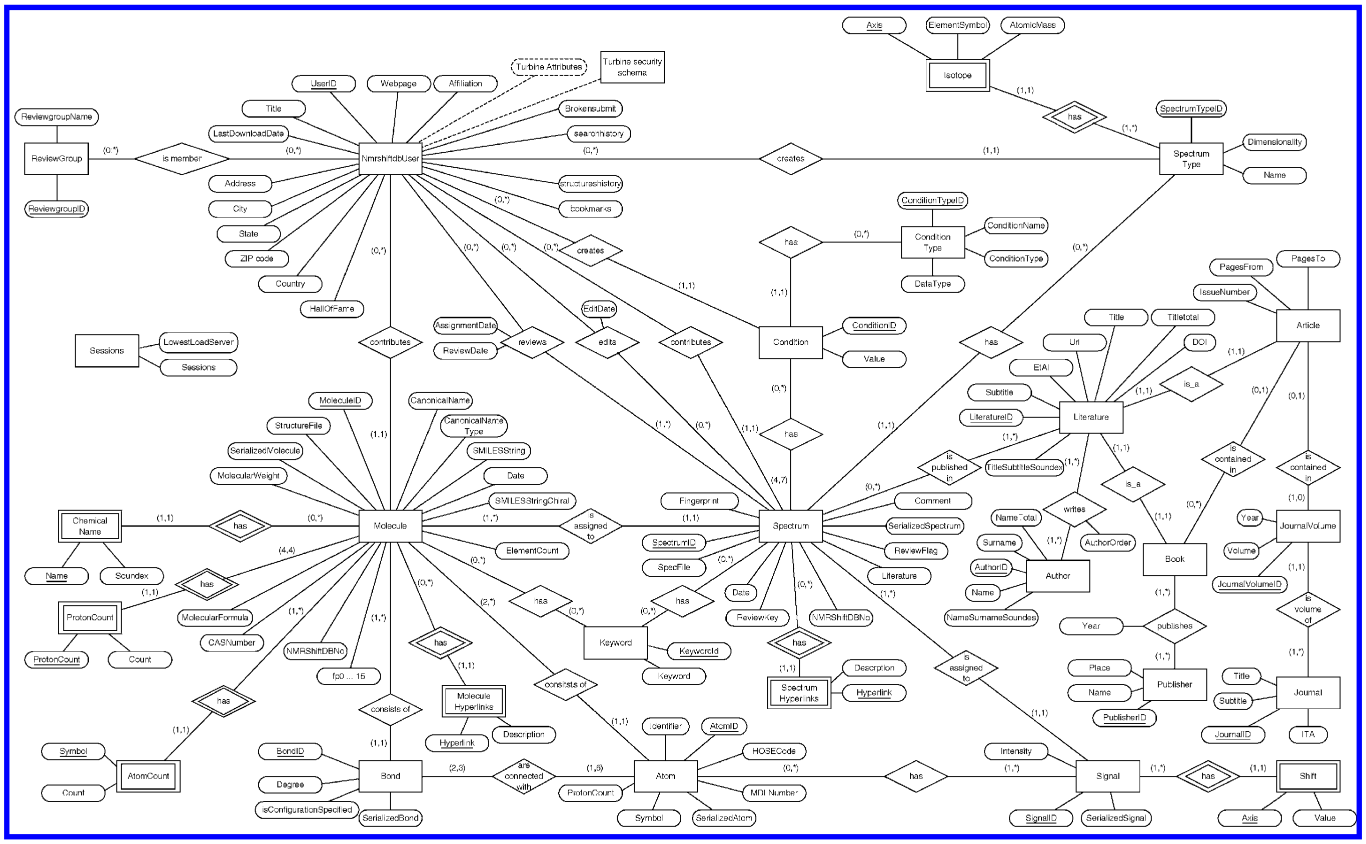

Figure 2. NMRShiftDB's entity relationship (ER) diagram.

in an *n*-dimensional spectrum. Beside these spectral information, additional data like chemical names for molecules and measurement conditions for each spectrum are recorded. Further, the database has a user administration taking care of rights management for submission and reviewing and also keeping track of who contributes, reviews, and edits spectra (via the *contributes*, *reviews*, and *edits* relationships). Finally there is a bibliography system for managing references. The full set of entities and their relations is shown in Figure 2.

This model does not strictly adhere to standard database theory. First, the model is not fully normalized, i.e., there are redundancies. Quite a few values could be calculated at runtime like the molecular weight or the spectrum fingerprint (see below). Further, the *AtomCount* and *ProtonCount* tables contain information that is also available in the Atom table. We decided to do this in order to speed up queries. Obviously, calculating the molecular weight of all molecules at every query would take more time. The *AtomCount* and *ProtonCount* tables avoid complex queries.

Second, we do not use weak entities as much as possible. The primary key of *Atom*, for example, could be composed of *MoleculeId* and *MdlNumber*. Taking the concept further, one could imagine almost everything to be a weak entity (except *NmrshiftdbUser*, *SpectrumType*, and *Condition*, which can exist without having a spectrum assigned). Obviously this creates rather complex primary keys, so we decided to restrict weak entities to reasonable cases.

It should be noted that the model is already prepared for handling multidimensional spectra, although these are currently not yet implemented. In case of an *x* dimensional spectrum a *Signal* would have *x* shifts with the *Axis* going from 1 to *x*. An atom would be assigned any number of signals (the cardinality of the *Atom-Signal* relationship is 1,* although currently only 1 is used). This design does not give any limits on the possible number of dimensions.

The first step for good database performance is the appropriate usage of indices. Since queries will be, especially in the long run, more frequent than inserts in NMRShiftDB, there was no reason for us not to use them. However, there are queries where the standard tuning measure of using indices does not help. This is the case especially with queries where calculations need to be done on values and not just values selected. Two of these cases are discussed in the following.

5. USER-DEFINED DATABASE FUNCTIONS SPECIFIC TO CHEMISTRY

The spectrum similarity search is probably the central feature in NMRShiftDB since it enables the user to find close matches to a spectrum. The algorithm employed for calculating the similarity is a two-step process which works as follows: At write time, the chemical shift axis of each spectrum is divided into 64 overlapping areas of equal size, corresponding to a 64-bit string, which are subsequently inspected for the existence of signals. If at least one signal is present, the matching bit in the 64-bit string is set to "1". This simple procedure yields a spectrum fingerprint that is stored with each spectrum. This type of fingerprint can be easily compared with the fingerprint of a search spectrum by a boolean "and" operation. The number of resulting 1-bits may, for example, serve as a similarity measure.

One can easily see that an indexing is not possible in this case so that a full table scan is unavoidable. Likewise, indexing is not possible for the function actually calculating the similarity. The solution we employed in order to speed up this time-consuming process was to use user defined functions (UDF), written in C++. A UDF is a shared library file (a dll on a windows system) which exposes a certain interface and is registered with the database. It can then be called in the SQL code like any built-in function. In our case a call could look like

```
SELECT spectrumsimilarity(SPECTRUM.SPECFILE, SEARCHSPECTRUM)
as similarity WHERE
fingerprint_compare64(SPECTRUM.FINGERPRINT, SEARCHFINGERPRINT) = "Y"
order by similarity desc;
```

The results are values between 0 and 100 for spectra in the database matching the spectrum fingerprint. The UDF *fingerprint_compare64()* is used for filtering spectra matching the fingerprint, and *spectrumsimilarity()* is used for calculating the similarity. Both SPECTRUM.SPECFILE and SPECTRUM.FINGERPRINT are fields from the database, SEARCHSPECTRUM the spectrum given by the user and SEARCHFINGERPRINT its fingerprint, both in a specific format). [The source code of the functions is part of the NMRShiftDB distribution and can be found in src/c++/spectrumsimilarity.cc and src/c++/udf_nmrshiftdb.cc.] The order by-clause even orders them with highest similarity first. NMRShiftDB offers two types of searches, one where the whole spectrum is used for calculating similarity and one where only the shifts found via the fingerprint are used, the rest neglected. Running such a subspectrum query against a database containing 3760 real-world spectra took 4.1 s on a Pentium III machine with 1.4 GHz, running MySQL as well as the application server. This time includes a fast bit-string matching operation and a much more time-consuming similarity calculation on the screened subset.

A standard problem in chemoinformatics is substructure searching in databases. Again, the indexing requires a solution specific to the chemistry context, thus calling for another user-defined function. A substructure algorithm needs to exhaustively map atoms in the query structure to those in the target structures until a full mapping is found. For chemical subgraph searches this time-consuming step at best runs in polynomial time.[16] To speed up the process we apply a two-step procedure of a fingerprint based prescreening followed by a full subgraph matching in the screening result. Fingerprints are one-dimensional bit arrays, where bits are set according to the occurrence of a particular structural feature. The CDK fingerprint implementation is based on Daylight, Inc., fingerprints, as described in their theory manual.[17] This elegant fingerprint algorithm works without a basic fragment index, thus being of greater generality.

We calculate the fingerprint of the search structure and filter all structures in the database where a bit is set for at least every bit set in the search fingerprint. This guarantees that all structures which contain the substructure are in the subset, but there may also be some surplus ones. When searching for pyrole, for example, this step leaves 121 structures of 3413 real-world structures, of which 41 are found to be exact substructures in the second step.

To do the prefiltering, a comparison of the search fingerprint with all fingerprints in the database needs to be done. Again, a UDF was written to implement this step. This two-step procedure gives reasonable search times: A substructure search for pyrole in a set of 3413 real-world structures took 2.3 s on the Pentium III machine. The prefiltering via the fingerprints took only 10 ms; the rest is used for the subgraph mapping. This means that search time is likely not to develop linearly when the database is growing, because not actual database size matters, but the number of structures surviving the fingerprint screening. The fingerprints of all structures are of course computed at entry time and saved in the database. The fingerprint search also offers a similarity search by only performing the fingerprint-based search and omitting the exact mapping.

The data necessary for the searches, the spectrum fingerprint as well as the spectrum in string format for the spectra, and finally the fingerprints for the molecules are kept in memory (HEAP tables in MySQL). [HEAP tables are not shown in the ER-diagram.] This gives a performance boost as compared to disk searches. These data are small enough to keep them in RAM, also with a large database, given 1 GB of RAM.

6. SHIFT PREDICTIONS

Shift predictions are one of the central tasks in a spectral database. Their quality depends on the size and the quality of the database as well as on the type of algorithm involved. We have decided to base our predictions on HOSE code tables,[7,19] a well accepted method used for shift prediction in many commercial applications and databases.

To perform a prediction for a user-defined structure, six-sphere HOSE codes of the atoms in the molecule are calculated and searched in the database. Since each HOSE code is linked to an atom and this again is linked to one or more chemical shift values, one may compute the average of all chemical shift values corresponding to a particular HOSE code, which is then considered to be an estimation for the shift. If 10 or more values are found the smallest and the largest values are taken as boundaries of a confidence limit. [The number of 10 values needed for the confidence limit has been found by experience and does not have a formal rationale.] If there are not enough values either for the estimation or for the confidence limit, a fall-back mechanism is employed which reduces the number of employed spheres by one. Technically this is achieved by storing the six-sphere HOSE code of every (carbon-)atom in the *Atom* table (see E-R-diagram in Figure 2). A query joining *Atom* on *Signal* and *Shift* can now return the desired values. In case of a less-than-six sphere query the higher sphere parts of the search HOSE code need to be cut off, and the query is done with the LIKE expression for HOSE codes starting with this code. To return only values of valid and reviewed spectra we also need a join with *Spectrum* and its *ReviewFlag* attribute. Furthermore, a join to *SpectrumType* is performed, so that only values from nuclei of the requested type are used. Because this procedure needs to be done for every atom and because of the fall-back mechanism, the number of queries needed for a prediction can be as high as 90 queries for 30 atoms and three fall-backs on average. Furthermore, the queries involve, as shown above, quite a few joins, so we decided to keep the HOSE codes, the values, and the *ConditionTypes* in an in-memory-table (table type HEAP in MySQL). This table is rebuilt hourly from the valid spectra and can be queried very fast.

7. QUALITY CONTROL

An important aspect of each scientific database is to maintain uniformly good quality of data. Too much flawed data can obviously render the database useless, so that a quality assurance mechanism is vital. NMRShiftDB employs both manual and automatic ways to performs such a quality control: First, data sets entered by users are subjected to scientific peer review by human reviewers. Currently four human reviewers are registered with NMRShiftDB, who have been recruited from the editorial staff and regular database users. This number will be increased as soon as the amount of submitted data sets requires it. These reviewers are supported by the system with precomputed quality control data, consisting in the HOSE code based shift prediction described above and a color coded table showing outliers and deviations. This mechanism helps the submitter and the reviewer to determine the validity of entered spectra. No data set is rejected automatically, neither due to a certain shift deveation from a predicted value, nor due to any other apparent error. At the current level of NMRShiftDB database size and design, human expertise cannot be replaced by an automatism. However, the human expert judging the new data set is and will be supported by a more and more elaborate set of quality judgment data provided by the database.

Second, outlyers are detected during shift prediction runs by a Nalimov test on the chemical shift set belonging to a particular HOSE code. These outlyers are reported to the database editors.

In the long run we intend to run regular checks where all spectra in the database are checked against the current data. NMRShiftDB's database design makes it easy to run SQL queries where the shifts of the molecule to be tested are taken out of the shift prediction mechanism, making it possible to cross-validate each spectrum against the rest of the database. This will also enable us to check molecules and spectra, which, at write time, could not be thoroughly checked because the database was too small at this time.

8. THE NMRSHIFTDB MIRROR SYSTEM

To make NMRShiftDB a reliable system and also to increase the cooperative element we decided to have NMR-ShiftDB as a distributed system of independent but linked servers. All these servers have their own database, Tomcat and Apache servers running. The databases are replicated between them via MySQL's replication function. They all count how many users are working on them and make these data available via a Web service. The servers ask for these figures at regular frequencies and save this information. If a user goes to the index.jsp page of any server, this server redirects him to the server with the lowest load. It is also possible to give load factors to systems, so that more powerful servers get more users. Obviously the load balancing assigns loads only roughly equally, but it avoids that some servers are not used at all and others are overcrowded. Reasons for differences in load (apart from the load factor) could be the following: (a) Not active users are counted, but sessions, which expire a certain time after the last action. This leads to inactive users still counting for a while, despite them having already left the system. (b) Users actually cause different loads, but are all considered equal. (c) Because the

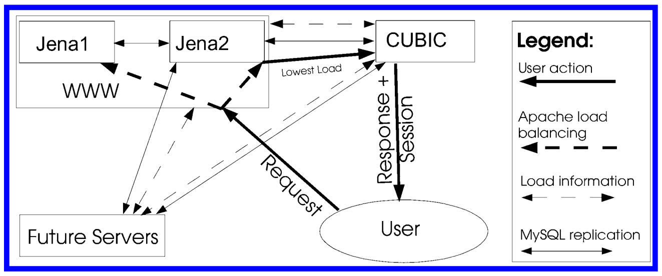

Figure 3. NMRShiftDB's mirroring system.

systems only ask for load figures from other systems once in a while, they might redirect people to a system which in the meantime got more users. This whole process is depicted in Figure 3. Instead of using mirrors such as http://cubic.nmrshiftdb.org directly, most people will probably access the central gateway on http://www.nmrshiftdb.org. Therefore we wanted to keep this server especially reliable and made it a fail-safe cluster in itself, which is visible to the outside just as one system and is also part of the NMRShiftDB cluster as one system. The two servers in this cluster both run MySQL and Tomcat, but only one runs Apache. The Tomcat Apache module does load balancing between the two Tomcat processes. If one server fails, the other one gets all requests. If the server running Apache fails, the other takes over its responsibilities. This is achieved via a software called *Heartbeat*,[20] which is running on both servers and monitors the availability of the other server. In case this is not available and the server is currently not running Apache, it starts Apache and takes over the IP number of http://www.nmrshiftdb.org. This system is transparent for users.

We combined these two clustering systems to enable each of the servers within the Jena subcluster to take over the IP number of http://www.nmrshiftdb.org, as required by the heartbeat system, and to be reachable with the IP known to the international Domain Name System (DNS). This is not possible if servers are in different locations and subnets. So for example a server in the network of the University of Cologne cannot take over an IP in the IP range of the Max-Planck-Institute of Chemical Ecology.

9. RESULTS

To build the current pre-1.0 release of NMRShiftDB mainly one developer was employed, working on it full-time for 16 month. There was also a student involved who mainly did database design in the context of his study thesis. A third developer contributed code. Altogether, the cost can be estimated at 18 man-month, not including time spent on entering data, which was also part of the testing procedure.

With this effort we reached a system which seems fit for use and already experienced good feedback from the scientific community. In August 2003 usage was like follows: 101 people registered for NMRShiftDB, although registration is only necessary for submitting data. The database currently contains approximately 5200 structures with a slightly higher number of spectra, which are mostly carbon-13 and a few proton NMR spectra. Shift predictions are thus available only for carbon-13 spectra, but the capability to predict proton NMR spectra will be added as soon as there is enough data available for this nucleus. Typically, there are 20−30 open sessions on the servers, i.e., 20−30 people are using or have recently been using NMRShiftDB. Typically something like 50 searches were done during a day. Three servers—jena1.nmrshiftdb.org, jena2.nmrshiftdb.org, and cubic.nmrshiftdb.org—are currently online to form a highly available set of mirrors. All servers are transparently accessed via load-balancing through a single URL http://www.nmrshiftdb.org.

The quality assurance process has already shown to be working. When entering data, the estimations typically are close to the actual data. When we started building the current database this was not always the case. We were able to track this down to software bugs in our own code but also in the libraries used. These improvements led to higher quality predictions, which are also able to identify errors in the literature.

10. EXPERIENCES WITH USING OPEN-SOURCE SOFTWARE

As mentioned before, all of the components used for building NMRShiftDB are open-source software with the exception of the Marvin-Applets used for displaying structures, which are free for certain purposes but not open source.[21] The general benefits of using open source systems not only in science have been discussed in great length, and we recommend Eric Raymond's articles on this subject as a starting point for reading.[1]

In the following we focus on some of our own experiences supporting these findings. Apart from the feeling that open source software might be appropriate for an open content system, there are also advantages in the development process, most prominently the possibility to extent and to change code. During the development of NMRShiftDB, using open source components proved beneficial in many respects. For structure identity checking, for example, we employed canonical SMILES generated by the respective class in the Chemistry Development Kit (CDK) to make sure that compounds are recorded only once in the database even if they are entered multiple times. It turned out that the CDK implementation did not respect stereochemical configurations, i.e., molecules which had different configurations were considered to be the same. Thanks to the CDK being an open source library we could enhance the *SmilesGenerator* to produce stereochemically enhanced SMILES. With a closed source product we would either have been forced to develop the SmilesGenerator from scratch, which would have taken longer and would not have had the proven stability of an existing solution, or to wait for the producer of the library to come up with a solution. Furthermore, the new enhanced *SmilesGenerator* has been contributed back to the CDK project and can now be used by other parties, which guarantees a more comprehensive testing. Indeed, there has already been feedback which helped improving the class.

Another example are the daemons provided by the Jetspeed system. These are classes the methods of which are called at defined intervals by the Jetspeed system. They can be used for tasks such as cleaning the database. In our case some tasks need to know the path to the directory where the Jetspeed system is installed. This information can usually be read via the *ServletConfig* class. Unfortunately this class is usually not available in the method of the daemon classes. Again due to the open nature of Jetspeed it did take only a short time to find where the daemon classes are instantiated and to make sure that *ServletConfig* is given to the Constructor as a parameter and set as an attribute, which can be used in the method actually executed.

These cases show that in an open-source system one is not confined to the original scope of the software but can use it for building new solutions incorporating existing pieces and inheriting the proven stability of these components.

Another advantage of open-source software is the possibility for the educated user to correct bugs due to direct access to the source code. This was for example the case in a class of the turbine framework needed for database access. There was a small bug having large effects, but it was a matter of literally minutes to find it and to change it.

Lack of support is often said to be a disadvantage of open-source products. This is only partly true, because first commercial support can be bought for many open-source products (e.g. for MySQL) and second there are typically newsgroups or mailing lists available. For the products we used we found these mechanisms to be working and helpful. Our experience is that if a question was specific enough to be answered, the answer was provided very quickly.

Altogether we found open-source software a valuable base for software development in a scientific environment. Not only does it help to build a trusted system, it also is a good platform for efficient and fast software development.

11. CONCLUSION

We have documented our efforts to build the chemical information system NMRShiftDB, a Web database for chemical structures and their NMR data, solely based on free software. NMRShiftDB is an experiment as to which extent its scientific user community is able and willing to exploit open-source, open-content ideas for building a free NMR database. Even the feedback prior to any publication with a larger audience has been very encouraging so far.

Our plans for the future include the possibility to enter multidimensional spectra and to provide the user with a stand-alone client for a more convenient data input, retrieval, and management. This client will be also be realized in Java, therefore it will be platform-independent, and it will offer the possibility to build local databases, which can then be synchronized with the NMRShiftDB servers. This could be useful for people working on spectra who want to submit them once all data have been collected and checked for errors offline. Probably the single most important feature to be added is the support for scalar coupling constants as soon as more proton spectra are added to the database.

The NMRShiftDB service is hosted on http://www.nmrshiftdb.org. We highly appreciate comments, suggestions, bug reports, and contributions of data sets and source code. The full source code of NMRShiftDB is available on http://www.sourceforge.net/projects/nmrshiftdb.

ACKNOWLEDGMENT

The authors would like to thank all members of the CDK project[11] for their contributions, corrections, and helpful comments. Further, our gratitude goes to those who contributed data sets, both via manual input and by providing data sets from in-house collections. We are also grateful to the German Research Council (Deutsche Forschungsgemeinschaft, DFG) for financial support of the NMRShiftDB project.

REFERENCES AND NOTES

(1) Raymond, E. S. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary;* O'Reilly and Associates: Sebastopol, CA, 1999.
(2) The Open Source Initiative (OSI), http://www.opensource.org (accessed on June 2003), 2003.
(3) The OpenContent License, http://www.opencontent.org (accessed on June 2003), 2003.
(4) Steinbeck, C. SENECA: A Platform-Independent, Distributed and Parallel System for Computer-Assisted Structure Elucidation in Organic Chemistry. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1500−1507.
(5) Steinbeck, C. Computer-Assisted Structure Elucidation. In *Handbook on Chemoinformatics;* Gasteiger, J., Ed.; Wiley-VCH: Weinheim, 2003; Vol. 2.
(6) Steinbeck, C. Correlations between Chemical Structures and NMR Data. In *Handbook on Chemoinformatics;* Gasteiger, J., Ed.; Wiley-VCH: Weinheim, 2003; Vol. 2.
(7) Bremser, W. HOSE − A Novel Substructure Code. *Anal. Chim. Acta* **1978**, *103*, 355−365.
(8) ACDLabs. ACD/CNMR DB Add-on, 2002.
(9) Neudert, R.; Penk, M. Enhanced structure elucidation. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 244−248.
(10) Schutz, V.; Purtuc, V.; Felsinger, S.; Robien, W. Csearch-Stereo − A New Generation of NMR Database Systems Allowing Three-Dimensional Spectrum Prediction. *Fresenius J. Anal. Chem.* **1997**, *359*, 33−41.
(11) Steinbeck, C.; Han, Y. Q.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. The Chemistry Development Kit (CDK): An open-source Java library for chemo- and bioinformatics. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 493−500.
(12) The Web Portal Framework Jetspeed, http://jakarta.apache.org/jetspeed, accessed on 05/23/03, 2003.
(13) Kuhn, S. Jetspeed. In *Portale und Applikationen mit Apache-Frameworks*; Theis, F., Ed.; Software und Support Verlag: Frankfurt, 2003.
(14) The Web Application Framework Turbine, http://jakarta.apache.org/turbine, accessed on 05/23/03, 2003.
(15) The Relational Database System MySQL, http://www.mysql.com/products/mysql/index.html, accessed on 06/06/03, 2003.
(16) Faulon, J. Isomorphism, automorphism partitioning, and canonical labeling can be solved in polynomial-time for molecular graphs. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 432−444.
(17) James, C. A.; Weininger, D.; Delany, J. Daylight Theory Manual, http://www.daylight.com/dayhtml/doc/theory/theory.toc.html (accessed on June 2003), 2000.
(18) Krause, S. NMRShiftDB − A Web-Based, Open-Access, Open-Submission Database for NMR−Chemical Shifts, Study Thesis, University of Jena, Germany, 2002.
(19) Schutz, V.; Purtuc, V.; Felsinger, S.; Robien, W. Csearch-stereo − a new generation of NMR database systems allowing three-dimensional spectrum prediction. *Fresenius J. Anal. Chem.* **1997**, *359*, 33−41.
(20) Linux High Availability Home Page, http://linux-ha.org/ (accessed on Jun 2003), 2003.
(21) The Marvin Applets for Structure Drawing and Viewing, http://www.chemaxon.com/marvin/, accessed on 06/06/03, 2003.

CI0341363