Donnellan, B., Fitzgerald, B., Lake, B. and Sturdy, J. (2005), 'Implementing an Open-Source Knowledge-Base', IEEE Software, 22(6), 92-96.

Editor: Christof Ebert ■ Alcatel ■ christof.ebert@alcatel.com

Implementing an Open Source Knowledge Base

Brian Donnellan, *National University of Ireland, Galway*

Brian Fitzgerald, Brian Lake, and John Sturdy, *University of Limerick*

Managing information is a key challenge for all of us. And there's no one method for it because we're humans with different communication, storage, or retrieval preferences. As with our desks: some like the big pile of notes, mail, and books from which they search effectively for what they're after; others keep a clean desk, sorting everything into files and finding it later according to predefined criteria. These differences are why we're devoting two Open Source columns to this topic. The September/October 2005 column looked at the first type of information retrieval—namely, searching the pile. This time, a research group working from the University of Limerick describes an open source knowledge base for sorting and accessing information. —*Christof Ebert*

The concept of *knowledge bases* originated in artificial intelligence as one side of expert systems—namely, the fundamental body of knowledge available to a domain. Although the inference engine side of such systems remains primitive by comparison to the human brain, KBs have nevertheless continued to evolve with advances in information technology.

KBs are particularly appropriate in knowledge-intensive activities like software development. They offer context-based access to complex information, including informal documents and multimedia, as well as a centralized means of storing and preserving digital assets. KBs can help software engineers with many tasks—from project management and design rationale to version control, defect tracking, code reuse, and staff training and development.

Two broad KB strategies are available:

- *Codification* focuses on capturing electronic information in a classification hierarchy.
- *Personalization* goes further to accommodate the unique and idiosyncratic ways that humans process and use knowledge.

Information libraries rely primarily on a codification strategy. Knowledge bases, on the other hand, offer additional functionality to facilitate personalization.

We recently implemented an open source KB to support the Consortium for Studying Open Source in Public Administrations (www.cospa-project.org). COSPA originated in an EU initiative to study the use of open source software to reduce public administrative software and system support costs. The KB project aimed to build a multilingual knowledge base for comparing and pooling knowledge and experience.

Project requirements

Like all KBs, the COSPA project had to handle a diverse range of documents including articles, preprints, working papers, reports, conference papers, computer programs, and multimedia publications such as video and web-

 0740-7459/05/$20.00 © 2005 IEEE

Table 1

Candidate knowledge base solutions

	Tool		
Features	**DSpace**	**Data Centric KMS**	**Lotus Notes**
Proprietary/open source	Open source	Open source	Proprietary (IBM)
Development status	Version 1.2, 13 Aug. 2004	Version 1.2, 12 Oct. 2003	Release 6.5.3, Sept. 2004
Platform	Operating system independent, Java server application with normal Web browser client	Operating system independent, Java client application	Server runs on several platforms, including Windows, Linux, HP-UX, and Sun Solaris; client runs on Windows and Mac OS
Download location	http://sourceforge.net/projects/dspace	www.3rdmill.com	www.ibm.com
Requirements	PostgresQL, Unix server, Java development environment	Oracle, Unix server, Java development environment	Domino server, Java development environment
Linkage to email systems	SMTP-based email	Any system supporting the Java mail API	SMTP-based email
Metadata standards	Dublin Core Open Standard, Open Archives Initiative Protocol for Metadata Harvesting	HL7 Medical Data Standard	Proprietary Notes Storage Facility format
Cost	Available free as open source under BSD distribution license	Licensed under GNU General Public License. Available free if use is 100% GPL compliant; in all other instances, a commercial license is available.	Prices vary, but a 1,000-seat license could cost about €170K
Customization	Users can modify DSpace to meet specific organizational needs	Similar to DSpace but with GPL license restrictions	Built-in scripting facilities on client

casts. Wherever possible, we adopted relevant open standards for creating, describing, and accessing information and information assets. For example, we used the Dublin Core (http://dublincore.org), a standard initially developed in 1995 to facilitate the discovery and retrieval of online resources, and the Open Archives Initiative Protocol for Metadata Harvesting (www.openarchives.org), an application-independent interoperability framework based on metadata harvesting.

Furthermore, we wanted adoption barriers to be low, so we were keen to release all software produced in the joint project under an open source license. The resource constraints on project partners in academe and public administrations were tight, so we had to keep the initial system as small as possible while still meeting early adopters' needs. This small footprint also supported experimenting with ways to use the system. Another requirement was to enable users both to read from the system and to contribute documents to it without needing client software beyond their normal Web browser.

Because the COSPA KB included government documents, archival issues were also important. We needed to ensure that the KB would outlive our project.

KB product comparison

Myriad proprietary and open source KBs are available. An open source KB was obviously well suited to COSPA, given its open source mission and global development community. Furthermore, open source products offer community-based support for the life of the product. In any case, we couldn't mandate that participants use a single proprietary system across multinational government entities.

Table 1 compares two open source candidates as well as a popular proprietary product that could meet our requirements. Knowledge base developers might also want to explore developing systems from scratch directly on a *LAMP stack*—that is, the Linux operating system; Apache Web server; MySql database management system; and Perl, PHP, or Python scripting language. We chose DSpace (http://dspace.org) on the basis of desk research—that is, consulting colleagues and bulletin boards—which suggested it was a robust product with an active and responsive development community.

DSpace is an open source digital repository designed to capture, preserve, and redistribute intellectual output in digital formats.

DSpace is an open source digital repository designed to capture, preserve, and redistribute intellectual output in digital formats. A consortium led by MIT and Hewlett-Packard developed DSpace to support the long-term preservation of digital material stored in repositories. This met our requirements for longevity of access as

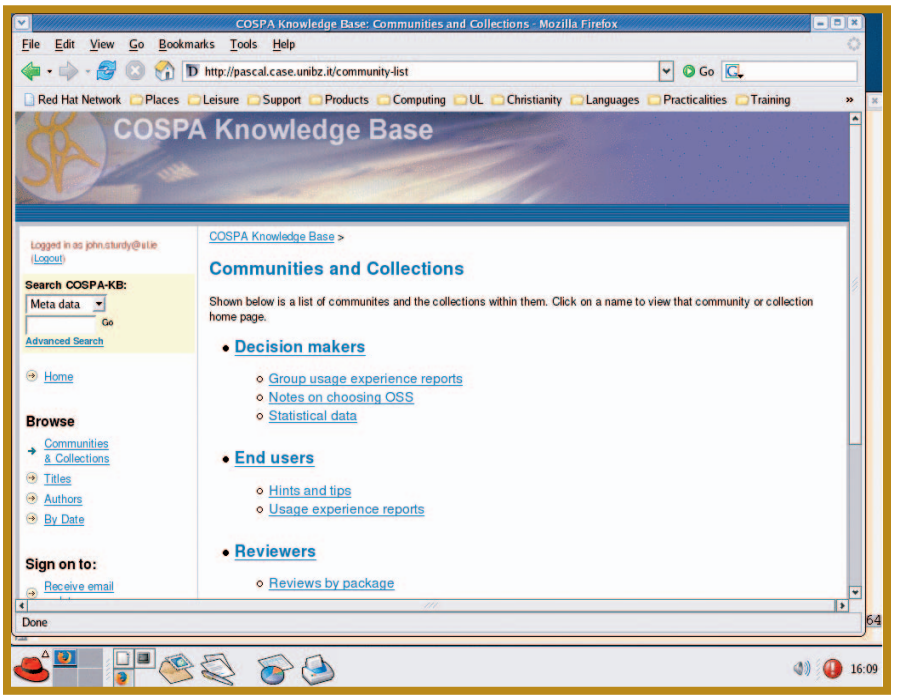


Figure 1. Homepage for the Consortium for Studying Open Source in Public Administrations (COSPA) knowledge base.

well as community-based support. The system is available free to any individual or company. Users can customize the system to suit their requirements. The DSpace Federation is a metacommunity that allows member user communities to adapt the system to meet particular needs and manage the submission process for items being added to the knowledge base.

Getting started with DSpace

The open source development model provided definite advantages. However, we found DSpace to be quite a heavyweight approach that took considerable work to change from an online library to a KB.

Support and security issues

We downloaded the DSpace files from SourceForge and began implementation according to the best instructions we could find on the DSpace site. System installation was quite complex—no "idiotproof" installation wizards were available. However, we readily found others who had been down this route and who advised us on the best server configuration for DSpace. In the end, we found someone who had run a DSpace installation workshop to help us for a small consulting fee. This informal service community shows how open source stimulates software development opportunities and levels the competitive playing field for small players. Our consultant quickly found that one of our Web server components was incompatible with the Linux version we were using.

This informal service community shows how open source levels the competitive playing field for small players.

The diverse community examining the DSpace code base ensures that defects or security concerns are quickly discovered and solved. Generous community support is one of the positive network externalities in using open source products.[1]

From a security perspective, the COSPA KB also benefits from using Web server components that are fundamental to many other projects. A large pool of labor exists that, though not directly involved with our project, is nevertheless intensely interested in supporting its software.

Initial configuration

Once we had the system running, we started experimenting with the main configuration file, which was well commented and worked as expected. Customizing the system presentation was comparatively simple, requiring little knowledge beyond that of a typical Web developer.

We implemented the interface in a manner consistent with the World Wide Web Consortium's design standards. We wrote much of it in JavaScript and the W3C's cascading stylesheets (CSS). Figure 1 shows the COSPA KB homepage.

Our initial customization of the DSpace presentation concentrated on replacing tables in the Web layout with a pure CSS layout. Using open source code let us tweak the code to precisely fit the contingencies of specific development contexts. For example, we wanted to ensure that even the smallest Web-accessible portable devices could display our KB in a meaningful and functional way.

It was straightforward to customize the interface parts that were defined purely in page template files, but some parts were defined in HTML output by Java tags called from the JavaServer Pages files. This was more difficult. In fact, we couldn't have done it on a closed-source system, but JSP technology provides a simple way to create dynamic Web content and enables rapid development of Web-based applications that are server- and platform-independent.

Further customizations

Following interface customizations, we began customizing system behavior. DSpace supports this task not only through text field parameters in the configuration files but also through locally modified page templates in a local JSP directory. Whenever DSpace looks for a JSP file, it first looks for the filename in the local JSP directory. If it doesn't find the file there, it looks in the main JSP directory. Thus, a modified installation of DSpace can override any of the pages in its user interface without disturbing the distributed DSpace files. Our changes included adding pages that reference our own JSP tags as defined in our added Java class libraries. This override capability also keeps the installation of an updated DSpace distribution from disturbing our modifications, as the original pages and the modified ones are in separate directories.

The search system that comes with DSpace is based on Apache Lucene, an open source search engine that DSpace uses to index only the metadata that users submit with their documents. We modified the submission process, in which users select the files to be fetched from their local machine and fill in the metadata. The modification fetches the file earlier in the process sequence. This let our modified DSpace code tokenize and analyze the contents to provide likely initial metadata values. Figure 2 shows the KB interface for editing document metadata.

Simultaneously with analyzing the document for likely keywords and so on, we also index its full text and provide a plain-text version for low-resolution user devices. Much of this work was straightforward once we understood the way the submission process moved from one page to another.

Commenting mechanism

To meet user community requests, we added functionality that lets users comment on documents and also on comments, in the style of blogging software such as geeklog or slashcode. This functionality adds value to a document after submission and helps the KB become a community meeting place rather than simply a static information repository.

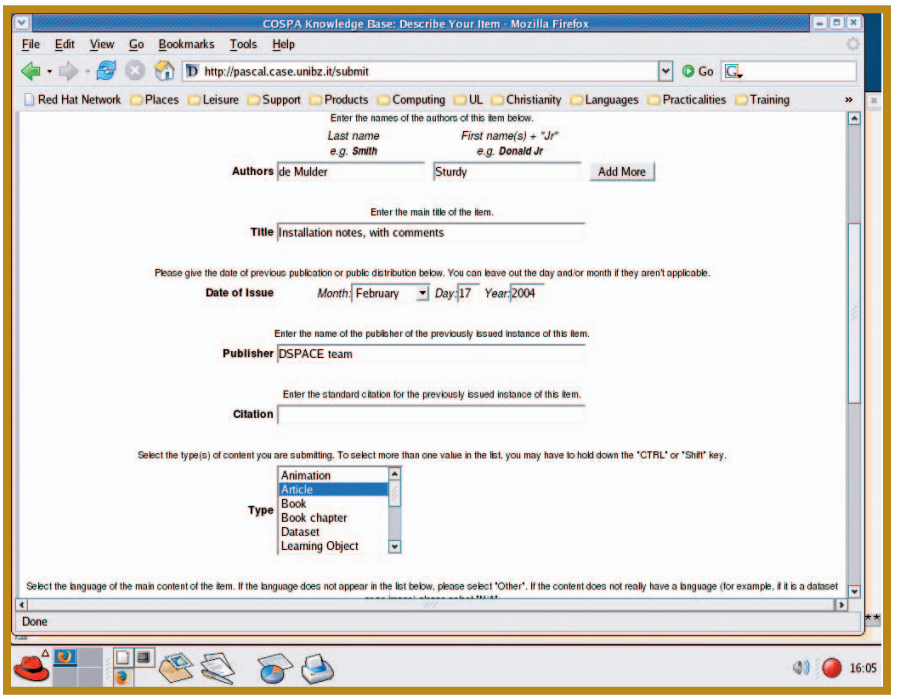


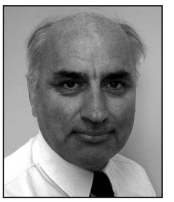
Figure 2. COSPA knowledge base document submission metadata editor.

The primary developers of our knowledge base were also its main users, with varying degrees of involvement depending on the task at hand. Ongoing user-centric evaluation was a vital component in determining how the KB should address specific issues and how effective the implemented solution was. We often added new functionality as a direct result of user desires to streamline a process or simplify a task. We couldn't have done this with a proprietary solution, given that many ideas for improvement required modifications to the system's core functionality—something a proprietary system wouldn't normally allow.

In our experience, a KB works well only when it's allowed to evolve as a tool. Dissatisfied users don't tend to be repeat users, and ensuring that good suggestions are acted upon has been important to this project's development. The open source development model allows the KB to evolve flexibly with the COSPA project and its members.

Reference

1. B. Fitzgerald and T. Kenny, "Developing an Information System Infrastructure with Open Source Software," *IEEE Software*, vol. 21, no. 1, 2004, pp. 50–55.



Brian Donnellan is a lecturer in information systems at the Department of Accountancy and Finance and the Centre for Innovation and Structural Change, National University of Ireland, Galway. Contact him at brian.donnellan@nuigalway.ie.

Brian Fitzgerald is the Frederick A. Krehbiel II Chair in Innovation in Global Business of the University of Limerick's Department of Computer Science and Information Systems. Contact him at bf@ul.ie.

Brian Lake is a PhD student in the University of Limerick's Department of Computer Science and Information Systems. Contact him at brian.lake@ul.ie.

John Sturdy is a postdoctoral research fellow in the University of Limerick's Department of Computer Science and Information Systems. Contact him at john.sturdy@ul.ie.