MMBase: An open-source content management system

J. Becking
S. Course
G. van Enk
H.T. Hangyi
J.J.M. Lahaye
D. Ockeloen
R. Peters
H. Rosbergen
R. van Wendel de Joode

MMBase is an open-source content management system (CMS) with portal functionality, originally created by the Dutch public broadcaster VPRO. MMBase, which is particularly suited for multimedia environments, is based on the concept of presenting objects on different channels. The system is highly platform-independent and has adopted standards such as Java™, XML (Extensible Markup Language), J2EE™ (Java™ 2 Enterprise Edition) and JDBC™. Initially used only by public broadcasters, MMBase has been adopted by a growing number of organizations. This paper presents the history of MMBase, introduces its community of users, and discusses its architecture, focusing on the innovative technical process underlying MMBase and its organizational structure. We identify and discuss three challenges facing the user community: the need to motivate and organize both users and developers to contribute to the development of MMBase, the need to make the software modular, and the need for more and better documentation of the MMBase core and its component packages.

MMBase is an open-source content management system (CMS), maintained and improved by the MMBase community and initially developed at VPRO, a Dutch public broadcasting organization. MMBase is a widely adopted open-source solution in the Netherlands. Its growing user base includes a number of well-known and highly respected organizations: Dutch government departments, the city of Amsterdam, and broadcasters like VPRO, EO (Evangelical Broadcasting Organization), Radio 538, Dutch Internet organizations for education, and many cultural organizations. Recently, there has been a growing interest in MMBase from markets outside of the Netherlands: its adoption by Vodafone is probably the best example.

CMSes typically separate the concerns of content, application logic, and visual make-up.[1] In addition to these characteristics, MMBase shares a number of characteristics with enterprise content management (ECM) systems. An ECM system is different from traditional CMSes in that it "integrate[s] information (content) from different sources, form[s] it into a collection (compound content), provide[s] it to users and applications, and add[s] value to the

represented information."[2] ECM systems provide the same functionality as traditional content management systems, but add two distinct features:

■ MMBase is continuously evolving ■

content infrastructure, which allows solution applications to be built without a large system integration effort, and *content integration*, which consolidates content from various sources to make it easily accessible to a network of users.[2]

The combination of a number of characteristics makes MMBase unique. MMBase is focused on structured content. This represents a difference when compared to a CMS like IBM Content Manager. The latter has the ability to "manage nontraditional, heterogeneous, unstructured data."[3] MMBase lacks such functionality and must be combined with other systems to store and retrieve other types of data and content. Therefore, MMBase should be characterized as a CMS and not as an ECM system.

MMBase is platform- and database-independent. Many of the bigger companies are highly heterogeneous: they have adopted a wide variety of databases and hardware platforms. MMBase supports this heterogeneity, as it can be used on most of the available platforms (UNIX**, Linux**, and Microsoft Windows**). Platform independence in MMBase is achieved through Java**,[4] and it can be used with many commonly used databases (e.g., Informix**, DB2*, Oracle**, PostgreSQL**, MySQL**). Heterogeneity is a significant advantage, as it allows companies to use MMBase while maintaining low operating costs by focusing on a single operating system and database.

MMBase strictly upholds the multitier architecture of Java's enterprise edition. Because data, application, structure, and presentation are conceptually separated, MMBase provides modular and replaceable implementations (i.e., printing, mobile, mail, flash). MMBase is object-oriented. Content in MMBase consists of objects, which are described in XML (Extensible Markup Language) and Java and are presented in an "object cloud." This has the advantage that content has to be stored only once. However, to many smaller organizations this has the disadvantage that every form of content must be described as a component and needs to be stored according to similar standards. This requires initial investment of time and effort upon storing the content. These costs can exceed the benefits, especially in smaller organizations where the complexities and heterogeneity of content are relatively low.

MMBase is licensed as an open-source software project. The license with which MMBase is protected is the Mozilla** Public License (MPL). This licensing scheme has several advantages. There is no licensing fee that needs to be paid for MMBase. This does not necessarily mean that MMBase is less expensive than other CMSes, as this depends on the total cost of ownership (TCO). However, it does make MMBase attractive, as the licensing costs are very clearly visible and quantifiable. In addition, compared to the GPL (General Public License), the MPL poses less risk to corporate users. One of the important characteristics of the MPL, as compared to the GPL, is that it is "non-viral"; that is, software integrated with the MPL does not need to be licensed under the MPL upon distribution.

MMBase is compliant with the J2EE** (Java 2 Enterprise Edition) platform. Currently, the market is dominated by two standards, namely .NET** and J2EE.[5] Although, it is not clear in which direction the market as a whole will move, it is clear that these are currently the two most likely standards to prevail in the future, and their dominance is most likely to increase.

MMBase supports the concept of multichanneling. Strict separation between content and layout and the object-oriented structure facilitate the presentation of content on various media channels. This characteristic is partly due to the origin of MMBase in the broadcasting industry. This industry has very stringent requirements regarding the management of content and its presentation.[6] MMBase does not itself store or offer streaming-media facilities. Streaming media are addressed by storing a reference to a resource on dedicated streaming servers. The MMBase media project offers a way to include information about the streams with these references, such as bandwidth and type, allowing for filtering of streams by user preference, and showing fragments of a stored stream (e.g., "the first 6 minutes").

MMBase has proved to be very scalable. Some of the organizations that have adopted MMBase manage millions of objects in various formats, such as video, audio, and plain-text documents. An example is VPRO, which currently has eight million objects stored.

MMBase is continuously evolving and gaining in functionality. Partly due to the fact that the software is open source, new functionality is constantly added. Everyone in the community is welcome and encouraged to contribute new add-ons to the existing system. Newly added functionality includes poll, chat, shop, and forum services. However, the voluntary community-like structure also has its drawbacks, such as "free-riding" (this issue will be discussed more extensively later in this paper).

MMBase poses a steep learning curve for first-time users. As mentioned previously, every item of content has to be stored as an individual object, which requires relatively high initial investment. In addition, MMBase consists of many configuration files whose relation can at times be unclear. First time users have to learn many syntaxes and functionalities, for instance "taglib" and edit wizards. A taglib, or tag library, is a set of actions that encapsulate functionality. These tags are then used within JSPs** (JavaServer Pages**). Finally, to write queries in MMBase is not straightforward and requires significant analysis. These are among the reasons why first-time users might experience difficulties when adopting MMBase and why they need to make relatively high initial investments.[7]

This paper addresses the growth of the MMBase software and community. First, it addresses the history of the software, relates the decisions and happenings at the heart of the software, and briefly introduces the structure of the MMBase community. Next, it presents the architecture of MMBase. The third section presents examples of organizations that adopted MMBase and describes how software vendors currently use the package to deliver value and performance to customers. This section is primarily based on interviews with stakeholders who were involved in the adoption and implementation of MMBase. The fourth and final section identifies and discusses three challenges facing the MMBase community and the software itself.

THE HISTORY OF MMBASE

The origin of MMBase can be traced back to VPRO. VPRO is a Dutch public broadcasting corporation with more than 320,000 member subscribers. The corporation is well-known in the Netherlands for its innovative programs and its focus on creativity.

In 1994, VPRO gathered a group of 20 people to move the corporation from the analog into the digital environment. The team created several products including television programs about digital issues, CD-ROMs, meetings, and lectures, and they identified and developed many kinds of online activities. Within weeks, VPRO was connected to the Internet, and a Web site was created. The team also set up a free public modem bank because at that time Internet access was still relatively difficult for many viewers. Via the modem bank, visitors could look at the broadcasters' Web site to check the VPRO program schedule or details of their favorite television shows.

VPRO's primary goal regarding digital media was to provide a learning environment for its content creators, approximately 200 in total. They were encouraged to explore the environment to find new ways of expressing themselves and stimulating the public. This resulted in numerous new and pioneering products, many of which were far removed from the then stereotypical activities of broadcasters. An example was the design of the VPRO Web site itself, which appeared in a special online gallery of the San Francisco Museum of Modern Art.

During this transition to digital content, the technical staff was challenged to provide and maintain an infrastructure and toolset that would allow the content creators to actualize their new concepts. In 1994, the market for Internet tools was still in its infancy, and few tools of real value were available. This situation forced VPRO to hire new staff (around 20) to create and maintain a whole range of new tools; they included programmers, graphic designers, and staff supporting the users of the new tools. Most of the tools were created in 1994 and ranged from games for children to new content for television and radio. They relied primarily on the CERN (Conseil Européen pour la Recherche Nucléaire) server and the Common Gateway Interface (CGI), a standard for interfacing external applications with information servers, like Web servers.

However, the VPRO Web site team faced a difficult capacity problem. All content on the site was based on CGI and was dynamically generated; by continuously promoting the Web site on television, the site was attracting many visitors, easily reaching 200,000 hits a day. At that time, the infrastructure was simply not able to deal with this amount of traffic, and the VPRO Web site faced serious performance problems as a result. Something had to be done to address this problem.

■ The MMBase Management Committee worked hard to create a community ■

Collaboration with Sun Microsystems, Inc.

In late 1994, a solution presented itself when VPRO was invited to join a testing program initiated by Sun Microsystems, Inc. for a project called OAK. OAK is the software language now better known as Java. At the outset, Sun asked the VPRO team to write client-side applets. Yet the VPRO team soon realized there were strong arguments to redesign all the server-side code in OAK because CGI simply did not perform well in the UNIX environment. With CGI, a fork had to be created each time a request was made to an external program. The team realized it would be better to "extend" the server and make the functions part of the server itself. The solution was to build an architecture based on "workers" and modules, similar to the usage today of servlets and persistent beans. These changes created a huge boost in both performance and flexibility on the server side for the VPRO projects.

At the time, another concern was the lack of Java Web servers. VPRO decided to craft a new Web server called Java HTTPD. By the time Java was released to the general public, most VPRO-managed Web sites were already running on server-side Java platforms.

The creation of MMBase software

In time, all of VPRO's tools, servers, scripting language, and database connectors were recoded in Java and deployed on VPRO's Java-based server called James. Typically, the code became unmanageable as both the Web and the number of projects grew. The team decided it was again necessary to rework all the code of each major project, an effort that had, as we will explain, mixed results.

In 1996, during a large Dutch festival which VPRO covered for radio, television, and the Internet, it became clear that something in their Internet system was faulty when the system "crashed." For three days the technical staff was unable to stabilize the servers. VPRO had anticipated such problems when using new technology in 1994, but not two years later. The system had to be redesigned, and the result was the creation of a new extension to James, to be called MMBase. After months of research the goals of the extension were defined:

- Create a way to store all content of all projects in a consistent manner;
- Implement an extensible flexible structure to store content;
- Create a tool for content creators to add, delete, and change content;
- Separate content from presentation;
- Create a "metatag" schema (i.e., define a data model wherein there is meta-data for each object of any type);
- Ensure that the system works across multiple machines (i.e., clusters); and
- Make sure no content (audio, video, or graphics) is destroyed by making it "Web ready"—that is, store all content in a high quality format.

Development of MMBase started in late 1996, and the first version of the system was in use in 1997. Back then, MMBase was a module in the James Web server. As time progressed, more of the identified goals were achieved, and gradually the outlines of MMBase as a CMS became visible.

Opening the source code

Parallel to the development and improvement of MMBase, the VPRO Web site began to gain acclaim and attract visitors worldwide. These visitors also discovered the MMBase development team. The team was proud of its product and created demos, which it showed to visitors. As a result, a growing number of requests to use the software arrived. Apart from being a publicly funded organization, saying no to such requests went against the VPRO corporate culture.

In 1999, VPRO took the first steps to make the MMBase software open source. There were two

main motivators for this move: Firstly, VPRO had become dependent on MMBase, but being the sole creator, maintainer, and user of the software posed a continuity risk. Opening up the source code would enlarge the user base and attract others to participate in development and maintenance. Secondly, MMBase was created using public funds which, VPRO reasoned, should in some way be returned to the public. Making the code open source was one way to return benefit to the public domain.

One developer was assigned the task of preparing some 500,000 lines of source code for its first open-source release. The primary challenge was to make MMBase less VPRO-specific and more generic. A secondary challenge was to ensure platform independence. In addition, an open-source infrastructure was required: a Web site, documentation, maintainers, and communication facilities such as mailing lists were set up in just five weeks. And of course a logo was created, which remains unchanged. The focus of the Web site and (early) communication was not on "selling a product," but on "spotlighting the concept."

The developers who had worked on MMBase for almost five years were very aware of the dilemma they faced. Further improvements of the software before the first open-source release would mean that they would continue to be isolated for some time. They also knew that they would never feel that MMBase was finished. They therefore decided to make the release quickly and accept the consequences. They hoped that many other developers would join them and want to contribute to the collaborative effort.

The creation and rise of the MMBase community

The first open-source release of MMBase was on April 3, 2000. The Web site that hosted the first release is still home to the MMBase community.[8] The software was licensed under MPL.[9] To help decide which license would be issued with MMBase and what the community structure would be, the team studied a number of open-source communities, especially the Apache** community. The MPL license was chosen because it allowed more freedom and opportunity for entrepreneurial initiative compared to the more popular GPL.[10] The rules and regulations governing the MMBase community were based on those for Apache; in fact, the Apache community was approached directly for permission to do so.

In line with the Apache community, the MMBase Management Committee (MMC) was set up. The first MMC consisted of four experienced project leaders: two from VPRO, one from an independent software vendor, and one from another public broadcaster. A year later, a fifth member was added. Membership in the MMC rotates annually. Each year the two longest-serving members give up their position, and two new members are chosen from among the "committers." To become a committer a developer has to be nominated by the current committers. The committers then vote to decide if a candidate developer will be awarded committer status. Once a developer has received committer status, he or she can add source code directly to the CVS (Concurrent Versions System).

The MMC worked hard to create a community and spent much energy in coordination. However, it soon realized that for the community to mature, more was required than a robust technical product and a developers' community. By late 2001, a number of organizations were discussing the desirability of creating a foundation that could support the roles of coordination, marketing, answering questions, responding to requests for information (RFIs), and establishing user collaboration and knowledge exchange. The MMBase Foundation was established in 2002.

The MMBase Foundation plays a key role in promoting and informing the public about MMBase and about open-source software and open standards in general. A nonprofit institute, its main objective is to facilitate the MMBase CMS for open-source usage with the related community, focusing on stability and innovation. Today the Foundation has grown to include a few dozen (commercial) partners. It has a CEO, a board, and an advisory committee of partners, developers (the MMC), and users.

The Foundation's primary focus is on the interests of the users and commercial partners, as the developers have their own organizational structure, but it supports the developers where and if necessary. The Foundation's main activities can be summarized as follows:

- *Coordination*—Bridging the gap that frequently arises between developers and users. Furthermore, the foundation creates a long-term vision for the software in close collaboration with the developers' community.
- *Moderation*—Moderation between the supply of MMBase software and the demand for it.
- *Knowledge transfer*—Taking an active role in disseminating knowledge, giving presentations about MMBase, organizing events, and providing training.
- *Documentation*—Active participation in the creation of documentation for the developers' community, service providers, and users.
- *Marketing and promotion*—Taking an active role to ensure steady growth of the MMBase community.
- *Research and development*—Stimulating and initiating research and development projects.
- *Collaborative development*—Organizing regular meetings with users and service providers to stimulate collective development efforts and assignments. An example is the annual MMBase event.

THE MMBASE ARCHITECTURE

This section discusses the basics of the MMBase architecture. It touches upon five elements in the architecture: object orientation, implementation languages, database abstraction, application server, and publication. The strength and presence of these elements, making MMBase a flexible and well-featured CMS solution, are a major reason for its adoption.

Object orientation

MMBase has made one concept central: namely, that each content item within the system is viewed as an object. This means that a person or program is represented by an object, rather than by a document describing them. In the world view of MMBase, content is an object cloud. Behind this simple yet powerful design is the idea that content stored today may be required for other uses in the future, on different channels, so it must be stored in an open format with relationships to other stored content. Publishing the content today (or in the future) is only the next step, not the end goal of the process. This sets MMBase apart from page-based and document-based systems that take publishing as their conceptual starting point.

All of the objects to be mapped into the MMBase "cloud" are defined in XML. The types of objects in MMBase have an internal definition because no two databases agree on a clear set of types. All objects in MMBase can be related to each other. These relations are also viewed and treated as objects themselves. The most basic relation is one that connects two objects but that does not contain any content. However, frequently it is not enough to make a relation between two objects. For instance, when adding news items to a magazine, one also needs a way of telling in which sequence the news items should be placed in that magazine. In MMBase, one can use the "posrel" relation for this purpose. The "posrel" relation not only connects two objects, but it also allows users to store an integer value for this relation in the "pos" field of the relation. In general, MMBase allows the creation of relations with whatever fields are needed for any particular content model. For instance, when connecting employees to departments, one can use a relation that contains an extra string field to store information on the functions that employees have in their departments.

Originally, the designers and users of MMBase required an architecture that allowed them to change how content was used because standards are constantly changing, as are user requirements and ideas for using content. Though MMBase has undergone many changes over the past eight years, two fundamental precepts have endured. The first is the base concept of storing content as objects with relationships (the result being that even the oldest installations can be kept compatible). The second enduring precept is that parts of the proprietary environment are replaced when clear standards have emerged and been accepted.

Implementation languages

XML and Java were the first two standards that became clear starting points for tooling during the conception of MMBase. All objects in MMBase are made up of a combination of both data and functions, and the best tools available for implementation during MMBase's development were XML for data and Java for functions. Both are open, platform-independent, well-defined standards with broad support within both the commercial and open-source community. Over time, both Java and XML have continued to change; MMBase changes with them when market forces indicate the time is

right, for example, the shift from Java Version 1.0.2 to Version 1.4. Taking these two technologies as a base reduces the learning curve for many people to join the MMBase community because XML and Java have a large following and are well understood within the developers' community.

Database abstraction

MMBase stores and maintains the object cloud through a storage manager, and queries the cloud using a search query handler. These are two more or less mutually independent abstraction frameworks, and they are discussed in some detail later. The two frameworks ensure that users are not confined to one database, which would result in a vendor lock-in. They also attract more people to use MMBase because most companies have one database brand for all of their systems; whereas, the open source community uses a number of popular databases such as MySQL and PostgreSQL.

There are many standards in the database world, like Structured Query Language 92 (SQL92)[11] and JDBC** (Java Database Connectivity),[12] but almost all larger databases break or extend these standards as they see fit. To deal with this problem, MMBase currently has its own object mapping mixed with standards such as JDBC to get a completely transparent storage model for its supported databases. Furthermore, the MMBase clouds can be moved to any supported database without any change in content, software, or scripting, and new databases can be supported if needed by adding new interfaces.

The storage-manager framework takes care of the creation of tables, inserts, and updates. Objects are mapped to the database depending on whether and how that database supports object-oriented data storage. In cases of non-object-oriented databases, MMBase simulates this structure by duplicating data in tables, specifically in the tables for the object types from which other types inherit: 'Object' and 'Insrel'. An alternate implementation is in development. This implementation will make use of 'views'. Because the storage layer is pluggable, it is possible to play on the strengths of the database, for instance, by using the object-oriented features of a database like Informix. Adapting the storage layer to support a new database can be done through implementation of a new Storage Manager class, or through the configuration of the existing base implementations, that is, the Database Storage Manager and the Relational Database Storage Manager. The existing base implementations include a wide variety of options, a set of query template strings which allow

■ Each content item within the system is viewed as an object ■

for variation in insert/update/create query syntaxes, and the mapping of MMBase field types to actual types in the tables.

Because support for the storage of binary data in JDBC varies widely among databases (even among versions of the same database), and because it often introduces dependency from external (often license-protected) source code, MMBase allows an option to store binary data outside the database, on a file system. This method may also be preferred for practical reasons (such as a better facility for backups). Local implementations are still possible, to extend the layer to include binary data in the database if there is no proper support in the JDBC driver for that database.

The query-handler framework is used to retrieve data from the database. It features query objects that can be used to model a SQL "select" query. Those objects can be handled and filled programmatically, and only on execution of the statement are they translated to actual SQL. Different databases may have a slightly variant query handler to perform this translation. The default implementations shipped with MMBase support a common-denominator version of SQL supported by all common databases. The query objects are used as a key for query result caching, substantially reducing database load.

The storage-manager framework is accessed only from the core in MMBase. External programs never deal with that layer directly, but instead pass their requests for changes to MMBase through a "bridge" called the MMBase Cloud Interface (MMCI). That interface also allows the creation of query objects as outlined earlier. MMBase has created its own interface, which does not yet conform to the Java Specification Request 170 (JSR 170) standard.[13] There are a few practical reasons why this is so. First, the MMCI was created before the JSR 170

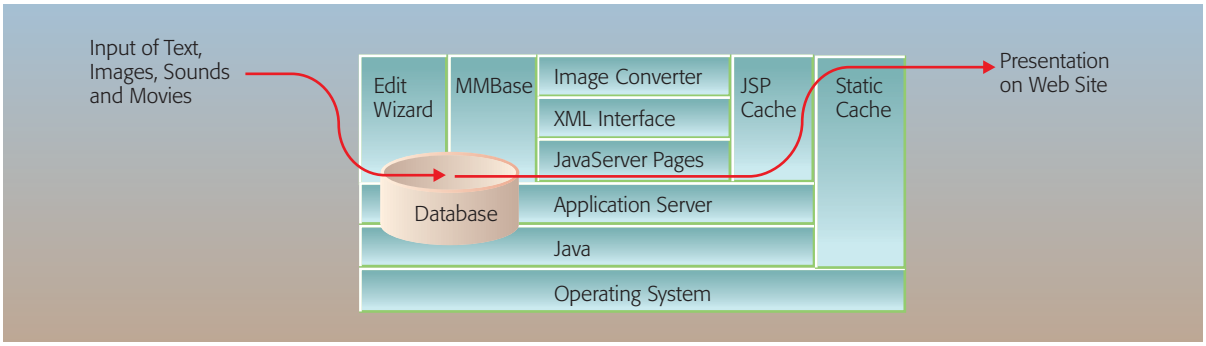

Figure 1
Overview of MMBase Web server infrastructure

existed. Second, the standard has been received with significant criticism. Third, changing the MMCI would require much effort, due to the changes that need to be made to other products that use the MMCI, of which the MMBase tag library is a good example.

Server

Like most CMS systems, when used to support Web sites, MMBase is paired with a (Java-enabled) Web server or an application server. See ***Figure 1*** for an overview of a Web server infrastructure with MMBase. Originally, VPRO started with James, a Web server created by the VPRO team. After the servlet API (application programming interface) gained acceptance, MMBase was adapted to support the various versions of that API. Currently, the servlet API is only a small part of the J2EE[14] specifications, a set that is not always clear because it covers a large area and is subject to constant change. However, J2EE is a good partner for MMBase because it demands a lot of the server resources that MMBase is running on.

MMBase is a J2EE application. As more of the J2EE specifications become stable and accepted, custom layers will be swapped out of MMBase and handled by the J2EE server. A good example is database abstraction. Currently, J2EE is not strong enough to cover the database abstraction now available in MMBase, but with successive iterations of the specification, the gap is shrinking. At some point, MMBase database abstraction may be replaced by abstractions provided by the J2EE server of choice.

MMBase is used as a Web application on an application server. Content is stored and maintained via import components, like the edit wizards or the XML importer. Different components can be used to extract content from the MMBase content repository. Both internal and external caching mechanisms are available to boost performance.

Publication

Fundamental for a CMS is the manner in which content is translated to the required published form. MMBase is often used to create Web sites. Unless the site in question has a very fixed user-interface design, tools are required to help the site developers create the necessary Web pages (or documents, streams, etc.). The number of frameworks available on the Internet, both large and small, is almost limitless; the same applies for the number of in-house created tools. MMBase started out with its own scripting language. As with most of these language types, its support is paramount. Most Web developers are trained in one or two of the main scripting systems, for example Active Server Pages** (ASP), Personal Home Pages (PHP), or JSP, so it is most efficient to make use of this familiarity. To enable broad support, a special layer between the MMBase core engine and the outside world was added, namely, the MMCI.

Over the years, a taglib has been created for MMBase for use in combination with other taglib sets or independently, to create dynamic Web pages filled with content stored in MMBase back-end servers. Many example packages are provided in the distribution set, which serves as a starting point for new developers of MMBase-driven Web sites. See ***Figure 2*** for the role of the MMCI, the taglib, and packages in the MMBase architecture. (Dove, one of the components shown in the figure, is a support class that

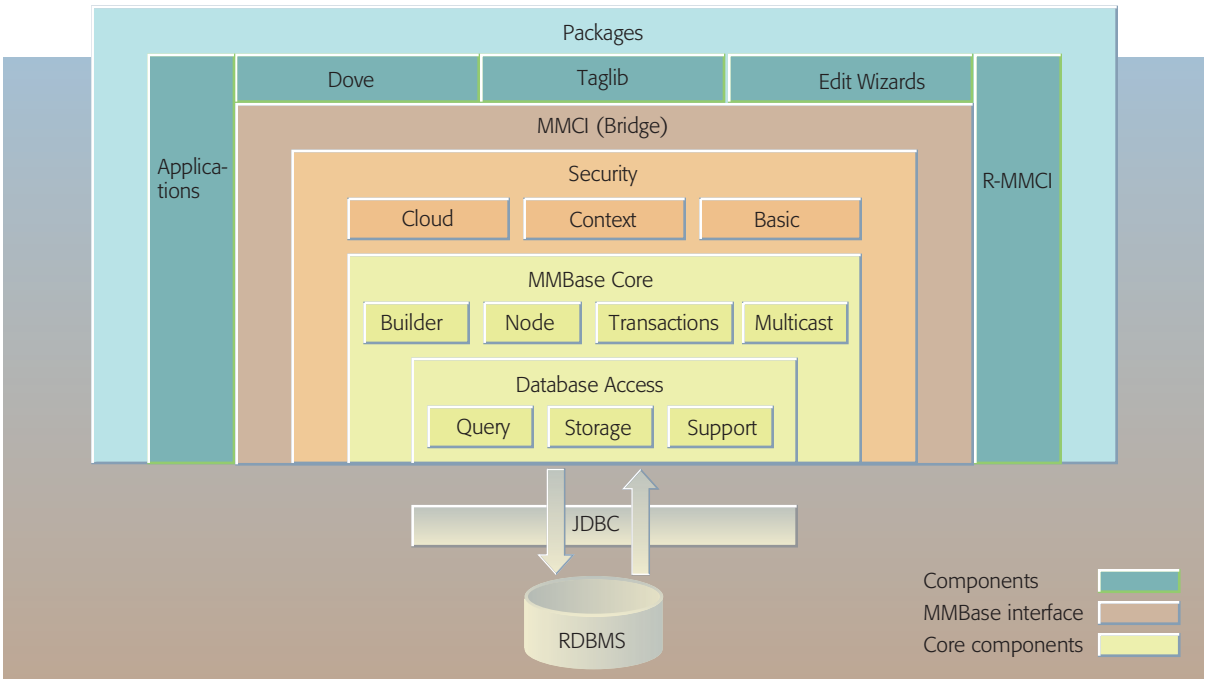

Figure 2
MMBase architecture

supplies a communication protocol for MMBase.) Over time, the core developers will adapt the scripting tools when new standard solutions to problems become available (currently the effects of JSP2 and how to convert to it are being hotly debated).

As shown in Figure 2, the MMBase architecture consists of five layers: (1) the Database Access layer, which connects MMBase to a database via the JDBC interface, (2) the MMBase Core, which maps the relational database into an object-oriented abstraction layer, (3) the Security layer, which provides a pluggable security interface for user authorization and authentication, (4) the MMCI, which facilitates communication between the Core and Security layer and the components, and (5) the components, which provide the basic building blocks to import into, edit, and extract from the MMBase object repository. Packages are MMBase (Web) applications, which use the components and offer a ready-to-use item of functionality like a forum, chat, or e-learning platform.

MMBASE USE CASES

As the MMBase software matured, a growing number of users, many from the public domain, began to employ it. One of the first users of MMBase was the Dutch Evangelical Broadcasting Organization (EO). It adopted MMBase for a number of reasons: (1) MMBase is open, allowing the organization to make its own modifications to the software; (2) MMBase is based on Java and XML, thus it is standards-based and platform-independent; (3) MMBase is designed to be flexible; and (4) MMBase was already successfully used to support VPRO's Web site, which meant that it would also likely fulfill EO's requirements.

After the EO, many more organizations began to adopt MMBase. A selection of these organizations follows. These case studies will primarily focus on the reasons that led to the adoption of MMBase and on the lessons that were learned.

Ilse Media

In the late nineties, Ilse was the leading search engine for the Dutch-language region. In 2000, Ilse merged with the magazine division of VNU Publishers, which is now part of Sanoma. The core competence of Ilse Media lay in search engine technology; the organization had much less knowledge of Web site technology. This posed a challenge because Ilse Media was now being asked to host the Web sites for all VNU-published

magazines in an efficient and cost-effective way. In the first months of 2001, Ilse Media conducted a CMS selection process. The programs under review included Mediasurface**, Roxen, Gauss, Tridion**, Zope**, and MMBase. MMBase was selected because it offered the latest technology, such as JSP templating, object orientation, platform independence, and XML import and export; it had a robust and growing installed base in the Netherlands; it was open source, so that an MMBase implementation could easily be exported to another CMS; and it came without licensing fees.

Within a few months, a group of three developers, one designer, and a project manager was able to build a dozen Web sites for VNU magazines. MMBase proved efficient in implementing tailor-made solutions generated from a generic blueprint. The main reason for this is MMBase's object-oriented architecture, which eases the implementation of generic building blocks. At the same time, MMBase offered the flexibility to diverge from and add to the blueprint where necessary.

In 2002, an important Dutch online news provider, nu.nl, also decided to adopt MMBase. This project resulted in a valuable spin-off called Wodan.[15] Wodan is a reverse proxy module for the Apache Web browser. Web clients retrieve content from the Wodan module instead of from the Web server. The module is especially useful in environments where a large amount of CPU power is required, but where the pages do not change often.

The city of Amsterdam

In 2000, Amsterdam faced an organizational problem. The city had an enormous number of Web sites, which were created and maintained by numerous people. The infrastructure had become a convoluted structure in which everything was connected in multiple ways. Each city district and department and their related agencies had their own Web sites based on their own preferences. Furthermore, the city knew from experience that top-down control to change this structure was unlikely to have positive effects. Traditionally, the departments are accustomed to a great deal of autonomy, which they value highly.

Amsterdam was fortunate to have an IT-conscious councillor for economic affairs and a capable senior project manager at its central information facility. Together, they decided to transform the city's IT infrastructure and to make the move to open source. The main reasons were that open source would allow the city to maintain independence from a supplier and would enable the city to share any written source code with the public. The adoption of open source was also considered a signal to the outside world, as it would enhance the openness of the municipality. The choice to move to MMBase was based on two factors. First, content in MMBase is stored in XML. This would enable the city to switch to another CMS if MMBase proved to be qualitatively inadequate. Second, the origin of the system in the broadcasting industry seemed perfect considering that city council meetings were to be broadcast on the Web in the future. One of those directly involved in the decision commented, "It seemed flexible and future minded, and for the rest, it was intuition."

For municipal use, the application *Web in a Box* (WIAB) was created, an MMBase-based environment with advanced editing facilities. The first version of WIAB was released in February 2002 and licensed with GPL. This decision resulted in some controversy. Although MPL and GPL are both open source licenses, they are not compatible. A special clause in GPL forbids developers from incorporating GPL-licensed source code into software licensed under MPL.[16] It took a year before the WIAB license was changed to MPL, which enables the WIAB project and the MMBase community to benefit from each other's activities and improvements. It also ensures that WIAB can remain compatible with the MMBase core.

In late 2003, a large shared development fund was established and contributions were provided by every agency using WIAB. In return, they received a strong say in the identification and definition of new requirements for future versions of WIAB. The system is gradually growing into a real network of interconnected databases, and the benefits of interoperability and the opportunities for knowledge exchange and Web service engineering are gradually becoming visible within the city as a whole. Currently, more than 28 independent agencies use the same components for Internet services, and seven agencies manage their whole intranet based on WIAB. The benefits of a developer community consisting of a large city's administration and end users and characterized by high coherence and common understanding of Web technologies are

already visible. The first advantage which is not obvious is the speed of useful metatagging among civil servants and the fact that one search engine is available to all agencies without any technical indexing problems. The organically grown balance between centralized databases and distributed databases using multiple retrieval mechanisms is something any systems architect would like to accomplish in such a complex political climate. Integration and compatibility with GIS (geographic information systems) appears to be easier than expected.

The initiators also expressed some of the disadvantages. They felt that adoption had necessitated complex organizational change and the creation of a culture of interdepartmental collaboration beyond that which the IT managers were accustomed to handling. They also mentioned some of the difficulties in setting up the maintenance organization. This differed from the situation in which a "simple" vendor would have been chosen. Finally, in hindsight, it would have been valuable to have known more about license issues and the fact that one type can exclude another.

Vodafone

The Vodafone Group commissioned research, development, and implementation of a software application for conferences and events aimed to support business and public events, providing extra information and assistance to attendees via mobile telephone. The content for such a system needs to be extremely dynamic, and the ability to straightforwardly make relationships between content items is crucial. MMBase proved ideal for these requirements. Among the major reasons for Vodafone's move to MMBase were the following:

- No commercial license costs
- Proven stability and performance
- Availability of support from MMBase developers' and software vendor communities
- Low development and maintenance costs, compared to other CMSes

Another motivation for Vodafone was the ability to combine MMBase with the open-source Xmedia portal, implemented at various other sites. The Xmedia portal allows MMBase-controlled content to be made available over multiple channels, such as mobile telephone and digital television, and also provides enhanced security and a service-oriented architecture.

Vodafone is converting more of its public systems to MMBase and is currently engaged in a set of pilot projects to examine the associated technical and business issues.

■ The open-source development model is based on the principle that knowledge is free and should be shared ■

Didactor

The Mediator Group is an innovator in the field of education and specializes in e-learning technology. Mediator serves organizations that want to use IT to support their primary education process. The Mediator Group wanted to create a simple Web-based e-learning platform that would free organizations to spend their innovation budgets on truly educational innovation. It faced a challenge in deciding which development platform to choose. It soon recognized open source as a promising route, as this would reduce commercial licensing costs for clients. Mediator approached a number of organizations and asked them to write proposals for a new platform. They defined three criteria: (1) the platform should be open source, (2) it should be based on the learning objects methodology, and (3) it should meet open and didactic standards.

In the meantime, a client asked The Mediator Group to collaborate on and provide support for an MMBase implementation. Mediator became involved in the MMBase community and soon realized that MMBase was suitable to create their generic e-learning platform. Their main reasons for selecting MMBase were that it had been developed by media-rich organizations and that it had a solid user and developer base. The Evangelical Broadcasting Organization joined the development effort, as it too was interested in creating an e-learning platform. Their combined effort resulted in creation of the open-source platform Didactor.

Within as well as outside of the MMbase community, several parties showed interest in Didactor

initially. However, very few actually supported Didactor by co-developing or funding. This resulted in the odd situation where the founding partners made large investments in terms of time and budget; whereas, others simply waited for Didactor to be finished. This is one of the less positive characteristics of open-source software, as it supports and perhaps even stimulates "free-riding." Software development depends on companies and individuals who are willing to do the job.

The platform is developed from a didactic perspective, which is considered to be one of the most crucial prerequisites for successful implementation of e-learning.[17] Furthermore, it is based on the methodology of *learning objects*, which is an important trend in e-learning.[18] The idea underlying this methodology is that educators can define and create pieces of knowledge and then group them into separate objects. These objects are stored in a database and described with educational meta-data. This structure allows educators to share objects and combine them to create lessons or courses.

The Didactor architecture meets international standards like the IEEE Learning Object Metadata (LOM) standard[19] and the Sharable Content Object Reference Model (SCORM).[20] Developers are now in the process of integrating the flexible generic language IMS (Instructional Management System) Learning Design.[21] The Dutch Open University collaborates with The Mediator Group in this process. Didactor consists of over 20 didactic components, which remain close to the components in MMBase; that is, they use the same forum and chat components. New modules created in Didactor are programmed in the same way as the components in MMBase. New modules are donated to the community.

Didactor is used by institutes for vocational training, as well as by companies seeking to redefine their learning methods. ConQuaestor, which used to be part of IBM Business Consulting Services, has also adopted Didactor. The Mediator Group and NETg, which is part of the Thomson Corporation, formed a strategic partnership to deliver NETg's roughly 3,000 e-learning courses through Didactor. The Free University, one of the leading universities in the Netherlands, is involved as a research partner in this project, measuring the efficiency of the platform in a learning environment.

One of the lessons learned by Mediator, starting with the development of its components, was that it was not easy to begin using MMBase. The most important reason for this was probably the fact that one cannot simply contact *the* MMBase vendor. There is no single point of contact. Corporate users can only rely on one of the supporting partners, but this can become problematic when their customers demand tight time schedules. The open-source communities, which consist of many volunteers who participate for various reasons, including fun and ideology, are less concerned with deadlines than the corporate users.[22] For Mediator, the solution for this issue was to become involved in the community and to build relationships with members based on trust. This way others will be more inclined to help Mediator when certain problems are experienced or deadlines have to be met.

Because Didactor is mission-critical for organizations, service delivery has become increasingly important. The main parties supporting the product are thus in the process of creating a shared service center. In the shared service center, trained professionals will manage the Didactor service agreements and be responsible for Web site hosting, fixing bugs, making updates available, and providing continuous service (i.e., "24/7"). For the Mediator Group, this is one of the most important issues affecting any open-source product: the variety and quality of services has to be the equal to or even better than that of proprietary products in order to be able to compete with them.

CHALLENGES FACING THE COMMUNITY

Three distinct challenges have arisen from the growth of the MMBase community. The first is how to deal with the potentially growing number of free riders. The second is to define and separate the core from the packages (previously called MMBase applications). The third challenge is documentation. These challenges are described in the following subsections.

Free riders, the other side of openness

The open-source development model is based on the principle that knowledge is free and should be shared. Richard Stallman compares the art of writing software with that of writing a symphony. He argues that Beethoven could not have written music without building on the ideas of other composers, and the same is true for software developers. They also

build on the ideas of other developers, and therefore, he argues, it is important for them to share and have access to software source code.[23]

However, having source code freely available also creates problems. Anyone can download, install, and modify source code. Some open-source licenses even allow users to create a proprietary, that is, closed, version of the software. These factors result in the potential for organizations to "free ride": they can take without giving.

Free-riding behavior is not unusual. It occurs in many open-source communities, and the MMBase community is not immune. It is not inherently problematic if a small number of users decide to free ride. However, a community like MMBase needs to retain a critical mass of users and software-developing companies who are willing to participate in the maintenance and improvement of the software. The question is, "How?" To understand the value of such involvement for developers and maintainers and how such communities can protect themselves, it is important to understand how participants are attracted to the community in the first place. Basically, there are two types of participants: the commercial vendors that build their business model on the open-source application (in this case, MMBase) and the users.

In the past, one would not have expected a commercial vendor to become a partner in an open-source community. Freely sharing and contributing improvements made at their own expense would seem counter to commercial principles. Traditionally, vendors have had an incentive to keep modified source code private, as this is a recognized way to build and protect a competitive advantage in the software market. However, to date, most commercial vendors of MMBase-related material have become partners in the community; most vendors want to join the community. Several factors help to explain why:

1. *The community's mediator role in the user market*—Firstly, the MMBase community is well known in the Netherlands and enjoys a good reputation. For this reason, many potential users first approach the community rather than a commercial vendor to find out more about the software. The community brings these users—potential clients—into contact with vendors that can customize the software to meet the users' needs. Thus, vendors have an incentive to collaborate within the community, as it is an important channel through which to reach potential customers.

2. *The community as source of continuity*—Secondly, some users (including major industry and government concerns) have made a strategic decision to use open-source software. They do so because (for example) they want to reduce licensing costs and dependency on commercial vendors. Such users often require commercial vendors to return any additions or modifications they contract to the community to ensure continuity.

3. *The community as learning environment for users*—The motivation for users to participate varies. Many smaller companies want a proven solution at minimum cost and might hire a commercial vendor to customize the software or decide to install the software themselves, using it as a finished product. However, for these users there is also an incentive to be involved in the community. Lack of proactive participation in the community could lead to a lack of attention from the developers and maintainers the next time an update is required or when they need information or experience a problem.

Thus, a community like that of MMBase has informal mechanisms to overcome the problem of free riding. These are not yet predictable processes, however, and time will tell whether they are solid enough to continue to motivate vendors and users to collaborate and invest in the community.

Managing flexibility: the core versus the packages

During the conception of MMBase, things were simple. There was one version of MMBase and one community of developers. These developers knew everything there was to know about the product. They were the core developers. New developments were added directly to MMBase, which slowly evolved into a monolithic piece of software. This process remained manageable due to the relatively small size of the community.

However, as time progressed, increasing numbers of organizations began to use the product and became

involved in its development and maintenance. New functionality was added. Application developers created new packages for use on top of MMBase. This sometimes meant that changes to MMBase were required, and these were integrated into privately owned versions of MMBase. Frequently, these changes were not integrated into the official MMBase releases. Therefore, every time an official release was made, an exponential time-consuming migration effort was required to maintain the changes.

The growing number of developers caused a steady rise in the number of packages available for MMBase, which extended the software's functionality. Packages included new editors, a media package enabling a comprehensive and generic way to handle audio and video files, and an e-mail package allowing users to send e-mail directly from the CMS. Many of these packages were added to the official MMBase release, causing the official release to grow to an almost unmanageable size. A growing number of packages and relationships had to be managed and stabilized before each new release. As a result, it became increasingly difficult and time-consuming to create new releases.

The community of developers realized that a clear separation was required between the core of MMBase and the packages built on top of this core (i.e., a more service-oriented architecture was needed). The separation was initiated in 2003 and currently dominates development activities. The separation has as its main goal the creation of a smaller, more maintainable core; creating an architecture that requires fewer changes at the core, resulting in a more stable core; increasing the number of official releases of MMBase packages while shortening the release cycles; increasing compatibility between releases; increasing coordination among active developers as a result of more manageable components; attracting more developers and teams to participate, as the effort required for effective participation is decreased; and defining solid interfaces between the core and the packages, ensuring that application developers can efficiently create new packages and improve existing ones.

The road to a complete separation between the core and packages is not without problems, however. There are three major problems: The first is distinguishing the core from the packages; that is, how to decide what belongs to the core of MMBase and what should be part of a package. The underlying problem is the lack of consensus among developers about what MMBase exactly is. The developers can be roughly divided into four camps. The first camp argues that MMBase is an object database; the second considers it to be a framework; the third views MMBase as a CMS; and the fourth sees it as a combination of the three. These different conceptions result in varying opinions about what belongs in the core and what does not. For example, if MMBase is viewed as an object database, much of the source code can be removed from the core, but from the CMS perspective, features like workflow and security would have to be included in the core.

The second problem is that of distribution and packaging. The challenge is to create a framework and format for how packages are to be created, distributed, and installed. The third problem relates to legacy code and backward compatibility. Part of the code currently in the MMBase core is legacy code. Only a limited number of organizations, those that have used MMBase since the beginning, still require this code. An example is the scripting language SCAN that is used to create templates. SCAN was replaced by the MMBase taglib (a standard technique to create templates). Yet, as some organizations still use the SCAN templates, this code needs to remain available for them.

The community is well aware of the challenges posed by separation of the MMBase core code from its packages and is working hard to solve the problems to achieve a healthy separation between the two. To this end two projects have been created, the first of which is cleaning. The goal of this project is "refactoring" (removing or rewriting old code). Some code will be moved into packages, redundant code will be completely removed, and some code will be rewritten for improved stability, performance, and maintainability. Already, much code has been "tagged" during the last two years. Tagging involves adding keywords to parts of code to describe future actions required (e.g., "rewrite," "add better documentation," and "remove"). These actions can be complex; hence the two-phase process of tagging and then implementation.

The second project is called packaging. Its aim is to simplify the installation process and management of

the packages. The result will be a small core that has basic functionality. Users would then add functionality by installing the packages, supported by a simple and uniform installation procedure.

Managing reusability: Writing and maintaining documentation

As the MMBase software matures and becomes more complex and extensive, the importance of documentation increases. Lerner and Tirole identified the creation of documentation as a key challenge facing many open-source communities: "Another challenge has been the apparently lesser emphasis on documentation... in at least some open-source projects."[24] This aptly describes the challenge facing the MMBase community with regard to documentation. Many developers prefer to write new source code and create new functionality and are not intrinsically motivated to write documentation. Users involved in the community seldom have the expertise to write the appropriate level of technical documentation. Moreover, they may have insufficient knowledge of the software and its functionality to do so. Nor do the commercial vendors spending time and money to create new MMBase software always have the long-term vision or incentive to invest in writing documentation for the software they create.

It was only in July 2002 that the MMBase community was able to motivate its participants sufficiently to invest in the creation of documentation. The community initiated the *documentation project* to write and maintain the documentation needed for future MMBase releases. At the start of the project, the XML Docbook[25] was chosen as the standard for writing documentation.

The community has since created and institutionalized two mechanisms to encourage developers to participate in the documentation project and motivate them to write new documentation. The first is the organization of regular face-to-face meetings. Participating developers have found that regular meetings have helped motivate them to create and improve documents. Peer pressure has thus proven a successful way to compensate for any original lack of incentive. At the meetings, developers review existing documents and discuss future directions. The second incentive is the community's new policy regarding releases: new releases are allowed only when the documentation is up to date. Again, the idea is to stimulate developers to participate in the documentation project. As of this writing, the new policy is undergoing initial trials, so whether it will indeed stimulate developers to participate is as yet unproven.

By now, a dozen documents have been written. The documentation currently available provides sufficient information for new potential contributors and users to find their way around the MMBase environment. However, there are still blank spots and a number of "to dos." The future will tell whether the documentation project and the two incentive mechanisms are sufficient to mobilize a critical mass of participants to keep the body of MMBase documentation up to date.

CONCLUSIONS

This paper has discussed how the MMBase CMS was created and how its community took form. It argued that the strength of MMBase is that it is stable, proven, platform-independent, object-oriented, and open source. The continuous growth of the MMBase open-source community and the rising number of its users provides empirical proof of the value of the characteristics mentioned. Furthermore, these two trends highlight the need for and importance of an organizational structure that is able to constantly adapt and evolve to changing requirements and demands. To date, the MMBase community appears to have succeeded in meeting these requirements.

ACKNOWLEDGMENTS

The authors would like to thank the members of the MMBase Management Committee, and in particular Rob Vermeulen and Pierre van Rooden, for their contributions and comments. We would like to thank Jeroen Visser (of IBM Netherlands) for his support and his comments. Finally, many thanks go to Bert Straatman, Michiel Meeuwissen and Rogier Schaaf for their input and Karien Stroucken for her support and coordination of the many activities needed to get this paper published.

CITED REFERENCES AND NOTES

1. M. Noga and F. Kruper, "Optimizing Content Management System Pipelines—Separation and Merging of Concerns," in *Generative Programming and Component Engineering*, Lecture Notes in Computer Science, No. 2487, Springer-Verlag, Heidelberg, Germany (2002), pp. 252–267.
2. M. Bandorf, T. Yoshizawa, Y. Takada, and G. Merbeth, "Enterprise Content Management with Interstage Contentbiz," *Fujitsu Scientific and Technical Journal* **40**, No. 11, 61–73 (2004).
3. A. Somani, D. Choy, and J. Kleewein, "Bringing Together Content and Data Management Systems: Challenges and Opportunities," *IBM Systems Journal* **41**, No. 4, 686–696 (2002).
4. P. Tyma, "Why Are We Using Java Again?" *Communications of the ACM* **41**, No. 6, 38–42 (1998).
5. See, for instance, A. Leroy, "J2EE and .NET: Interoperability with Webservices," Lecture Notes in Computer Science, No. 2952, Springer-Verlag, Heidelberg, Germany (2004), p. 155; and J. Williams, "The Web Services Debate: J2EE vs. .NET," *Communications of the ACM* **46**, No. 6, 58–63 (2003).
6. See also S. Cheong, K. M. Azhar, and M. Hanmandlu, "Personalization of Interactive News through J2EE, XML, XSLT, and SMIL in a Web-Based Multimedia Content Management System," *Proceedings of PCM 2002—Advances in Multimedia Information Processing*, Lecture Notes in Computer Science, No. 2532, pp. 287–294 (2002); and J. D. Litke, "Strategic Implications for Future Content Management Systems," *Society of Motion Picture and Television Engineers Journal* **110**, No. 1, 23–27 (2001).
7. The case of The Mediator Group introduces another reason, namely the lack of one MMBase vendor that is a clear point of reference.
8. MMBase Homepage, http://www.mmbase.org (July 2004).
9. The Mozilla Public License is an open-source license. The latest version of the license can be found at http://www.mozilla.org/MPL/MPL-1.0.html (July 2004).
10. The text of the GPL can be found at http://www.gnu.org/copyleft/gpl.html (July 2004).
11. SQL was first created by IBM Research in the 1970s.
12. See http://java.sun.com/products/jdbc/ (July 2004) for a description of JDBC. Graham Hamilton, Rick Cattell, and Maydene Fisher wrote a manual on JDBC called *JDBC API Tutorial and Reference*, Addison-Wesley Professional, Second Edition (1999).
13. Java Specification Request 170 (JSR 170), http://www.jcp.org/en/jsr/detail?id = 170.
14. For more information, see S. Bodoff, A. Armstrong, J. Ball, and D. Bode Carson, *The J2EE Tutorial, Second Edition*, Addison-Wesley Professional (2002).
15. For a description of Wodan and its community, see http://www.wodan.net/ (July 2004).
16. For a more elaborate description of open-source licenses and their effects, see R. van Wendel de Joode, J. A. de Bruijn, and M. J. G. van Eeten, *Protecting the Virtual Commons: Self-Organizing Open Source Communities and Innovative Intellectual Property Regimes*, T. M. C. Asser Press, The Hague (2003).
17. T. Govindasamy, "Successful Implementation of E-learning Pedagogical Considerations," *Internet and Higher Education* **4**, 287–299 (2002).
18. L. Mortimer, "(Learning) Objects of Desire: Promise and Practicality" (2002), http://www.learningcircuits.org/2002/apr2002/mortimer.html.
19. *Learning Object Metadata*, Learning Technology Standards Committee, Working Group 12, http://ltsc.ieee.org/wg12/ (July 2004).
20. Shareable Content Object Reference Model (SCORM) Initiative, http://xml.coverpages.org/scorm.html (July 2004).
21. The manual can be downloaded from http://www.imsglobal.org/learningdesign/index.cfm (July 2004).
22. See for instance: R. van Wendel de Joode, "Conflicts in Open Source Communities," *Electronic Markets* **14**, No. 2, 104–113 (2004).
23. R. M. Stallman, *Free Software, Free Society: Selected Essays of Richard M. Stallman*, GNU Press, Boston, MA (2002).
24. J. Lerner and J. Tirole, "Some Simple Economics of Open Source," *Journal of Industrial Economics* **50**, No. 2, 197–234 (2002).
25. The Docbook XML software (Version 4.2) and the project can be found at http://www.oasis-open.org/docbook/xml/ (July 2004).

Accepted for publication October 21, 2004.
Published online April 20, 2005.

Joost Becking
The Mediator Group, P.O. Box 59295, 1040 KG Amsterdam, The Netherlands (joost@mediatorgroup.com). Mr. Becking is co-founder and principal educational architect of the open-source e-learning platform Didactor, and is responsible for open-source strategy. He specifically focuses on expanding the worldwide developer and user community. Mr. Becking co-authored the book *Internet Method* on integrating Web applications in primary business processes. His interests are in the areas of open-source business modeling, standardization, and learning object technology. He strongly believes that IT can do a great deal of good in educational settings, but only if it becomes part of the educational strategy for primary processes, and so he strongly believes in the power of blended learning.

Steve Course
Quantiq Xmedia B.V., 11–13 Koninginneweg, 1217 KP Hilversum, The Netherlands (steve.course@quantiq.com). Mr. Course is CTO and co-founder of Quantiq Xmedia. Quantiq brings IT and marketing and communication specialists together to create value for clients, helping them become cross-media driven by developing concepts and technical implementations. Previously, he was the technical manager at Vodafone, where he was responsible for the implementation of the Vodafone *Live!* platform and the Vizzavi portal.

Gerard van Enk
Million Pieces, Kiekstraat 167, 1087 GT Amsterdam, The Netherlands (gerard@mmbase.org). Mr. van Euk is a member of the MMBase Management Committee and MMBase release manager. His company, Million Pieces, specializes in Internet-related development and consultancy. He is the senior Web application developer for the Evangelical Broadcasting Organization, a Dutch public broadcaster. His interests include Java, open source (software, communities, licenses, etc.), and free culture.

Hendrik Theodoor Hangyi
MMatch/MMBase Consultancy and Implementation, Hommelstraat 9A, 3061 VA, Rotterdam, The Netherlands (hangyi@xs4all.nl). Mr. Hangyi is the project manager of the MMBase documentation project. He has worked with MMBase since 2001, focusing mainly on consultancy, implementation, and education. Before starting his own firm, he worked as a consultant at CMG and Arthur Andersen.

Jo Lahaye
MMBase Foundation, Neuweg 83, 1214 GM Hilversum, The Netherlands, (Jo@mmbase.org). Mr. Lahaye is the first CEO of The MMBase Foundation. He has a background in journalism, worked as an ICT (information and communication technology) project manager, and implemented many different CMSes, especially in higher education. He also advises many organizations on the use of open-source software and is the author of many (Dutch) articles on open-source and open-standards issues as well as (software) patents.

Daniel Ockeloen
Submarine.nl crossmedia production company, Rapenburgerstraat 109, 1011VL Amsterdam (daniel@submarine.nl). Mr. Ockeloen is a software designer. With Rico Jansen, he was responsible for the development of James and MMBase for VPRO. In 1999 and 2000, he prepared MMBase for open-source release. In 2000, he left VPRO to co-found the company Submarine, which produces cross-media products. Mr. Ockeloen was part of the MMBase Management Committee for the first few years of the open-source phase. Currently, he works for Submarine and several other supporting organizations to create new extensions for MMBase. He plays a positive role in the community by giving talks and lectures when time allows. He spends much of his free time with music and movies in his self-built home theater.

Rob Peters
University of Amsterdam/Faculty of Law, Leibniz Center for Law, Oude Manhuispoort 4, PO Box 1030, 1000BA Amsterdam, The Netherlands (rob@lri.jur.uva.nl). Mr. Peters organized the first conference on electronic commerce in Europe in 1993. He was responsible for building some of the Netherlands' major government sites, like www.PortofRotterdam.com and www.overheid.nl. He is now in charge of a number of European research projects, such as www.addwijzer.org. Currently, he works for a small consultancy firm for e-government called Zenc, specializing in the link between content production and content navigation. At the University of Amsterdam, he is completing a Ph.D. degree on open source as a catalyst for e-government innovation.

Hessel Rosbergen
Finalist IT Group, 3 Wibautstraat 9th floor, 1091CH Amsterdam, The Netherlands (Hessel.Rosbergen@finalist.com). Mr. Rosbergen is account manager at Finalist IT Group. Finalist specializes in technical consulting and development services for the Internet, based on Java technology. Most of the projects are based on open-source technology, such as MMBase and JBOSS. Because Finalist also uses SUN, HP, BEA, IBM and X-Hive, different technologies can be thoroughly compared.

Ruben van Wendel de Joode
Delft, University of Technology, Faculty of Technology, Policy and Management, P.O. Box 5015, 2600 GA Delft, The Netherlands (rubenw@tbm.tudelft.nl). Mr. van Wendel de Joode is a Ph.D. degree student. His research focuses on the organization of popular open-source communities like Apache and Linux. He received two grants from the Netherlands Organization for Scientific Research (NWO) for research related to open-source communities. The first grant was to study the interplay between intellectual property rights and open-source communities. The results are published in *Governing the Virtual Commons* (Cambridge University Press, 2003). He has written numerous articles on open source, which have appeared in journals like *Electronic Markets, Knowledge, Technology and Policy,* and the *International Journal of IT Standards & Standardisation Research.* ■