The use of open-source software in the IBM corporate portal

N. Yan
D. Leip
K. Gupta

This paper describes the role that open-source software has played in the evolution of the IBM corporate portal; in particular, the use of the Apache™ Web server, Perl, XML, and Struts.

The growing popularity of open-source software in recent years has led to its increased acceptance in the enterprise environment. In a recent informal survey, a large percentage of respondents (60 percent) reported using a mix of open-source and commercial software products at their companies.[1] The exclusive use of open-source software appears to be rather limited (only 2 percent of the respondents). The IBM corporate Web site, which originally used only IBM proprietary software, has gradually incorporated a number of major open-source software components. These have delivered cost savings, contributed to the rapid adoption of new technologies, and opened up new business opportunities. The IBM corporate portal, www.ibm.com, provides a gateway to corporate information and services. It serves customers, business partners, investors, and the general public in 84 countries around the world.

To satisfy the requirements for availability, the portal infrastructure is distributed across three separate locations as illustrated in ***Figure 1*** (sites A, B, and C). Sufficient capacity to process incoming requests is provided as long as two out of the three locations are operational. Whereas on a typical day the infrastructure handles over 40,000 hits per minute, the rate can be much higher during special events.

Each location is served by multiple Internet Service Providers (ISPs). The traffic is distributed across the locations by using a combination of the Border Gateway Protocol (BGP)[2,3] and Global Server Load Balancing (GSLB).[4] If one location becomes unavailable due to either a disaster or scheduled maintenance, the BGP routes to that location are "de-advertised," and traffic flows to the other two locations without interruption of service for users.

Given that at least two sites are needed for availability, a three-site configuration is probably best. A numerical example makes this clear. Suppose 30 servers are needed to handle the transaction-processing load. For a configuration with two sites, a total of 60 servers are needed (30 in each location). For a three-site configuration, however, only 45 servers are needed (15 in each

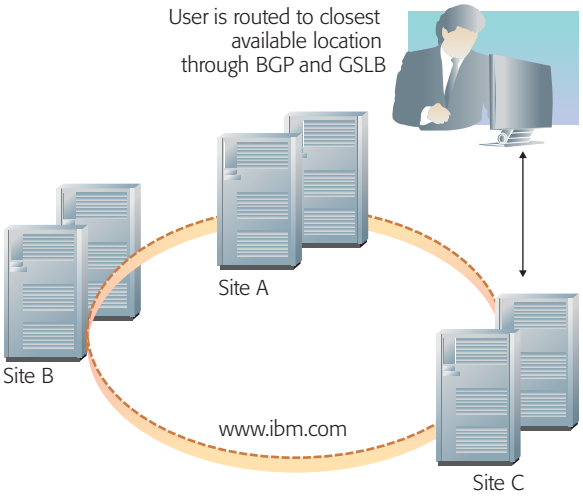


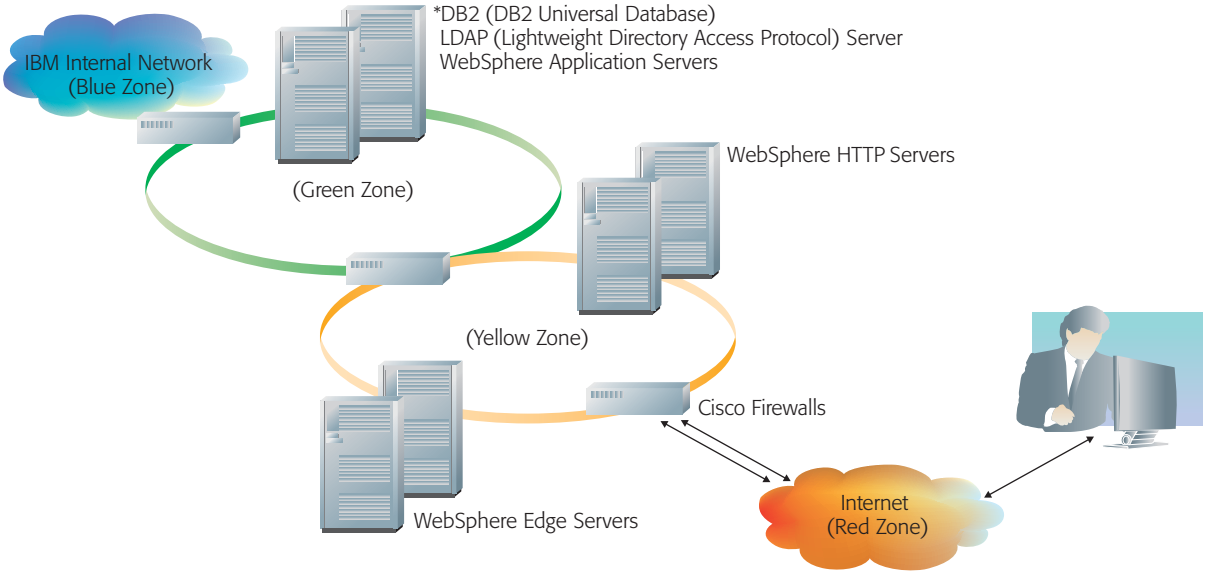
Figure 1
IBM portal sites

location)—if any one location is down, there will be the required 30 servers to process the load. A more accurate estimate of the cost should take into account other factors, such as the cost of the infrastructure for each site (firewalls, load balancers, storage servers, etc.).[5]

The configuration of one of the portal sites is illustrated in ***Figure 2***. To enforce security procedures, the network is logically divided into four separate zones. From any given zone, requests can be routed only to adjacent zones. For example, it is possible to route requests from the red zone to the yellow zone, but not from the red zone to the green zone. The Web servers, IBM HTTP servers (IHS),[6] are hosted mostly on IBM pSeries*[7] servers running AIX*[8]; whereas, various administrative functions, such as monitoring, run on IBM xSeries*[9] servers with Linux.** The incoming requests are distributed among the servers by WebSphere* Edge Servers.[10]

The IBM portal has operated at 100 percent availability for over three years. Although at times parts of the infrastructure have been taken down for maintenance and upgrades, there has not been a customer-impacting outage since June of 2001, when a 5-minute outage was caused by human error.

IBM supports the open-source model, which is based on open standards, shared source code, and collaborative development that takes place in public view. Whereas the IBM portal initially relied mostly on proprietary software, it now has a strong open-source component. Our participation in these projects is at the level of users and occasional contributors (e.g., we reported problems with the default character set on redirect headers and the



*DB2 is a registered trademark of International Business Machines Corporation

Figure 2
Configuration of a portal site

need for a configuration option to allow the use of specific encoded character in URLs—uniform resource locators).

This paper describes the use of open-source software by the portal Webmaster team and focuses on four major open-source projects: Apache** Web server, Perl, XML (Extensible Markup Language), and Struts.

APACHE WEB SERVER

When the IBM corporate Web site became operational in 1994, it ran on the NCSA[11] Web server software. This software was distributed as open source, and having access to the source code allowed us to customize it for our special needs.

The NCSA Web server software was used until late 1998 when IBM's own Web server product Domino Go Webserver took its place. IBM began collaborating with the Apache[12] project, an offspring of the NCSA Web server. As a result, Domino Go Webserver was phased out, and by 2000, we migrated to the IBM version of the Apache Web server, the IBM HTTP Server.

Use of the Apache Web server opens up access to the large community of system administrators and developers familiar with this product. Help is thus available for problems involving configuring and running Apache servers and also for software development and problem resolution.

The value of having access to the source code was illustrated to us in the following incident: when we migrated to IHS, we discovered that the Apache server was rejecting the URL-encoded slash character ('%2f') in the URI (Uniform Resource Identifier) in a manner that was inconsistent with the HTTP (Hypertext Transfer Protocol) specification, thus creating a problem with one of our applications. Access to the source code allowed our developers to design a temporary fix for the problem until a permanent solution was distributed with a new release of the product. Had the product been proprietary, it would have taken much longer to come up with a workable solution.

On another occasion, we discovered that when issuing a redirect with HTTP code 302, Apache specified its default content-encoding; if the page that is the target of the redirect is in a character set which is different from this specified encoding, it may appear broken to the user. This problem was resolved in Version 2.0.40, by the addition of a configurable environment variable.

In both these cases, we may not have been the only users—or even the first—to report the problems; but the open-source nature of the product ensured that we were able to quickly report the problems, have them validated by other users and experts, and finally have them resolved by the open-source community.

The Apache Web server is designed in a way that allows others to enhance it by writing their own extensions or "modules." This means that we are able not only to write our own extensions, but also to benefit from third-party modules. Among the several extensions to the IHS server that we wrote, there is an IHS module that generates the portal home page and displays customized content based on a combination of browser settings, user preferences, and application logic.

PERL

Perl 1.0 was released to the usenet newsgroup alt.comp.sources by Larry Wall in 1987. We have used Perl as a scripting language since the IBM Web site was first set up. At first we used Perl for handling simple HTML (Hypertext Markup Language) forms. However, by early 1995, we started looking for tools to support the creation of content and other ways of enhancing the Web site.

We realized that Perl, as an interpreted rather than a compiled language, has the advantages that its source code is easy to inspect, modify, and roll out on Web servers, and the language is well suited for quick delivery of small applications. The popularity of the language in the Web developer community provided the opportunity to share code snippets, to learn from others' experiences, and to have access to the fairly large library of Perl modules and Perl libraries that were available on the Internet.[13]

In late 1995 we wanted to create a tool that would allow Web site owners throughout IBM to implement the IBM standards for Web publishing that had been issued earlier that year. These standards mandated the use on all IBM Web pages of a masthead with a specified look and with a specified font for the text embedded in it. Because many Web site owners did not have the necessary graphical

skills to create the masthead images themselves, we used the open-source gd C libraries[14] and the GD[15] Perl modules to build a tool that can generate a masthead graphic in the necessary size and form with text specified by the user. One problem still remained: the font support was rather limited and did not include the special font required for the IBM masthead. The access to the source code of these libraries allowed us to implement an extension that provided support for the font files required.

As the number of Web sites inside IBM grew, internal standards for content were put in place to ensure that IBM Web sites displayed a consistent and professional look. Tools were needed to help Web site owners follow these guidelines, and Perl was used to create link checkers and HTML template checkers (this important suite of tools became known within IBM as the "No-Excuses Toolkit," a tongue-in-check reference to the excuses people usually offered for not following the guidelines).

In 1995 Perl's strong string manipulation features were leveraged to create the first, albeit primitive, content management system (CMS) for the portal. Content editors used the tool to submit information to a database from which Web pages were dynamically created in response to a client request. This information, which included items such as title of article, URL, and scheduled publication date on the home page, was used to control the users' view of the home page. Each request for access to the IBM home page was processed by an application that used the client particulars (browser type, domain, etc.) and the information stored in the database to create the HTML for the client. The client information was used to select a browser-specific HTML template, to which the currently scheduled articles (determined from the database of articles) were added and a customized version of the page returned to the user. The Perl application, which involves mainly manipulation of text content, performed adequately.

Perl was also used for developing more complex functions such as Web-site monitoring. We developed tools that processed Web server log files in near real time (say, every minute). These tracked the number of requests processed for pages and images, the number of application invocations, the number of server errors recorded, and the Web server CPU and memory utilizations, and presented all this information in a single coherent report to be used by the Webmasters.

XML

We undertook two major XML-related activities: adopt XHTML**[16] (Extensible HTML) as the standard for all IBM Web pages and manage portal content in XML using a state-of-the-art CMS.

Using XHTML as a standard for all IBM Web pages means that all Web pages that we deliver, whether static or dynamically generated, are "well-formed"; that is, they conform to a document type definition (DTD). With this practice in place, it became much easier to validate the syntactic correctness of documents simply by running them through a validator such as the Xerces parser.[17]

The second effort we undertook was to adopt an approach for managing the portal content that is based on XML fragments managed via an internally developed CMS known as Franklin.[18] This tool, which is based on the open WebDAV standard,[19] enables content editors around the world to create and maintain the content for their respective country portals. Content is, for the most part, not translated but rather created and maintained within a consistent encompassing framework by country editors. There are some exceptions to this practice, such as the Privacy and the "Terms of Use" statements, which are tightly controlled at the corporate level. An element of a news story, an XML fragment, can be included in several Web pages, thus allowing for significant content reuse. The headline of a news story, for example, can be used on the Web page dedicated to this news item. In addition, it can be used as an item in the list of daily news items (or on the news archive listing page, if older), and it can also be used on the IBM home page. Each of those Web pages has its own stylesheet, written in Extensible Stylesheet Language (XSL), and the (open source) Xalan[20] processor is used to transform the XML (through a process referred to as an XSL transformation or XSLT) to create the final Web page with the appropriate look and feel.

The DTD that is defined by the Franklin CMS design for any given content document or fragment type also defines the data-entry user interface for that sort of content. This approach is different from most content management systems, which use a propri-

etary language to define the data-entry user interface.

The portal content is used not only on the Web site. The XML for the news stories, for example, is translated to other formats such as Wireless Markup Language (WML),[21] Handheld Device Markup Language (HDML),[22] and cHTML,[23] for use on our wireless Web site, which supports Web-enabled mobile phones and Personal Digital Assistants (PDAs). The same XML source is also translated into the popular XML news syndication format Really Simple Syndication (RSS),[24] and can be found at news syndication indices such as Syndic8.[25]

By separating content and data from presentation information, we could more easily also extend the reach of various information fragments across multiple countries through an approach call Extended Reach.[26] This approach allows the sharing of content (fragments or whole documents) across multiple country portals, with certain content variables, such as country names, URIs, and meta-tag information, being resolved through XSLT.

STRUTS

Struts[27] is a framework for writing servlet-based applications in Java that support the Model-View-Controller (MVC) design pattern.[28] Struts was developed within the Jakarta project of the Apache Software Foundation and became popular after Jakarta Release 1.0 in 2001.

The first Java applications for ibm.com were based on the Java servlet technology and were fairly small and simple. Typically they did not interact with other IBM applications, and their functionality was similar to simple CGI (Common Gateway Interface) programs. Later we began to develop and use a homegrown Web application framework for developing more complex Web applications. We soon realized that developing and maintaining the many components involved in a Web application framework, such as design patterns, validation, and logging, is a daunting task. We also began to work with portlets for the IBM WebSphere* Portal Server (WPS),[29] in which portlets are individual applications that can be grouped as logical components on a Web page.

When Struts became available, we had compelling reasons to make use of it. First, it provided us with a common Web application framework for our application development, with the added benefit that Struts servlet-based Web applications could also be easily integrated into our existing portlet-based framework. Keeping the Struts Web application development flexible and maintaining the ability to convert our Web applications into portlet-based Web applications was a significant advantage from a strategic perspective. Because the WPS portal framework provides good integration points for Struts Web applications and adheres to the portlet open standard,[30] we can easily modify the Struts Web application and thus minimize our development effort.

Second, Struts follows the MVC design pattern of J2EE**[31] and fits well with our application architecture. Struts is especially useful in the development of the controller tier and in making connections with the view and model tiers. As a development team in the corporate portal environment, our focus is on Web server availability, security, performance, Web content, and Web standards. The fairly complex business components at the model tier, such as the processing of customer orders, are handled by back-end systems. Struts proved to be a very flexible tool in this environment.

Third, Struts technology has shown itself to be quite mature in terms of integration with existing tools, community support, and code stability. We have used Struts together with other integrated components, such as the JUnit[32] testing framework. IBM WebSphere Application Studio[33] equipped with Eclipse[34] has become our favorite IDE (integrated development environment) tool.

When using Struts, the biggest challenge for a developer is the lack of on-call support. However, books, best-practice articles, and user groups more than make up for this lack of formal support, and have proved to be very helpful resources.

Struts uses many common libraries from the Apache Jakarta project. One of these, Digester,[35] has proved to be a very useful and reusable component, which we used within Web applications for converting XML documents into Java objects. We have also used stxx[36] for generating Portable Document Format (PDF) documents.

CONCLUSION

As the use of open-source software has grown in popularity in recent years, our use of open-source software has also grown. The availability of the vast body of knowledge represented by the open-source community has made it easier to build more reliable and maintainable systems, while cutting costs and shortening development cycles.

The two key principles of open-source software are the sharing of source code and the sharing of knowledge. Embracing the open-source paradigm does raise some challenges though. Our experience shows that selecting an open-source software component is not an easy task. Typically there is a large pool of open-source software components that have to be evaluated. If the component does not have a large user base, then the development team has to allocate time to study the source code and to determine its applicability to the problem at hand. The complexity of the evaluation process is generally proportional to the complexity of the solution to be built. Another challenge is to determine how well the selected component integrates with the other parts of the system. Other factors to be considered are: the industry standards it adheres to, the application API, and the quantity of documentation.

That our portal has operated at 100 percent availability for over three years is a testament to the robustness of the Apache Web server and the applications developed using open-source software. Yet the use of open source to build individual components does not in itself ensure the robustness of an entire system. The proper software engineering discipline must still be followed if high quality and availability are to be achieved.

Application development with open-source software is normally associated with shorter development cycles and lower development and maintenance costs. However, there can be some costs in using open-source software that are not obvious. In case of a defect, for example, solving the problem requires a different approach from that with commercial products. Sometimes it is possible to purchase support for open-source products from one or more vendors, but that is not always the case. When such support is not available, there can be additional costs in tracking down defects.

The rapid growth in the use of open-source software does not mean that the use of commercial products will stop any time soon, especially in business-critical areas. However, the open-source phenomenon and the related open standards have already affected the use of commercial products. Although our use of the IBM middleware products will not change significantly because of the requirements for dedicated product support and professional maintenance, the benefits that accrue from flexible interfaces and interoperability with open-source products will become a larger factor when selecting commercial software products.

In addition to traditional application development knowledge and skills, developers nowadays also need skills to effectively use open-source software. Such skills include the ability to participate in the open-source community activities such as news groups and discussion forums. Developers are more productive when they learn to embrace the open-source culture.

We have integrated open-source software into the IBM corporate portal, including the high-performance Apache Web server, content management software, and components for various administration tasks. We continue to follow open-source trends and to make use of the latest innovations in order to address our special business goals. Our experience has demonstrated that the use of open-source software can provide significant benefits in enterprise environments.

CITED REFERENCES AND NOTE

1. H. D'Antoni, "Open-Source Software Use Joins The Mix," Information Week (Nov. 1, 2004). http://www.informationweek.com/story/showArticle.jhtml?articleID=51201599.
2. Y. Rekhter and T. Li, *A Border Gateway Protocol 4 BGP-4*, Request for Comments (RFC) 1771, Network Working Group, Internet Engineering Task Force (March 1995).
3. I. van Beijnum, *BGP*, O'Reilly Media, Sebastopol, CA (Sep. 2002).
4. A. Barbir, B. Cain, R. Nair, and O. Spatscheck, *Known Content Network Request-Routing Mechanisms*, Request

for Comments (RFC) 3568, Network Working Group, Internet Engineering Task Force (July 2003).

5. Total cost is $\delta\ (\alpha\beta/(\beta - 1) + \gamma\beta)$ where α is the total number of Web serving servers needed to meet capacity (without redundancy), β is the number of hosting locations ($\beta > 1$), γ is the fixed number of additional servers per location (firewalls, load balancers, storage servers, etc.), and δ is the cost of a server.
6. *IBM HTTP Server*, IBM Corporation, http://www.ibm.com/software/webservers/httpservers/.
7. *IBM pSeries 630: Virtual Tour*, IBM Corporation, http://www.ibm.com/servers/eserver/pseries/hardware/tour/630_text.html.
8. *IBM AIX 5L: UNIX Operating System*, IBM Corporation, http://www.ibm.com/servers/aix/.
9. *IBM eServer xSeries*, IBM Corporation, http://www.ibm.com/servers/eserver/xseries/.
10. *WebSphere Edge Server*, IBM Corporation, http://www.ibm.com/software/webservers/edgeserver/.
11. *NCSA HTTPd homepage*, NCSA HTTPd development team, http://hoohoo.ncsa.uiuc.edu/.
12. *The Apache Web Server*, Apache Software Foundation, http://www.apache.org/.
13. *CPAN*, Comprehensive Perl Archive Network, http://www.cpan.org/.
14. *gd library homepage*, Boutell.Com, Inc., http://www.boutell.com/gd/.
15. *GD Perl module homepage*, Cold Spring Harbor Laboratory, http://stein.cshl.org/WWW/software/GD/.
16. S. Pemberton, et al., *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)*, W3C Recommendation 26 January 2000, World Wide Web Consortium (W3C) (August 2002).
17. *Xerces: XML parsers*, Apache Software Foundation, http://xml.apache.org/#xerces.
18. L. Weitzman, S. Elo-Dean, D. Meliksetian, K. Gupta, N. Zhou, and J. Wu, "Transforming the Content Management Process at ibm.com," *Proceedings of CHI2002/AIGA Experience Design Forum* (April 2002), pp. 1–15.
19. *Web-Based Distributed Authoring and Versioning*, IETF WebDAV Working Group, Internet Engineering Task Force, http://www.webdav.org/.
20. *Xalan: XSL stylesheet processors*, Apache Software Foundation, http://xml.apache.org/#xalan.
21. *Wireless Markup Language*, Open Mobile Alliance, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html.
22. *Handheld Device Markup Language*, World Wide Web Consortium (W3C), http://www.w3.org/TR/NOTE-Submission-HDML-spec.html.
23. *Compact HTML Submission Request to W3C*, World Wide Web Consortium (W3C), http://www.w3.org/Submission/1998/04/.
24. *RDF Site Summary*, web.resource.org, http://web.resource.org/rss/1.0/spec/.
25. *Syndic8 homepage*, Syndic8.com, http://www.syndic8.com/.
26. S. Monheit, S. Elo-Dean, D. Leip, and H. Shirayama, *Extended Reach: An Efficient Content Management Technique for Sharing and Localizing Content*, IBM Technical Report TR-40.0032, IBM Corporation, http://www.ibm.com/webmaster/papers/TR-40-0032.pdf.
27. *The Apache Struts Project*, Apache Software Foundation, http://struts.apache.org/.
28. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Pattern: Element of Reusable Objected-Oriented Software*, Addison-Wesley, Boston, MA (1994).
29. *WebSphere Portal for Multiplatforms*, IBM Corporation, http://www.ibm.com/software/genservers/portal/.
30. *JSR 168: Portlet Specification*, Sun Microsystems, Inc., http://www.jcp.org/en/jsr/detail?id = 168.
31. *Java 2 Platform Enterprise Edition (J2EE)*, Sun Microsystems, Inc., http://java.sun.com/j2ee/index.jsp.
32. *Project: JUnit: Summary*, Open Source Technology Group, http://sourceforge.net/projects/junit/.
33. K. Brown et al, "*Enterprise Java Programming with IBM WebSphere*," *Addison-Wesley Professional, Boston, MA* (November 2003).
34. Eclipse.org, Eclipse Foundation, http://www.eclipse.org/.
35. *Commons Digester*, Apache Software Foundation, http://jakarta.apache.org/commons/digester/.
36. Struts for Transforming XML with XSL (stxx), Open Source Technology Group, http://stxx.sourceforge.net/.

Accepted for publication December 2, 2004.
Published online April 26, 2005.

Ning Yan
IBM Corporation, 19 Skyline Drive, Hawthorne, NY 10532, USA (nyan@us.ibm.com). Mr. Yan has been with the ibm.com Webmaster team since 2002 and is a software engineer with expertise in the development of Web applications using open-source software, DB2, and WebSphere. Before joining the corporate Webmaster team, he worked on the WebSphere Commerce Server. He is an IBM-certified DB2 specialist and a Brainbench-certified Web service engineer and Web developer. He received an M.S. degree in computer science in 1996 from State University of New York (SUNY) at Albany, New York.

David Leip
IBM Corporation, 19 Skyline Drive, Hawthorne, NY 10532, USA (leip@us.ibm.com). Mr. Leip is a Senior Technical Staff Member and the IBM corporate Webmaster and has managed the corporate Webmaster team since 1999. Since joining IBM in 1992, he has been involved in Internet-related activities and, in particular, in directing the IBM Web presence since its inception. He is on the advisory board of the Center for Agile Development at Pace University and on the advisory board for the World Organization of Webmasters. He has an M.Sc. degree in computer science from the University of Guelph and 18 technical certifications.

Kapil Gupta
IBM Corporation, 19 Skyline Drive, Hawthorne, NY 10532, USA (kapil@us.ibm.com). Mr. Gupta is a Senior Software Engineer on the ibm.com Webmaster team. His area of expertise is running large Web sites, and he has been involved in developing solutions for the Web since 1994. He has used open-source software since his days in graduate school in the early nineties and continues to be a strong believer in the benefits of the open-source movement. He has an M.S. degree in computer science from the University of Houston (1996) and an M.B.A. degree from the New York University Stern School of Business (2004). ■