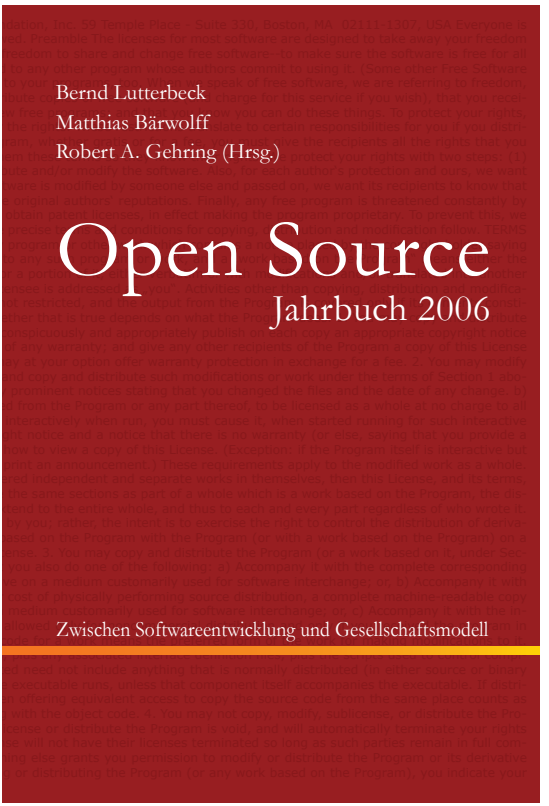
This article has originally been published in German as part of the



available at www.opensourcejahrbuch.de.

The *Open Source Jahrbuch 2006* is an extensive compendium dealing with the various aspects of open source software and beyond. Whilst most articles have been written in German, this is one of the articles that have originally been written in English and subsequently been translated into German. Refer to our website for more English articles as well as our translation wiki.

A Look Inside Microsoft's Linux/Open Source Software Lab

BILL HILF*

In the heart of what may be the world's biggest Microsoft software environment we have built an open source software lab with the aim of understanding open source software and to help with interoperability between our software and that of the open source community. To this end we have employed a number of open source experts and developers and created an impressive environment of Linux and Unix machines. Microsoft is trying to understand the nature of open source, and the structures and processes behind it.

Keywords: Linux-/Open-Source-Software-Labor · Microsoft · Cooperation · Interoperability · Competition

1 Introduction

To say that the Linux/Open Source Software Lab at Microsoft is anything shy of an incredibly ambitious research effort is an understatement. The lab houses more than 300 servers of all types and sizes that collectively run over 15 versions of Unix and more than 48 different Linux distributions. It boasts a research team of senior-level Linux and Unix programmers and system administrators, some of whom were the chief architects of popular Linux distributions or authors of well-regarded books on Unix. In short, it is one of the few labs in the world running this much equipment with so much complexity and such a high level of expertise, all in the name of open source research.

When I talk about the Linux/Open Source Software Lab and my involvement as its director, the usual response is, "At Microsoft? Why manage Linux in a mixed

* Bill Hilf is the *Platform Technology Strategy Director* of the technology department *Linux and Open Source* at Microsoft. There he is responsible for the Shared Source initiative since December 2005.

environment at Microsoft?" While theories abound—ranging from "Microsoft is working on its own Linux implementation," to "Microsoft is considering porting to Linux"—the truth is, the Linux/Open Source Software Lab provides Microsoft with a deep line of sight into the world of open source software and helps improve the way Microsoft products work with Linux and a host of other open source applications.

Contrary to a common assumption that Microsoft is anti-open source, the reality is not so black and white. Certainly, most customers don't live in that either/or world. They choose a technology—an operating system or an application—based on its ability to solve a particular problem and to serve a certain business need, not based on its development model. By running Linux and a variety of other open source software (OSS) in a Microsoft environment, we are learning how those technologies can better interoperate with Microsoft's technologies, so that Microsoft customers can benefit from a wider variety of interoperable software.

For instance, one of the issues we've worked hard to address is how Microsoft management tools can do a better job in heterogeneous environments. So if a customer is using *Microsoft Systems Management Server (SMS)* or *Microsoft Operations Management (MOM)* tools and wants to be able to manage a Linux or Unix server, we can provide input to our customers and to the Microsoft product teams based on our testing of a variety of third party technologies that can be used to enable this type of scenario.

Another example is the work the lab has done with the "Release Candidate 2" (R2) version of Microsoft Windows Server 2003. It incorporates a variety of technologies, collectively called the Subsystem for UNIX-based Applications that provide services for interoperability with Unix and Linux systems. These technologies include Unix network services like *Network File Sharing (NFS)* and *Network Information Service (NIS)*. We've done extensive testing to see how well Windows Server 2003 R2 actually interoperates with other UNIX and Linux systems. For instance, we've tested different open-source applications, *NFS* and *NIS* running in that subsystem to see how interoperable those applications and services are with other elements in the data-center environment.

2 Practicing the Art of Coopetition

While testing interoperability between open source software and Microsoft products is one of the lab's main objectives, it isn't the only one. Another important objective is a more competitive one—to help Microsoft build better products by deeply understanding Linux and open source. We analyze, test and benchmark aspects of open source software we want to compare to Microsoft products, such as various server workloads, desktop scenarios, virtualization technologies, security technologies, management tools or just applications that are specific to certain vertical industries. We use this analysis and feedback with the product teams so that they can address the

results as they plan and build their products.

A recent example of this type of research is our testing of a beta 2 version of the *Microsoft Windows Compute Cluster Server 2003*, which Microsoft just announced last month as part of the company's entrance into the high performance computing (HPC) market. That market today is largely dominated by Linux. When the product team first began building this product, they asked the Linux/Open Source Software Lab to determine what the best possible HPC solution is from an open source perspective.

Because our staff at the lab has a variety of expertise in HPC, we were able to build a large clustered system and conduct extensive application testing. We did benchmarks on the Linux side, and then we wiped out that installation and did the same testing on the *Windows Compute Cluster Server (CCS)* using the same hardware and network setup. We shared the results with the product team, so that they can understand the strengths and weaknesses of the various Linux HPC solutions, which will help them make *Windows CCS* a more compelling product to customers when it's released next year.

This art of "coopetition", of competing with open source and at the same time pursing interoperability as a common goal, is something that Microsoft knows well. Both Microsoft and OSS technologies will continue to be around for years to come, it's important that Microsoft work toward both these goals simultaneously. As such, the central objectives of the Linux/Open Source Software Lab reflect that larger goal.

3 A Piece of Fiber and a Hole in the Wall

Of course, there are different ways one could gain this kind of knowledge aboutopen source software. It might be easier to rely on third-party interpretive data, but in many ways that approach would be a lot like trying to understand a foreign country without ever spending much time there. Buying Berlitz language CDs or a tourist travel guides might make you feel like you're a part of the culture when you visit, but you'll remain a tourist unless you actually live there for awhile.

Microsoft has embraced this philosophy in building the open source lab. Rather than function as a third-party that's trying to look in from the outside and understand open source, the company wants instead to be deep in this space and have experts who can provide fact-based, unbiased scientific information. It has directed the lab to find the science that proves statements made about Linux, so that the company doesn't philosophize or guess what's going on from the outside. By being a center of competency around open source software for the rest of Microsoft, we at the lab can provide hard data and conclusions from open source experts to Microsoft product teams when they ask questions such as, "What's the state of management tools in Linux?" or "What's the state of the Linux desktop?"

When Microsoft hired me in 2003 to build the Linux/Open Source Software Lab

and lead a team of Linux and Unix researchers so that the company could better understand open source software, I had no idea how literal they were about my "building the lab."

Because the IT department at Microsoft runs all-Microsoft software, I was charged with building the lab so that it would reflect a true Linux/OSS environment. What that meant was the company had an existing room for the lab, and that was all. During the first days on the job, I stood in a very empty and cavernous room while some IT guys on the floor overhead threaded a network cable through a freshly poked hole in the ceiling. I was still standing there, staring at the piece of fiber in my hand, when the IT guys came downstairs and said, "That's it, that's all we can give you. You're kind of on your own from here." Except for the walls, ceiling and that little piece of cable, we literally had to build the lab from the ground up.

The first phase, of course, was to hire the staff, the most important asset the lab would have. We now have a mix of fulltime employees and contractors, all of whom have been expert developers or systems administrators in the open source community. Some of the team members, such as Daniel Robbins, the founder of *Gentoo Linux* who joined us earlier this year, have been chief architects or leads of Linux distributions. Others have deep Unix expertise and are authors of popular Unix books or tools. Some are Linux/OSS security experts, embedded developers, virtualization and clustering experts, or developers with strong backgrounds in *GTK+*, *GNOME/KDE*, and Localization. Additionally, we have members of the team familiar with running Microsoft products in large data center environments, such as MSN. The breadth of the expertise on the team is what's most impressive, and nearly all of them have experience working in large, highly mixed IT environments.

The second phase involved building out the lab. In less than two years, we put together a wide array of different technologies that use an enormous variety of hardware, software and applications. With more than 300 servers from vendors as wide-ranging as Dell, Hewlett-Packard, IBM, Microtel, Penquin, Pogo, and Sun, and more than 20 versions of Unix and 48 versions of Linux, which include lesser known distributions like *Asianux*, *CentOS* and *NetBSD*, the Linux/Open Source lab has to be one of the most unique research facilities devoted to open source discovery.

Because we run dozens of different versions of everything, we have the opportunity to test open-source interoperability capability in a multitude of scenarios. And because everything we do in the lab we do on our own, from running our own network and security services to doing our own patching and updating, our environment mimics real customers' environments. I frequently tell the Microsoft product teams that if their product makes it through this lab, it probably will survive 90 percent of the Linux/Unix/OSS oriented customer environments out there.

Operating System	Version/Distribution
Windows	Windows 2000 Server, Windows Server 2003 Enterprise, Windows Vista, Windows XP
Linux	Arch Linux, Ark Linux, Asianux, Crux Linux, Debian, Fedora Core, Foresight Linux, Freedows, Linux From Scratch, Gentoo, Libranet, Mandrake Linux, Mandriva, MEPIS, Novell Open Enterprise Server, Red Hat Enterprise Linux, Red Hat Linux, Rocks, Slackware, SuSE Linux Enterprise Server, SuSE Linux Standard Server, SuSE Pro, Tinysofa, TurboLinux, Vector Linux, Vida Linux, Ubuntu
Unix	AIX5L, FreeBSD, OpenBSD, NetBSD, Solaris, Java Desktop System
Others	MacOS

Table 1: Installed operating systems at the Open-Source-Software-Lab

Vendor	Hardware
Compaq	Proliant DL580, Proliant BL10e, nx5000
Dell	PowerEdge 2450, PowerEdge 4350, PowerEdge 1855, PowerEdge 1500SC, Optiplex GX280, Optiplex GX270, PowerEdge 1855, PowerVault 745N
HP	Proliant DL380, Proliant DL585
HP Compaq	nx5000, D530
IBM	xSeries x342, xSeries x340, xSeries x330, xSeries x350, pSeries 630
Microtel	Computer System SYSMAR715
Neoware	CA5
Pogo	PW 1464, PW 1180, Vorticon64
Sun	SunFire V240, SunFire V20Z, SunFire 280R
Toshiba	Tecra M2, Protégé

Table 2: Hardware used at the Open-Source-Software-Lab (without customized products)

4 An Open Source Bubble Swimming in a Sea of Microsoft

One of the more interesting and unexpected dynamics at the lab is that this very large Linux and Unix shop is surrounded by the world's largest all-Microsoft environment, which supports all of Microsoft's employees. This includes the company's Windows–based security services, Internet proxies, mail services, human resource services, and all the other systems that run the company.

So we've had to figure out ways that we can interoperate, not just within the lab, with its incredibly complex environment, and not just with some aspect of the Microsoft environment, such as *Exchange* or *Active Directory*, but between the lab and the rest of Microsoft.

So many customers running mixed environments have asked how we manage Linux, UNIX and other open source software (OSS) inside such a Microsoft-centric IT environment. "How do you get these things to work together? How do you do deployment of software? What kind of tools do you use?"

The breadth of management tools we use parallels the number of servers, operating systems and other applications we have running in the lab. For software management and distribution, we use tools like *Microsoft Systems Management Server* with *Vintela Management Extensions (VMX)*, *Kickstart*, *Red Carpet*, *Portage* and *Red Hat Network*. To remotely manage the lab infrastructure, we use *SSH*, *VNC*, *X-Windows Tunneling* and *Windows Terminal Services*. Again, it is unlikely any single customer would use all of these tools. But we try to mimic a variety of scenarios that a multitude of customers might be using so that we can deeply understand and hopefully play a role in remedying the issues that are out there.

Having all of these different Linux-oriented workloads, servers, desktops, laptops, software and device configurations inside such a huge Microsoft environment has allowed us to experiment with and test interoperability on a daily basis. In the process, we're learning some very interesting things. Some of them are simple issues, like how to get on the Internet running Linux in a Windows environment, and others are more complex, like how to authenticate against *Active Directory* from Linux clients or how to run OSS mail clients with *Microsoft Exchange Server*, and so on.

5 Building and Testing Interoperability at the Lab

One of the more interesting areas of discovery that came about while testing interoperability in the management tools we use, focused on extending *Microsoft Systems Management Server (SMS)* so that it can be used to manage Unix, Linux and even Apple systems. *SMS* was built so that it can use an open protocol called *OpenWBEM* to communicate with other pieces of software, such as *VMX*, that run on non-Microsoft systems. By extending *SMS* using *VMX* in the lab, we're able to use the *SMS framework* to manage all of our servers and clients. This type of setup provides those in

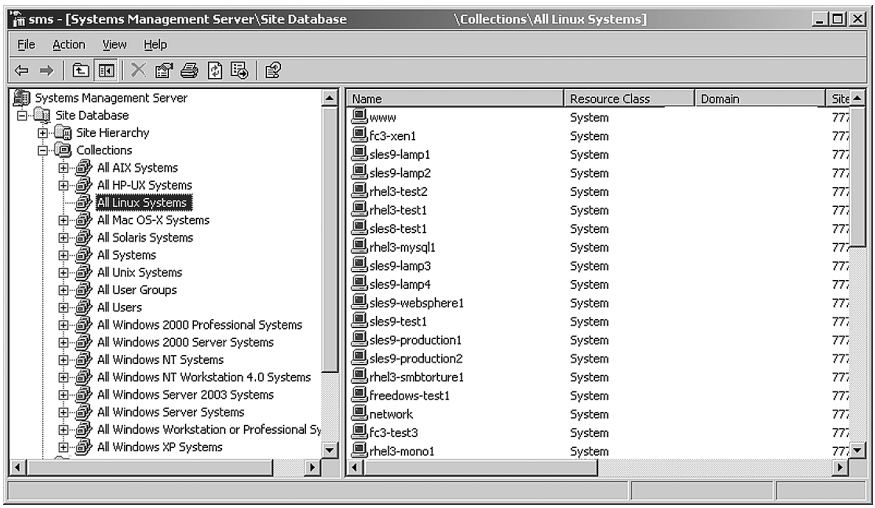


Figure 1: SMS-Interface to manage the Linux server at the Microsoft-Open-Source-Lab

the lab used to working with Windows a familiar tool for managing non-Windows systems, while at the same time providing our Linux and Unix researchers with the management tools they're familiar with, such as *SSH clients*, *X-Windows* and *Red Hat Config tools*. Figure 1 is a picture of the *SMS* interface for managing our Linux servers in the lab.

Another valuable lesson we've learned about using commercial third-party software to extend Microsoft products involved using *Centrify DirectControl* solution to integrate our Unix and Linux platforms with *Microsoft Active Directory's* identity, access and policy management services.

Before we started using *DirectControl*, if one of us wanted to log in to any of our servers, we'd have to authenticate a username and password for each of those servers—a daunting task when you have over 300 servers with a large mix of operating systems in your environment. But by setting up an *Active Directory* domain so that each of us has one domain username and password, we can access any type and number of servers to which we are authenticated after logging on once to that domain. This gives us a really powerful single authentication solution. Being able to authenticate from a Linux server to an *Active Directory* server is something that many people are not aware of—I've given presentations to CIOs who were unaware of these types of solutions.

The Linux/Open Source Lab also played a role in helping Microsoft's test support for Linux into *Microsoft Virtual Server 2005 SP1*, which can virtualize both Linux and Sun Solaris operating systems on servers running Windows. We tested this support

extensively using *Virtual Server 2005* on a single machine to run all 48 of our Linux distributions as guest operating systems. This setup allowed us to test drive different Linux distributions and have all sorts of different servers running on a Microsoft operating system without having to use separate servers for each Linux machine.

6 Helping Unix/Linux Pros Speak a Familiar Language Using Windows

Having product interoperability is important, but a larger issue that many Unix and Linux professionals have with Windows is familiarity.

As a long time Unix professional, one of the biggest hurdles for me in a Windows data center was the lack of crossover between my language, which comprised my Unix skills and knowledge, and the language of the Windows platform. There are a variety of ways to approach this incongruity, such as using the Subsystem for Unix Applications in R2 or by using a third-party product like *CygWin*, or even using a virtualization product. But many of these are migration tools that really are more like stepping stones for moving an application from one platform to another. What I really needed was an integrated, technology that was part of Windows that could help me, a long-time Unix professional, speak a familiar language on a Windows server.

Creating this type of technology is precisely what one of our Windows Server management product teams is currently doing. Called *Monad*, it is an extraordinarily powerful next-generation command line and shell environment for Windows that runs on .NET. Microsoft's vision for *Monad* is to enable simple automation for local and remote administration by delivering a consistent, fast, comprehensive and composable command line scripting environment that spans the Windows platform. For a Unix pro, the ability to compose my command line and scripting environment in a way that looks and behaves similar to what I've learned in a Unix/Linux world, gives me much more control over my Windows environment and significantly reduces the barrier between moving between Unix and Windows.

One of the things the Linux/Open Source Lab is doing with *Monad* involves writing plug-ins using a popular open source tool called *VIM (VI Improved)* to call on *Monad*. One of our researchers used *VIM*, which edits text files and code, to write a plug-in that calls on *Monad* to digitally sign a script when the script is ready to be deployed. The advantage of using a plug-in like this is that it allows *VIM* users (a large population in the Linux/Unix world) the ability to extend their favorite tool into monad while simultaneously adding value to *Monad*. We use *Monad* extensively in the lab to for administration tasks and we are looking at a variety of other scenarios similar to using this *VIM* plug-in that can help the Unix/Linux user community feel at home in the next generation Windows command line and scripting environment.

Figure 2: Number of file changes between the original Linux Kernel (from kernel.org) and the same version of that kernel provided by the corresponding Linux distribution

7 Using Science to Test Perceptions

We also hear from customers a variety of broad assumptions around Linux, Open Source and Windows. We use our lab to analyze these perceptions and rely on science, versus ideology or belief, to test these perceptions.

For example, a common perception about Linux is that it will run on just about any piece of hardware, so we decided to test legacy hardware support for Linux to see if that perception was true.

We tested eight current Linux distributions, including *SuSE Pro 9.2*, *Xandros* and *Fedora Core 3*, along with Windows XP and Windows Server 2003, by attempting to install each of them on computers representative of the average computer available in 1995, 1997, 1999 and 2001. This was not a "Can you modify the operating system?" test; it was simply an out-of-the-box installation from a CD-ROM. If we were able to successfully install the operating systems, we then looked at their performance. What we found was that there were not many cases in which a modern Linux distribution could run on PCs that were older than what Windows could run on as well. So in this example, we were able to use science to disprove a common perception.

We also track other open source software dynamics, such as how different the commercial and non-commercial distributions of Linux are from their original open source projects. The chart in 2 illustrates some of these differences—the number of files changes between the original open source Linux kernel (from kernel.org) and the same version of that kernel as shipped by a commercial Linux distribution. This allows us to better understand the "community versus commercial" dynamics at work in the open source model.

In addition, we attempt to quantify the ecosystems so that we can understand the

real numbers underneath the market trends. For example, last August we counted the number of supported PCs and servers for *Red Hat* and *Novell SuSE* operating system products and compared those against our own supported hardware numbers.

We looked at certified systems for three versions of *Red Hat Enterprise Linux*: versions 2.1, 3 and 4, and found that these versions supported 241, 745, and 173 server systems respectively (http://bugzilla.redhat.com/hwcert/). *SuSE Linux Enterprise Server* version 8 had 490 certified systems and version 9 had 537 (http://developer.novell.com/yessearch/Search.jsp). On the desktop Red Hat Desktop supported 360 systems while Novell Linux Desktop 9 supported 165. These numbers stand in stark contrast to the number of certified systems for Microsoft Windows Server 2003, which is more than 5,000 and for Windows XP, which is over 85,600.

This type of analysis helps us understand the real size and growth of the current market-leading Linux distributions and how Windows compares. It also gives us an indication of what type of systems customers might be using to deploy with, which in turn helps us to understand the configurations of our lab.

8 Diving Deep Into the Sociological Aspect of Open Source

One of the most significant areas of open source software that the lab looks closely at is the phenomenon of community development, and it is one of the key characteristics that Microsoft is learning from the open source community.

We spend about 20 percent of our time learning about this process and helping Microsoft developers and testers also learn how they can be more aware of and ensure Microsoft products are more accessible to the development community. As engineers and technologists, they are fascinated by this model—they want to understand how testing actually happens in this collaborative community; what tools are used; how test cases are written; how bugs are filed, tracked and regressed; and what type of training testers have had. We approach the community model objectively, analyzing the good and the bad aspects of it, separating fact from fiction, so that we fully understand the engineering pros and cons of community development without the surrounding philosophy and hype.

The community development model has helped Microsoft find new ways to think about its own development projects, such as *Microsoft Shared Source*, and how it can take better advantage of the community process.[1] Educators from our lab work with the product teams to ensure they've thought through all the elements of what it's like to use this process and what types of issues are likely to surface. And just being able to engage with the open source community about how it develops software has also helped Microsoft mature its thinking around how to participate with developers who are building software using different development models.

1 Information about *Microsoft Shared Source* are available at http://www.microsoft.com/sharedsource

In turn, the lab over the past two years has become a kind of gateway for open source developers at the engineering level who need to communicate with Microsoft. For example, when they're working on an interoperability issue with a Microsoft product and have questions, they increasingly are contacting us. Here they find other developers and researchers who understand the open source community, their language and their issues. This has to be a positive development—for too long Microsoft and the open source community have been regarded as separate realities. Just the idea that there might be some bridge-building happening on this level is very encouraging, so we're progressing in this realm with real results.

9 Looking Forward: The Future of Open Source

Another aspect of research we're pursuing at the Linux/Open Source Software Lab is the historical trending of open source software. We've done deep analysis of the last major versions of the Linux kernel in an attempt to answer a number of questions about its evolution. Is the code getting simpler or increasingly more complex? Does it have more or fewer defects every year? Is it growing larger or getting smaller? Is it becoming more or less efficient? Through all this research, we've identified three common trends.

First, if you focus only on the software characteristics and the code, open source software appears to be growing fairly linearly. So every year the lines of code increase, complexity grows as a result, and, of course, with complexity comes more defects. This isn't surprising, nor is it a knock against open source software—this is how all software, commercial or non-commercial, has evolved. This growth is not to be confused with the modularity of open source software. Many people mistakenly believe that if something is modular, it doesn't grow and is always simple to maintain. This is not the case—even with modular software there's still growth.

What may be unique to open source software, however, is whether the community development model will be able to continue to adequately respond to this increasing complexity. Can this loosely coupled model, in which developers work on and distribute Linux in different areas around the world with very loosely defined authority, planning, testing, or structure, sustain the growth of the software? Or is it possible that because of the way the community development model works, the software could plateau rather than continue to grow and get more complex?

Again, this is a software engineering research question—we are not looking for a positive or negative answer. This is something we're watching closely and something we are investigating in other open source applications beyond the kernel.

The second trend we're seeing is the growth of commercial and professional open source software companies. Over time, we have tracked the developer contributions for various OSS, identifying who is working on the code. By looking at research in the repositories, academic research in this area, and our communication with the

community, the pattern has been clear: Over the past five years or so, more and more contributions are coming from developers employed by a commercial entity that either directly makes money from the OSS project (such as MySQL or JBoss) or indirectly through hardware, commercial software and services (IBM, Novell, HP). To those in the OSS developer community, it's not a huge surprise, but to those in the broader market who might still believe that people are working on Linux or other OSS projects "in their free time," it is often a surprise.

The third trend we are seeing is a market realization of the OSS model overall—what the essence of the phenomenon really is. In surveying and analyzing the large amount of open source software available, a large amount of these projects are system software that has been developed for other developers or system administrators. So I think it's fair to say that in the larger, historical perspective, open source software has largely been a developer phenomenon. And therein lies an essential difference between how commercial software companies build software and how open source works today: Commercial software companies design and engineer software to serve a customer need, whereas open source software is largely designed by and for developers and technical users.

In some domains this phenomenon is quite powerful and has enabled rapid growth in various areas. Of course, there are some exceptions, but this differentiator that is becoming more apparent in modern thinking around open source software. Moreover, it shows that a variety of development models can and will coexist in the software ecosystem—indeed, we have found that many popular OSS server applications have a large and growing business on Windows (such as JBoss and MySQL).

10 Adding Value by Providing a Balanced View of OSS Trends

By exploring the dynamics of the open source software phenomenon in an impartial and unbiased manner that relies on hard technical data, the Linux/Open Source Software Lab at Microsoft has been able to drive improvements and changes to both internal Microsoft groups and customers who have asked us to look into common Linux/OSS questions and issues. And while we're very proud of the work we've accomplished so far, by continuing to practice the fine balance between cooperation and competition with open source software, we are equally confident that our future research will benefit Microsoft, its customers and partners, and the open source community. It is an exciting time indeed!