



Open source software: Perspectives on this technology's promise

Technology Viewpoints

Volume 1. Issue 2.

CGI's Technology Viewpoints are a quarterly look into the latest trends in the ever-evolving technology industry. This edition examines the hot topic of open source software. It was developed through an "open source style," where content was shared, reviewed and refined through CGI's Technology Focus Network. The use of open source software is commonplace within CGI—used to manage our servers and to develop and enhance our solutions—and is governed by a corporate open source policy. This issue takes a look at the topic from six key viewpoints: security, emerging networks, application infrastructure, IT infrastructure, virtualization and risk mitigation.

Security

Are we more vulnerable?

Open source software provides visibility into its design. Some argue that this increases the potential of security threats, while others contend that increased exposure uncovers more security flaws and hence decreases system vulnerability.

Generally, history has shown that open source software is no less secure than its closed, proprietary counterparts. It is worth noting that many successful encryption algorithms are published, and a popular school of thought discourages the use of unpublished encryption algorithms altogether. Vulnerabilities are generally less severe on the open source side than on the proprietary side. This is true for a couple of reasons. First, open software is generally architected to be more secure in terms of how it functions. Additionally, large bugs—the low hanging fruit—get picked off quickly because of peer reviews. One should also not discount the resourcing factor: organizations that produce software have limited resources, especially when it comes to the security auditing of code. Open source theoretically provides for unlimited resources, although based upon its usage, improvements can occur at vastly different paces. The more widespread the use, the better it gets, faster. An important caveat: Open source software should always be obtained from reputable sources.

CGI's stance: Security is neither diluted nor enhanced by the inclusion of open source components in a system.

Knowledge sharing increases security

In the world of IT security, the "bad guys" are perceived to know a lot more than the "good guys." Yet according to renowned hacker Kevin Mitnick, almost every high-level security vulnerability comes as a result of broken business processes or social engineering—not technical expertise.

The application of industry best practices obtained through open and shared knowledge provides the upper hand in making systems secure. In some instances, individuals involved in open source software know a lot more about systems and security than individuals involved in strictly proprietary software. This is because they share both vulnerabilities and vulnerability testing tools. Access to and use of this

knowledge and these tools has led to more secure systems. There are multiple reasons for this, including the fact that the source code is always available as a source of learning.

In short, open and shared knowledge balances the “bad” and the “good.”

Emerging Networks

Open source implementations of industry standards

Networking has long moved from the days of proprietary designs to industry standards. The logical extension of industry standards are free and openly available implementations of those standards.

We are witnessing open source implementations that provide functionality in the realm of proprietary solutions, in such areas as PBX applications, routers, firewalls and VoIP solutions. Where once IT shops viewed these types of components as outside their scope, convergence of network and applications is changing the model. Organizations now have an open source option as they add new and incremental capabilities and capacity to their networks. Beyond e-mail, communications applications such as instant messaging, blogs and Wikis are available in open source implementations. The open source implementations of emerging technologies and standards also have been the catalyst to drive inclusion of their support by proprietary solution providers; real-life examples include SIP, RSS and OpenDocument standards.

Organizations, which are so inclined, have open source alternatives at their disposal in areas that were once dominated by proprietary solutions.

Application Infrastructure

Fee-free application infrastructure

Applications can be built without license fees.

Application development and execution environments are two related yet distinct aspects of application infrastructure. When we refer to application development, we tend to talk about interpreters/compiler, frameworks, integrated development environments and standards compliance testing tools. Execution environments for applications cover components such as applications servers, databases and rules engines. Further to development and execution environments are customizable open source applications that provide a foundation to develop applications as both an infrastructure and a template. There are an abundance of software products built for developers by developers that are open source, widely available and free.

A global, open community of developers using and evolving application development and execution software is good for the IT industry. It leads to higher productivity and lower cost.

Fee-free applications

Business packages can be implemented without license fees.

Tangible mid-market solutions are available in open source form. CRM (Compiere, SugarCRM), ERP (Compiere) and Web portal (Mambo) tools are examples. Open source business solutions can be found with well-staffed developer communities supporting them.

Open source typically does not extend to the broadest capability products; however, if the functional fit exists for an open source product, a respectable open source package is a viable choice.

Multiple options exist

At least one and typically multiple open source options exist for a variety of application infrastructure components.

Open source implementations of standardized specifications such as RFCs and JSRs, as well as non-standardized products available under a dual licensing model, allow us to defer vendor selection and spending decisions to later in the application development cycle. One can start developing an application built on open source implementations and switch to proprietary implementations or purchase a supported license if and when necessary. This can be quite late in the application development cycle.

Such open source implementations allow us to offer applications and configurations at different price points according to the capabilities they offer. For example, developing applications using object persistence technology truly decouples an application from the underlying database. Moving up the database scale is a relatively small matter of changing the database driver in a configuration file and thorough retesting.

An important aspect to pay attention to is licensing and design documentation. One needs to be clear what open source components have been used and under what licensing agreement. This is most important at the application level, where the coupling of custom and open source code is closer.

Open source application infrastructure provides multiple options for application execution.

IT Infrastructure

Fee-free end-user applications

Open source end-user applications provide a viable alternative to commercial software (COTS).

Take these scenarios: An amateur photographer is looking for a free software product to manipulate digital photographs. The open source software GIMP meets this need. A movie enthusiast is looking for a MPEG-4 codec. FFDSHow fits the bill. A podcaster needs a tool to mix audio. Audacity works like a charm. The list of open source software goes on and on and includes word processors, spreadsheets, utilities and image and video processing software. The quantity of available open source products is measured in the thousands.

Typically, quality and feature functionality meets expectations. When it doesn't, there is a level of forgiveness on the part of the user since the price is right; free keeps expectations in check. And in some instances, quality and functionality of open source software rivals that of COTS products and even exceeds that of proprietary alternatives.

How do these examples affect large organizations? The lesson is that almost any standard end-user computing functionality requirement can be met with an open source software solution.

Linux is in the enterprise

One would be hard pressed to not find Linux somewhere within a large enterprise. Linux—and is the “poster child” of open source—is proven, reliable and has measurable penetration.

There are ROI arguments for and against Linux. While the holy grail of the desktop operating system (OS) still belongs to Microsoft, this has not prevented Linux from showing up on appliances, Internet-facing servers, application and database servers, and even on the mainframe. The best measure of the acceptance of Linux as an OS is its availability as a preinstalled option from all major hardware suppliers.

Bottom line: It would be a bad decision not to consider Linux as an OS.

Fee-free management tools

Standalone tools and extensions to proprietary management tools exist in open source form and are in data centers everywhere.

Within an organization, the functionality required of management tools crosses business boundaries. The same opportunities are addressed and problems resolved, leading to the enterprise’s logical adoption of free and open source management software. Open source tools can be installed and made operational in less time than it takes to develop a project plan for a custom tool or proprietary equivalent. This makes adoption of outside software a common sense decision. Should problems arise, the component can be fixed or replaced, often with the fix already available on the Internet.

There also are open source tools for which there are no proprietary equivalents. One example is the free tool cfengine. With the proper infrastructure in place, cfengine has been used to centralize and automate the management of thousands of UNIX and Linux systems. Further, a sizable number of commercial java-based applications make use of the open source JLOGGER4 API and components.

Another point can be made for open source UNIX platforms such as Linux. Due to their fundamental architecture there are many more operations and administration management options available, most of which are available for free.

Open source management tools in the data center are a cost effective and low risk solution, but may be limited depending on the platforms deployed within an environment.

Virtualization

Distributed, virtual teams

The question used to be “Why isn’t Sam coding?” Then it was “Where is Sam coding?” Now it’s “Who coded this?”

Open source software development leverages virtualization approaches. In many open source projects, we experience teams of developers working on projects that cross traditional organizational boundaries. Common interests create communities that develop, maintain and evolve a single software entity. Software tools supporting virtual teams have evolved, as have the governance processes, repositories and forums that support its use. Successful open source development projects have provided a model for distributed, virtual teams within enterprises.

The open source development model is one that clearly demonstrates the success of virtual teams in software development.

Risk Mitigation

Approaches to mitigate risk

Much of the upside of open source has been discussed throughout these viewpoints. The downside is the risk of dependence on an unsupported product in the long term, as well as the risk of being faced with an unfixed bug in the short term. It is important to note that open source does not equate to unsupported. In fact, commercial support is available for a broad range of open source products, often from multiple organizations that frequently include a commercial entity built around the open source project itself.

Many software products are available in a dual-licensing model: a community-supported product version available under an open source license and a vendor-supported product under a commercial license. For example, Sun's Solaris operating system is available under such a model.

With well-managed open source communities, code bugs are fixed at the same rate as commercial software; roadmap scopes and planning are similar and execution as fast. Whereas vendors may have professional services to assist implementation, open sources have equally credible third party implementers. Popular open source packages, on balance, seem to be as well supported as typical commercial products without the operating costs.

Consider these viable risk mitigation approaches to using open source: Create contingency plans to replace an open source product or prepare to use in-house resources to support the software.