he inevitable has happened. Open source has moved all the way up the enterprise stack. No longer are en-terprises contemplating whether they should use open source, but rather, *where* to use open source. In theory, an enterprise could use open source to deliver its entire application, information, and infrastructure portfo-lios.\*1 That doesn’t mean hand crafting using the LAMP stack (Linux Apache MySQL, Perl/PHP/Python), but actually acquiring solutions for collaboration, database management, content management, application infrastruc-ture, business applications (CRM, ERP, POS), and reporting. Of course, we emphasize the “in theory,” because it is too early for a full open source strategy.

However, it’s not too early to contemplate open source beyond the traditional implementations of Web server (Apache), operating system (Linux), development tooling (Eclipse, Struts, Spring, Hibernate), J2EE run-times (Apache Tomcat, Geronimo, Axis, and JBoss), and email (Sendmail).

The natural question then is: where else should you consider open source? Since every enterprise is differ-ent, we believe it would be irresponsible to make blanket statements as to where you should, or shouldn’t, be us-ing open source. Instead, in accordance with our standard research practices, we will provide insights for you to make informed decisions for your business.

In that, we are starting with this brief report on open source considerations. In the future, when it makes sense, we will be including open source solutions in our product research. Our first forays will be in the customer portal platforms and enterprise service bus spaces.

**OPEN SOURCE BASICS**

Before we get into specific considerations, we start with some basics, an open source definition, some high-level pluses and minuses regarding open source adoption, and some no-brainer uses of open source.

**What Is Open Source?**

The Open Source Initiative\*2 provides the following sound bite (OSI’s term) on open source.

*Open source promotes software reliability and quality by supporting independent peer re-view and rapid evolution of source code. To be OSI certified, the software must be distributed un-der a license that guarantees the right to read, redistribute, modify, and use the software freely.*

**FREEDOM.**The important word in the above is “freely.” To quote Richard Stallman, president of the Free Software Foundation, “‘Free software’ is a matter of liberty, not price. To understand the concept, you should think of ‘free’ as in ‘free speech,’ not as in ‘free beer.’”\*3 In other words, not all open source software is free of charge, but there are fundamental rights pertaining to the use of open source software. Those rights are, as stated above, to read, use, modify, and redistribute the software.

**OBLIGATIONS.**However, depending on the particular open source license, there are varying obliga-tions that accompany those rights. Typically, these obligations pertain to the distribution of modifications and/or derivative works. For example, if you modify software licensed under the GNU General Public License (GPL) \*4 and you distribute that software beyond your enterprise to a customer or partner, then you are obligated to con-tribute that modification (or derivative work) back to the open source community. This is true even if you distrib-ute the software for free.

**OPEN STANDARDS NOT REQUIRED.** A common misperception of open source is that open source software means open standards. This is probably due to the success (and press) around projects such as Apache Axis (Java Web Services) and Open Office (office suite using OpenDocument Format) which provide infrastruc-ture for, or an initial implementation of, open standards. However, open source code could be written in proprie-tary languages such as Visual Basic, PowerBuilder, and COBOL.

**Why Open Source?**

There are some very compelling reasons to adopt open source solutions:

1. The software produced represents the collective intelligence of a community of people who were motivated to solve a problem on their own time.
2. The initial financial outlay is significantly lower than proprietary commercial software.
3. There is greater accessibility and transparency. You can try before you commit. You can see exactly what you are getting. You can learn by looking at project code.
4. There are IT productivity gains to be had, including the availability of developer tooling, pre-built widgets, and full-blown applications.

**Why Not Open Source?**

There are also some risks in open source adoption:

1. The software is produced and maintained by a volunteer community, who may move on to solve a new prob-lem.
2. The projects (majority) are not undertaken with enterprise consumption in mind. The projects often lack pro-ductization features such as documentation, product roadmaps, release plans, administration, and management tooling.
3. There are intellectual property and legal risks that need to be recognized and understood. If a solution you use violates a patent, you may be responsible. If you modify and distribute open source software, you may have to share your competitive IP with the community.\*

**No-Brainer Uses of Open Source**

Even with the risks mentioned above, there are some no-brainer uses of open source:

1. For R&D to get familiar with a new technology.
2. Open source embedded in commercial products, such as Apache Web Server.
3. Low-cost infrastructure (Web server, application server, and database) for development and initial test envi-ronments from well-established providers such as ObjectWeb, JBoss, and Apache. However, your pre-production environment (stage, quality) should mimic the production environment.
4. Developer and administrator tooling (Eclipse, Perl, PHP) that stays within the enterprise.
5. Non-modified implementations of commercially-backed operating system distributions (Linux, Solaris 10).

**CONSIDERING OPEN SOURCE**

Given the rewards and risks presented above, what specifically should you be looking at, as you consider open source beyond the no-brainer uses? You need to look at three things: your project, the open source project, and your enterprise IT environment.

**First, Your Project. Does Open Source Make Sense?**

**RISK FACTORS.** As with any IT project, you need to ask yourself the big questions. What is the pur-pose of this project? Does it provide strategic advantage? Does it provide core (critical path) processing? What are the service level requirements? What are the security requirements? What are the compliance requirements?

If your answers point to a high-risk project, then open source isn’t the right choice, especially for an initial foray. However, you still may find that components of the project fit in the no-brainer category. Remember, in a modular, service-oriented world, you don’t have to make all-or-nothing decisions.

**LICENSE RESTRICTIONS.**If your answers don’t scream risk, and the idea of an open source jump-start (code base and low financial entry) is appealing, then you need to think about the license obligations you can live with. (Our assumption here is that your project will modify, extend, or create a larger work from the open source code.) The questions: Will you be mixing proprietary and open source code in your solution? Will you im-plement any components outside of your walls? Is there a competitive risk to your business if your code is re-leased to the community? If you answered “yes” to any of these questions, open source may still be an option, but GPL licensed code might not be

**EVALUATION FRAMEWORK.** At this point, you’ve screened your project to determine if open source is a viable consideration. Now, you need to perform normal project activities to identify requirements, cre-ate an evaluation framework, and prepare a short list of candidates. For commercial products, we recommend (and produce) evaluation frameworks with the following six aspects:

1. **Features and Functions.**The functional requirements for the solution, expressed in business terms, often in terms of customer, business, or integration scenarios.
2. **Design, Development, and Deployment.** Design, development, and deployment looks at how you deliver the scenarios. This includes tools for all involved roles: designers, developers, testers, and business and systems administrators.
3. **Management and Monitoring.**Management and monitoring of the technology solution and the business scenario.
4. **Architecture.** The objectives of the architecture evaluation are to understand how the solution works, its fit in the target (your) environment, and with your architecture. To do this, you need to look at the solu-tions organization, the interoperability of its parts, deployment environment requirements, enterprise infrastruc-ture dependencies (and conflicts), and its capabilities for quality of service and quality of protection.

**Product Viability.** Product viability criteria consider the business aspects of the solution and its supplier.

**Company Viability.** A company’s history and current financial statistics are key markers for its fu-ture viability.

For open source solutions, we recommend replacing the last two aspects (product and company viability) with an open source project viability assessment, as described next.

**Second, The Open Source Project. Is it Viable?**

As we mentioned earlier, most open source projects are not undertaken with the goal of enterprise consumption in mind. So, in your evaluation of the open source project, you need to think about how to resolve the gap between project and production-ready product. Typical items to be resolved are: documentation, installation scripts, product roadmap/evolution, patch and release management, systems management hooks, and problem/emergency support. These items might be resolved by the project community, by a commercially-oriented ecosystem, by your staff, or most likely, by a combination of all three

Specifically, as you evaluate an open source project for viability , you need to get answers to the follow-ing:

1. **Project Origin and Objective.** Did this project evolve from a group of programmers solving a problem for themselves? Is this project an industry sponsored technology reference (or proof of concept) imple-mentation? Is this project a welcome donation (Eclipse) of commercial software that will form a common founda-tion? Is this project a dump of commercial software, which has reached end of life?
2. **Maturity.** When was the project established? What is the current project release? What is the adop-tion rate? Is adoption building, level, or declining?
3. **Backing.** Is there money behind the project? How much, and from who? Who is the community host? Is the host credible?
4. **Project Leadership.**Who are the project leaders? What is their industry track record (open source, commercial, and/or enterprise)? Is the project well managed?
5. **Community Engagement.**Is there an active community of developers, testers, and end users? Are there forums, email lists, and FAQs? How often is the source code updated? Are bugs reported? Fixed?
6. **Project to Product Gap.**What is the gap? Does the project just consist of source code? Are there binaries and installation scripts? Is there documentation? Does the documentation cover the architecture, code, and/or end use? How often are patches released? Can patches be applied incrementally? How often are releases available? Is there a project roadmap? Are there administration scripts? Are there hooks for systems management? Is there a security model? Are there programming APIs?
7. **Commercial Ecosystem.\*7** Is there a commercial ecosystem––consulting, training, code valida-tion, distribution management, support––for this project? Is it cost effective? Are the firms viable? Is the ecosys-tem expanding, level, declining?
8. Code Base. What language(s) are used? Are open standards used? How is the code quality? Is there a modu-lar, extensible design? Is the architecture apparent? Are there APIs?
9. Testing Practices. How does the project test? Is it a formal process? Are testing assets distributed with the code base?
10. License. What license(s) are the project under? What are your rights and obligations? Are there dual license options (open source and commercial)?
11. Commercial Conflicts. Are there any commercial conflicts (the project, and/or your business)? Does the code base have a questionable history? Is there any pending litigation?
12. Competitors. Who competes with this project? Are the competitors open source? Commercial?
13. Chatter. What are people saying about the project? Check email lists, blogs, community boards, and trade press for participant (developer, end-user) and observer (press, analyst, competitor, partner) chatter.
14. **Third, Your Enterprise. Is There a Fit? Is It the Best Fit?**
15.
16. In this last step, you need to determine if the viable open source project(s) are a good fit with your enter-prise. While you are concerned with the normal factors of architectural, operational, organizational and budgetary fit, the lens is slightly different. Essentially, you are mixing buy and build models. There is the code base jump-start of a commercial purchase, and the ongoing care and feeding of a build.
17. **COULD YOU SURVIVE ALONE?**With this in mind, the first big question is, “If the community dried up, what would you do?” Can you support this software? Is that the best use of your resources? Would your busi-ness be at risk?
18. **CAN YOU SPAN THE GAPS?** Next, you need to address any gaps discovered between open source project and production-ready product. Does the investment (time, skills, people, and dollars) to close that gap fit your enterprise plan? Are there opportunity costs? Are you willing to make a contribution (people, dollars, code) to the open source community to bridge this gap?
19. **DO YOU FIT IN THE OPEN SOURCE COMMUNITY?** Finally, you need to consider if your enter-prise fits into the open source community. After all, they are doing the initial work! Are you willing to participate in the community, contribute to the project’s evolution? Donate back some of the less cool, but very important, production readiness deliverables?
20. **BEST FIT?** After determining the fit of the open source solution, you then need to weigh your op-tions. Does an open source solution provide a better/worse fit and value against competing commercial and in-house build solutions? As we mentioned at the onset, there won’t be a single right answer. However, there will be a best answer for each business
21. **NEXT STEPS**
22.
23. We will be incorporating the open source project viability assessment in our evaluation frameworks to fa-cilitate our, and end-user, review of open source solutions. Our first research areas to evaluate open source solu-tions will be customer portal platforms and enterprise service bus. Stay tuned!
24. \*1 For a good article on open source options, see “InfoWorld’s Special Report: Build Your Business with Open Source,” http://www.infoworld.com/reports/32SRoss.html
25. \*2 Open Source Initiative (OSI) is a non-profit corporation dedicated to managing and promoting the Open Source Defi-nition (the essential qualities of open source software) for the good of the community, specifically through the OSI Certified Open Source Software certification mark and program. The certification mark (OSI Certified) indicates the marked software is distributed under an approved Open Source License. There are more than 50 OSI-approved licenses. For more information see: http://www.opensource.org/docs/definition.php, http://www.opensource.org/docs/certification\_mark.php and http://www.opensource.org/licenses/index.php
26. \*3 Free Software Foundation Web site: http://www.fsf.org/ Richard Stallman’s Free Software, Free Society: Selected Es-says, http://www.gnu.org/doc/book13.html
27. \*4 Depending on your perspective (open source community, commercial software vendor, enterprise) the GPL is either the most community-friendly or restrictive open source license. The GPL has a copyleft restriction, which states that all software developed from GPL software must be GPL software. If you distribute these works, they must be GPL, and released to the community––all IP is public. In addition, you can’t mix proprietary software with GPL software to form a larger work. For more information see: http://www.opensource.org/licenses/gpl-license.php
28. \*5 We recommend proper due diligence, leveraging legal professionals, before embarking on initiatives that will distribute, or redistribute, code under an open source (or really any) software license. In addition, there are companies that specialize in open source risk mitigation, such as Black Duck (http://www.blackducksoftware.com/ index.html) and Open Source Risk Management (http://www.osriskmanagement.com/index.shtml).