

# The DoD and Open Source Software

*An Oracle White Paper*  
*February 2009*

# The DoD and Open Source Software

**Oracle is committed to offering choice, flexibility, and a lower cost of computing for end users. By investing significant resources in developing, testing, optimizing, and supporting open source technologies such as Linux, PHP, Apache, Eclipse, Berkeley DB, and InnoDB, Oracle is clearly embracing and offering open source solutions as a viable choice for development and deployment.**

## INTRODUCTION

When discussing Open Source Software, people often associate it with a few common characteristics. In many cases the software itself is hosted publicly by a sponsoring group or organization – often a not-for-profit organization. (The Apache Software Foundation or the FreeBSD Foundation for example). Open Source Software also brings to mind the varied community of people and organizations (commercial and/or non-commercial) that support discuss or maintain the software. But the most important thing to understand about Open Source Software is that it is a set of licensing terms and conditions, not the cost of the code itself. There is no escaping these terms and conditions. Major software vendors, those selling commercial products and those marketing themselves as Open Source vendors, are acutely aware of this reality.

Many major software companies, including Oracle<sup>1</sup>, utilize Open Source Software in their products. Most commercial software is actually *blended* software; software that is sold under different licensing terms from Open Source, but is probably built using a variety of Open Source components.

In this paper we ask two questions:

- Should the U.S. Department of Defense deploy substantial amounts of software based on a community development model, in lieu of commercial software, in support of their Service Oriented Architecture requirements?
- Is a community-based software development approach a viable strategy for large-scale Service Oriented Architecture deployments, especially in the Department of Defense?

## Software Costs in DoD Programs

The move toward information-driven warfighting doctrine has made the cost of new IT systems a major concern within the DoD. Government organizations must ultimately create a business case that will drive the software infrastructure decisions for individual programs. Carefully refining requirements to balance costs and expected innovations, acquisition officials and program managers design strategies that minimize risks. Comprehensively, the business case should be based on cost, schedule, performance and risk across the program lifecycle where:

---

<sup>1</sup> For more information on Oracle's commitment to Open Source, see: <http://oss.oracle.com/>

- Cost is the total outlay required to operate the program from inception through deployment, routine maintenance and upgrades as well as the comparative cost of decommissioning and replacement.
- Schedule is the program timetable, as measured against requirements milestones.
- Performance is the degree to which the program fulfills its stated requirements.
- Risk is the quantitative degree of probability to fail to meet stated cost, schedule and/or performance requirements.

Government organizations must build the business case to address these considerations across the entire program lifecycle, not just up-front program inception considerations. With this approach, organizations will be better equipped to develop a business case that properly assesses and measures the impact of software infrastructure choices.

Acquisition officials in the Department of Defense should be focused on the greatest return on investment. Often, even in the case of software, this does not mean buying the cheapest product, even if the license costs are negligible.

### **Understanding Software Costs**

In most cases, hard costs, such as up front license fees, support contracts and hardware account for a relatively small percent of the total cost of a software project. The remainder of the cost of a software project is soft costs such as facilities, power, cooling and labor for the development, testing of load and stress, deployment, and maintenance of the system. There is also the lost productivity when downtime occurs.

Labor is easily the largest cost factor of a software development project and is impacted by factors such as usability, learning curve, quality of tooling and the need to modify or build required infrastructure functionality. Developer productivity, technical support, quality of documentation, ease of software deployments, system administration, and system reliability all contribute to program costs. Even small improvements in these areas can have profound impacts on the total cost of the program. Open source testing tools do exist, but this is an area where commercial technologies far surpass open source alternatives. In using open source as an alternative to commercial quality software, the government pays for the responsibility of testing and integrating the major components of a large system-of-systems common in DoD.

Focusing on the easily identifiable and predictable hard costs (i.e. software licensing and support) is an easy trap to fall into and can be detrimental to the lifecycle cost of a program. Making technology decisions based on short-term savings in hard

**“In reality, most popular open source software is written by employees of the commercial entity most associated with it ... [T]here are exceptions ...but the tendency is to merge the project with the company as the company matures.”<sup>2</sup>**

---

<sup>2</sup> [http://www.wikinvest.com/concept/Open\\_Source](http://www.wikinvest.com/concept/Open_Source)

costs could negatively impact and dramatically increase the soft costs that represent the majority of the total program. The best value to the Government comes through optimizing developer productivity, providing reliable and scalable infrastructure, and reducing program soft costs.

Conversely, license costs associated with Commercial off-the-shelf (COTS) products typically represent a small fraction of expenses when analyzing program spending across the lifecycle of a program. The inflated operations support and sustainment cost can be attributed to a number of factors across the program lifecycle. These factors include, but are not limited to, process, tooling, asset and human considerations as well as a lack of long-term organization and governance planning. As a result, Government organizations continue to allocate ever-increasing resources toward sustaining existing software infrastructure and find themselves limited in their ability to introduce new capabilities. Government organizations should analyze historical spending and allocation of resources across the lifecycle of programs. Through this analysis, organizations will be able to make the appropriate decisions with respect to the processes, tooling, assets and people to support future programs. These decisions will serve as the foundation for a long-term organization and governance strategy.

### **Mission Assurance and Reliability of Software**

Perhaps the most important issue in a major Department of Defense IT system is its reliability. Reliability has many facets, but they include the robustness of the system, its ability to handle load and stress, strong information assurance features, and other factors. Mission Assurance can be expensive, as program executives are all too aware. Robust testing and certification of an end-to-end solution can be extraordinarily expensive, especially if a system is being changed frequently.

Load testing, system performance tuning, and system optimization are tasks that usually assigned to talented individuals. These skills are acquired over time, through years of experience, and these individuals are generally paid quite well. The cost differential of Open Source, community-based software development models, especially when it comes to the cost of testing for robustness and reliability under load, should not be underestimated.

Should the Federal Government pay an integrator to shoulder the load for testing and tuning Open Source software? Designing contractual terms and conditions that encompass each and every aspect of a system's potential performance characteristic needs are extraordinarily difficult – if not impossible. A certain amount of robustness must be assumed, or acknowledged as inevitable, if a system is widely used in a commercial setting.

Commercial companies are required to spend a substantial amount of money developing software that can be instrumented for load testing, designing the testing methodology, conducting the testing, and ensuring that end-to-end performance is achieved. They test their products at this level of rigor as part of their basic business model. Does the Federal Government want to be in this business?

## Free Software and Open Source Software

Richard Stallman's contribution to the debate over free vs. open source software has an interesting anecdotal component:

*This is a matter of freedom, not price, so think of "free speech," not "free beer."*

But it's not just a debate about money, or speech, or beer. The Open Source vs. Free Software debate brings the issue of licensing terms and conditions into much sharper focus. For those who are merely concerned with vendor lock-in, and look eagerly towards Open Source as means of "owning" the code they want to run, a quick glance at the Open Source Initiative (OSI) can be daunting. As of this writing, there are over seventy licensing regimes that have gone through the license approval process. Open Source is not free, nor is it easy to understand the strict legal terms and conditions associated with its use.

## Indemnification and Open Source

A key issue that should be of concern to all government acquisition executives is the issue of indemnification. When a software company provides a product to its customers, it is expected to stand by its product, and is usually legally required to do so. In the case of Open Source, there are remarkably few companies willing or able to offer indemnification for their "products" and most typically refuse to do so. This lack of indemnification for a major component of mission capability is lost in the shuffle of paperwork when a new program office makes their decisions, often without professional legal consultation on the implications of fielding a system at some point in the future that has no liability associated with the underlying software.

## CONCLUSION

Today, commercial software vendors all leverage Open Source Software. Many are the primary contributors to major open source projects. In fact, their level of expertise and rigor has helped make Open Source Software what it is today. Furthermore, COTS today is really a "blended source" model and it is blended for a reason. The skill required to successfully – and economically – blend source code into a commercially viable product is relatively scarce. It should not be done directly at Government expense.

Community development approaches to building software lack the financial incentives of commercial companies to produce low-defect, well-documented code and are not subject to the same market pressure at the software code level.

**"Every time you say, 'free software' rather than 'open source,' you help our campaign."**

**Richard Stallman**



The DoD and Open Source Software

February 2009

Author: Peter Bostrom

Contributing Authors: Umesh Vehmuri, Corbin Stark

Oracle Corporation

World Headquarters

500 Oracle Parkway

Redwood Shores, CA 94065

U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

[oracle.com](http://oracle.com)

Copyright © 2009, Oracle and/or its affiliates. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners. 0408